SOAR: Sub-Symbolic Mechanisms

Danilo Fernando Lucentini , Ricardo Gudwin (Supervisor)

Department of Computer Engineering and Industrial Automation (DCA)
School of Electrical and Computer Engineering (FEEC)
State University of Campinas (Unicamp)
P.O. box 6101, 13083-970 – Campinas, SP, Brazil

d032112@dac.unicamp.br and gudwin@dca.fee.unicamp.br

Abstract – This article has the purpose to analyze, using a sub-symbolic approach, one of the major existing cognitive architectures: SOAR. Each sub-symbolic module of the architecture will be detailed and, moreover, it will be described what are the benefits that these modules bring to the whole architecture.

Keywords – cognitive architecture, artificial intelligence, SOAR, symbolic, sub-symbolic

1. Introduction

A cognitive architecture is basically a computational model that aims to describe, as accurately as possible, the way that human beings resolve problems and learn new knowledges across different domains and knowledge bases. Cognitive architectures have evolved insofar the cognitive science itself has evolved, thus new knowledges and assumptions have been incorporated in the different existing architectures.

Many papers were published in this area making it very prolific in recent years. Some of these have tried to make a comparison among the different architectures [9, 4] saying what one architecture does and the others do not or which is fastest in a given scenario and so forth. This type of study is very important, however just looking this area by the comparativist perspective misses some fundamental aspects of human cogntive process. Ron Sun paper *"Desiderata for cognitive architectures"* [11] describes the issues and requiriments that the development of a bio-inspired cognitive architecture provides. With this biological basement, it is possible to analyze the architectures in a much wider, real and improved way. One of these requirements is the *symbolic* and *sub-symbolic* dichotomy that will be argued in this paper.

Symbols are entities that make reference to another objects, but not through a temporal-spatial relationship, but by a convention, a cause-consequence relation, a law [1]. Symbols can be handled in order to generate another symbols that will have new meanings and so on. For example: the white flag is a symbol that represents the peace, the red cross is a symbol that represents the hospitals and so on. This approach is based on Charles Sanders Peirce theory and it is the central pillar of semiotic science.

On the other hand, sub-symbolic is any processing that is not symbolic, that is, do not exist clearly this symbol-object relationship. As Nilsson [10] emphasized, the subsymbolic has a "bottom-up" style and, at the lowest levels, the concept of symbols is not as appropriate as is the concept of signal. This approach emphasizes that the main problem with the symbolic systems is the symbol grounding problem [3]. As previously mentioned, in the symbolic point of view, all the symbols are in some way associated with the object that they represent, but how this association is made? According to symbolic approach, this association is made via an adapter module that gets the input data from sensors and transforms them into symbols for future manipulation. This is the main problem: it is unclear how this adapter module works (homunculus problem). However, it is clear that this adapter module has a knowledge below the symbolic layer, that is, a sub-symbolic layer that is acquired through experience along the time and has a decisive role in intelligent behaviors.

By the 70's and 80's, prevailed in artificial intelligence the *cognitivism* (also called GOFAI[2]). This approach claims that many aspects of intelligence can be obtained simply by manipulating symbols. Many cognitive architectures arose in this period and all were influenced by this symbolic perspective. One of the major cognitive architecture that arose in that period was SOAR[1].

SOAR was originally proposed by John Laird, Allen Newell and Paul Rosembloom [8]

[1]State, Operator And Result
[2]Good Old-Fashioned Artificial Intelligence

around 1983. At the beginning, the architecture was entirely symbolic, however over the years the sub-symbolic perspective was gradually being incorporated and this will be the proposal of this paper: to expose which are the sub-symbolic components of SOAR and what are the benefits that they bring.

2. Proposal

This analysis will utilize the most recent version available for SOAR, that is, version is 9.3.1 (June 2011). In 2008, Laird has pointed [5] the structure of an extended SOAR architecture with symbolic and sub-symbolic modules. Nevertheless, as Laird said, all of the new modules have been built, but there is not a single unified system that has all the components running at once. So, based on the current SOAR version (9.3.1), the next figure was remade to ilustrate the modules available.

