Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
2326-9865

Intelligent Multi–Memory System using- SOAR

Ms. Neha Rajan[1], Dr. SunderrajanSrinivasan[2],Prof. Dr. Sarvottam Dixit[3]
[1]Scholar , Computer Science, Mewar University Chittorgarh
[2]Professor , Computer Science, Mewar University Chittorgarh
[3]Professor , Computer Science, Mewar University Chittorgarh

Article Info
Page Number: 10849 - 10857
Publication Issue:
Vol 71 No. 4 (2022)

Article History
Article Received*: 15 August 2022*
Revised*: 25 September 2022*
Accepted*: 14 October 2022*
Publication*: 21 November 2022*

Abstract
We study the intelligent multi-memory system of agents. Several memory systems enhance intelligence of agents. In our work we consider SOAR agent and its multi-memory system. SOAR agent is allowed to play games and study how it improves its knowledge and thereby its intelligence also how multi-memory system helps SOAR agents to improve intelligence. We consider Water Jug Problem as an example in this paper and show how SOAR agent solves this problem so intelligently and gives the perfect solution in an easy way.

Keywords —Cognitive, Working memory, Episodic memory, Semantic memory, Chunking.

1. INTRODUCTION

One definition of an integrated cognitive architecture is a system that can generate all facets of behavior while maintaining consistency across domains and bodies of knowledge. This system would be made up of several modules (or components) that interact with one another to achieve a behavior. These components include knowledge representations, memory for storing material, and procedures for using and obtaining such content. Multiple types of human behavior may be explained by integrated cognitive architectures[1], and artificial intelligence can be made to emulate many of the human mind's skills, because of this.

We can say that cognitive architecture models the human brain.It suggests computer programs that mimic human behavior and, by extension, display intelligence. To break it down further, cognitive architectures are a subset of agent architectures in general. What Makes Up a Cognitive Architecture:-

- Capacity for long-term memory storage
- The computational mechanism that gathers, organizes, and stores information
- Knowledge-representation languages; means of encoding and decoding that knowledge.

We have to discuss following issues in context of cognitive architecture

1.1Representation of Information

All kinds of cognitive architectures have two kinds of memories for storing the information. They are:

- Long term memory
- Short term memory

Information is encoded as a symbolic graph structure in short-term memory, while production rules are used to store information in long-term memory. This allows for the representation of

objects in short-term memory, complete with their associated characteristics and relationships. The symbolic working memory stores the agent's evaluation of the present state of affairs, which is based on sensory input and information retrieved from the agent's long-term memory.

1.2 Memory system of cognitive Architecture

The brain's cognitive structure is mostly a compilation of many memories:-

- Working Memory
- Episodic Memory
- Semantic Memory
- Procedural Memory

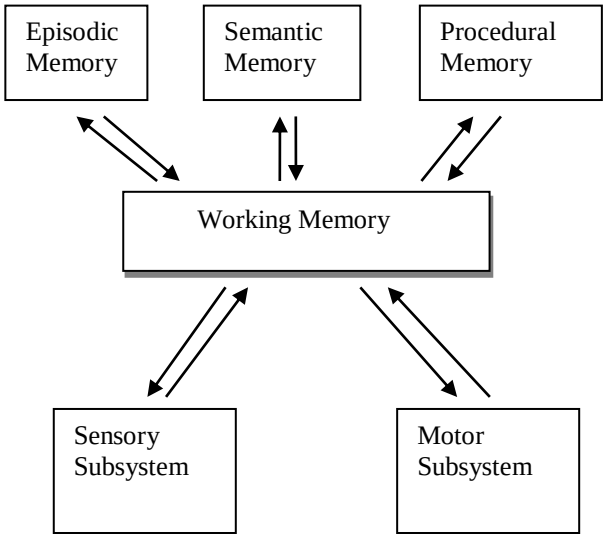


Fig 1.1 Representations of Memories

1.2.1 Working Memory

Our long-term memory, on the other hand, is almost endless and may be used to retain schemas with varied degrees of automaticity, whereas our working memory is responsible for all of our conscious operations but is restricted in size. Working memory elements (wmes)[2] are the building blocks of a working memory and consist of an id, an attr, and a value (value can be either another identifier or a constant). Our short-term memories are organized in a graph-like form. The input-link and output-link are architectural working memory structures that facilitate communication between a system and its external environment.

1.2.2 Semantic Memory

It's a recollection of broad, overarching information, not tied to a particular time or place in which it was learnt. There are logical representations of reality included inside its pages. Putting a cue in a dedicated working memory buffer allows you to access a structure stored in semantic memory. In order to extract the relevant information from semantic memory and into working memory, the cue is first employed to find the best partial match. The system allows users to record and retrieve statements about the world, such as "tables have legs" or "dogs are animals."

1.2.3 Episodic Memory

Time-based recollections are stored mostly in episodic memory. Working memory "snapshots" are stored in a sequential sequence throughout time. The agent's perceptions, motor instructions, and internal data structures together make up the agent's present state, which is what we mean by an episode. The agent's experience is encoded as a series of snapshots of pictures in working memory, and these images are all stored in episodic memory. The information that can be recovered from episodic memory is the most relevant match. You can only get one episode back at a time. Cues are created on purpose to retrieve episodes from working memory by providing a partial specification of the episode in a dedicated buffer. After forming a cue, the most promising partial match is identified (with some bias for propriety and activated working memory) and retrieved into a distinct area of working memory (to avoid confusion between a memory and the current situation). You may play back an experience as a series of episodes by retrieving the next episode in the series.

1.2.4 Procedural Memory

Specifically, it stores procedural knowledge in the form of production rules that include condition action pairs. Matching production rules to components of short-term working memory requires a root'state' match.