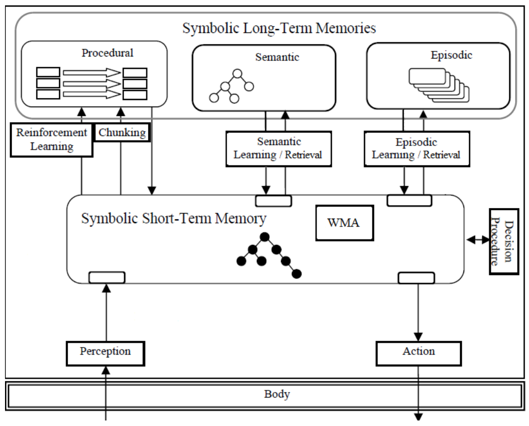
Symbolic Long-Term Memories
Procedural
Semantic
Episodic
Reinforcement Learning
Chunking
Semantic Learning / Retrieval
Episodic Learning / Retrieval
Symbolic Short-Term Memory
WMA
Decision Procedure
Perception
Action
Body

Figure 1. Soar modules in version 9.3.1

So, in the next sections the sub-symbolic modules will be detailed in order to determine what are the benefits that they bring to the whole framework. But, before detailing each module, it is necessary a brief overview about the SOAR basic components. SOAR is based on two entities: *operator* and *state*.

The state is the current problem representation and this is constituted by a set of WME's[3]. As the own SOAR manual describes: "Each WME contains a very specific piece of information (...) Several WME's collectively may provide more information about the same object" [7]. So, the WME is an entity that helps to describe an object that is in SOAR problem-solving context.

[3]Working Memory Elements

Working memory is the cradle of WME's. As said previously, the WME's discribe the current state of the problem and it is in working memory that these entities are linked to each other in order to create a full context of the problem. The working memory can be viewed as a theater of operations, because all modules are linked through the working memory (as can be viewed in Figure 1).

Operators are modifiers that make changes in the current state and they are formed by a set of rules. Each rule has two parts: the precondition and the result. In the precondition, there are boolean expressions that compare the current state of the problem, in the other hand, the result effectively changes the current state of the problem. So, for the result can be executed, it is necessary that the precondition is true.

An operator is generally described by two rules: one rule proposes the operator given a precondition set, and the another verifies if the operator is selected to execution and apply the changes in the current state. This two rules are required, because not necessarily if an operator has the precondition that matches with the current state of the problem it will be executed. SOAR just runs one operator at a time. This cycle of selecting operators is named *decision cycle*.

With such information in the hands, the SOAR sub-symbolic modules will be detailed in the next sections.

2.1. Decision Procedure

This decision procedure is divided into three steps. The first is to determine what are the operators that are candidates to execute, so SOAR will search what are the operators that match with the current working memory state. At the end of this step, there will be a set of operators that are able to execute.

The second step consists in evaluate the *preferences*. Preferences are suggestions or imperatives about the current operator, or information about how suggested operators compare to other operators [7]. SOAR will compare the preferences among all the selected operators. So, in a simple way, if there are two operators A and B and the operator A has a higher preference than the operator B, so the operator A will be selected to execute.

The third and final step consists in evaluate *impasses*. If even after the first and second steps, there are still more than one plausible operator for execution, an impasse occurs. An impasse is a substate in SOAR where the current objective is to resolve the impasse. In this substate, there are lots of information like the type of the impasse (there are four types of impasses), what are the operators that result the impasse and so on. The substate is also resolved using rules in something like: "if there is an impasse between operator A and B, operator A will have higher preference than operator B". If the substate is not resolved, another will be generated and so on until a limit is reached. With this hierarchy of impasses, it is possible to resolve impasses step by step, for example: if there is an impasse between operators A, B and C, first the impasse between A and B can be resolved and, in sequence, the winner of the first impasse against C.

2.2. Reinforcement Learning

Inspired in the behaviourist psychology, the idea is to determine what the best action to be taken in such a way that for each action executed is provided a reinforcement value which can be positive or negative. So, actions that receive positive reinforcement will tend to be executed more times.

In SOAR, it is necessary to rewrite the statement "what the best action to be taken" to "what the best operator to be chosen". Specific operators must be declared in the architecture in order to the reinforcement learning module can handle them, these operators are called *RL operators*. The RL operators are handled like any other operator in SOAR, susceptible to impasses and to the decision procedure mechanism. It is possbile to note a clear distribution of duties in the architecture: the reinforcement learning module is responsible to adjust the preferences of each RL operator and the decision procedure module is responsible to find the best operator to be executed in a given situation (as described in section 2.1.).

But, how this adjusting of preferences is done? It is described in more details in [7]; but, in short: after an operator has been executed, the agent must to provide a reward signal to SOAR through a specific WME in working memory and it is this reward signal that reports if the operator had a good or bad performance. At the beginning of every decision cycle, SOAR will read this reward signal and associate it with the last executed operator. Finally, SOAR will update the preferences of RL operators using a history of reward signals, so, at the end, the operators with the best performance will have a high preference value and hence will tend to execute more times than operators with low preference value.

2.3. Working Memory Activation (WMA)

Each WME in working memory has an activation value. The basic logic behind this module is: the activation decays over time according an equation and any time that a WME was tested by a production that fired, the WME activation value will be increased.

SOAR constantly checks each WME activation in working memory, if one of them has an activation value below a defined threshold, it can be removed from working memory.

3. Results

The sub-symbolic modules bring a new perspective to the architecture. The decision procedure is the first sub-symbolic mechanism and the core of SOAR, because it is responsible for the decision cycle and for the handling of impasse. It is important to emphasize that an impasse is not result of a bad elaboration of operators and rules. Quite the reverse, it is crucial to the architecture. For example: if there is an operator "catch a ball" and in the case that there are two balls in the current working memory state, two operators "catch a ball" will be proposed and an impasse will probably arise, but thanks to impasse it is possible to make a decision like "catch the closest ball". Therefore, this is the power of impasse: decide at running time what is the best operator.

The reinforcement learning module allows that an agent becomes an autonomous learning system, because this type of learning provides a quick adaptation in dynamic environments with independence of supervision. This kind of adaptation was not possible with the chunking learning process (the first SOAR learning module).

As Nuxoll [6] said, the WMA is very usefull for two scenarios: which stored episode is the best match for the current situation and to support forgetting in working memory. The forgetting process is crucial, because it privileges things that are more important from the others and WMA provides an artificial forgetting process. Furthermore, the WMA helps to retrieve episodes from Episodic Memory. There are a large number of episodes stored in episodic memory and one way of biasing the match to the most relevant scenario is using working memory activation.

4. Conclusion

This symbolic and sub-symbolic debate is far from reaching the end. In fact, there are some kinds of cognitive tasks that can be better resolved using symbolic representation, while others using sub-symbolic. As consequence, each approach brings pros and cons as Eliasmith and Bechtel have detailed [2].

Therefore, the new tendency is to incorporate symbolic and sub-symbolic mechanisms in cognitive architectures in order to obtain the benefits of each approach. This incorporation is already present in the new cognitive architectures and even in the classic ones (that born in the GOFAI period). So, as exposed in this paper, this incorporation can bring a lot of benefits to the architectures in order to resolve cognitive tasks.

References

[1] R.T. Craig and H. J. Muller. *What is a sign ? In: Theorizing communication: readings across traditions*. Thousand Oaks, CA: Sage, 2007.

[2] C. Eliasmith and W. Bechtel. Symbolic versus subsymbolic computation and representation. *Encyclopedia of Cognitve Science*, 2003.

[3] S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[4] T. R. Johnson. A comparison of act-r and soar. *Mind modeling - A cognitive science approach to reason-ing, learning and discovery*, pages 17–38, 1998.

[5] J.E. Laird. Extending the soar cognitive architecture. In *Artificial General Intelligence Conference*, 2008.

[6] J.E. Laird, A. M. Nuxoll, and M. James. Comprehensive working memory activation in soar. In *Sixth International Conference on Cognitive Modeling*, 2004.

[7] John E. Laird and Clare B. Congdon. *The Soar User's Manual Version 9.3.1*, 2011.

[8] John E. Laird, Allen Newell, and Paul S. Rosenbloom. Soar: an architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[9] Tijmen Joppe Muller, Annerieke Heuvelink, and Fiemke Both. Implementing a cognitive model in soar and act-r: A comparison. *In proceedings of Sixth International Workshop "From Agent Theory to Agent Implementation"*, 2008.

[10] N.J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, San Mateo, CA, 1998.

[11] Ron Sun. Desiderata for cognitive architectures. *Philosophical Psychology*, 17:341–373, 2004.