1.2.5 Motor Subsystem

The working memory feeds the motor processor with commands, which it then executes. The information stored in each memory is processed, and the outcomes are stored in the working memory.

1.2.6 Sensory Subsystems

The sensory subsystems are set up to absorb data from the outside world, which is then processed in working memory before being sent to the outside world through the brain's many communication pathways.

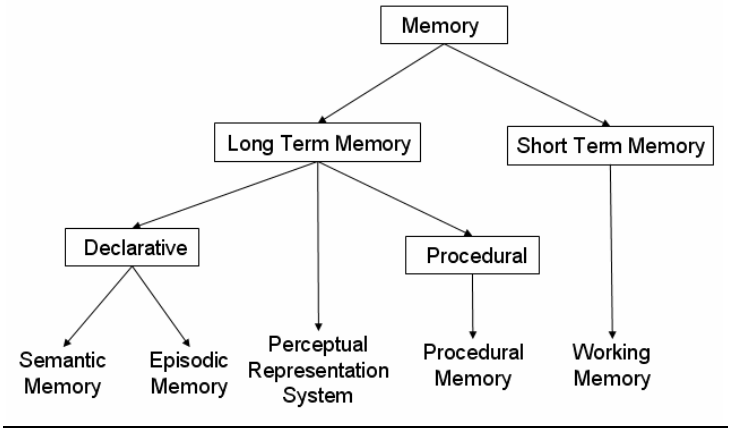


Fig 1.2 Representation of Memory system of cognitive Architecture

2. SOAR ARCHITECTURE

Place, Process, and Outcome (Soar). SOAR is one of the first suggested cognitive architectures, with its origins in classical AI. Soar's primary goal is to manage all of the skills of a smart agent by using a generic framework for learning from experience. As a result, Soar integrates a number of strategies for resolving issues and learning what it needs to know about a job before it can do it

successfully. In classical AI, it is used to learn the processes behind intelligent behavior and incorporate those processes into a more comprehensive cognitive architecture.

The Soar architecture is made up of many types of storage and decision-making mechanisms that connect perception with behavior. In addition to traditional forms of long-term memory, the Soar architecture also has working memory, which may be thought of as a kind of short-term memory. Perception brings environmental information into working memory, where it may be processed and used to inform action selection during domain-general problem solving. The three main types of long-term memory—procedural, semantic, and episodic—are where we keep our acquired knowledge. Understanding how to do a job is stored in one's procedural memory. As a kind of declarative knowledge, semantic memory[3] keeps track of big picture information about the world. All the relevant information for the present scenario is stored in the working memory of the Soar cognitive architecture. All of the operators, objectives, and state hierarchy are included. The status updates come from the individual states (and sometimes even county levels). The operator specifies the procedure to be followed when addressing a problem, while the objective guides the architecture to the intended outcome. Elements stored in working memory may prompt not only the transfer of relevant information from long-term memory but also the execution of a corresponding motor response.

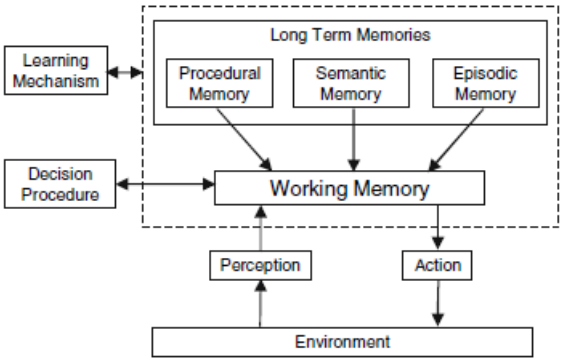


Fig 1.3 Soar Cognitive Architecture

Like a semantic network, Soar's working memory is a relational graph consisting of nodes and connections. Soar's procedural knowledge is stored as production rules, and these rules are used to generate a cue in working memory that may be later used to retrieve a specific episode. A cue is a mental snapshot of an episode's details that is stored in a dedicated section of working memory. We bring into short-term memory the scene from the program that most closely fits the cue. The similarity between two episodes increases proportionally with the amount of cue items that are shared between them. If many episodes have the same level of similarity, the most recent one is retrieved.

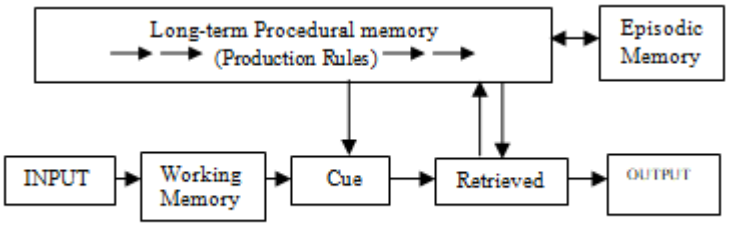


Fig 1.4A detailed processing of SOAR

In this work, we'll look specifically at how to link WMEs to the information stored in episodic memory.

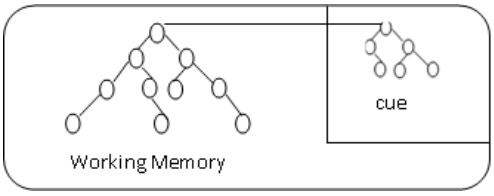
The "state" of an agent is the set of beliefs and expectations about its present environment that it has stored in its working memory. Input, internal data structure, and output are all parts of an agent's state that make up an episode.

The steps in the formation of an episodic memory are as follows:

- Encoding
- Storage
- Retrieval

Working memory is taken into account throughout the development and implementation of these stages. A memory cue is created in the "cue" region whenever an agent tries to recall a previous episode.

Individual components of the working memory, known as working memory elements (WMEs), include: (identifier-attribute-value). Since the value of one WME may serve as the identifier of another, the elements of the WME graph can be represented by a directed acyclic graph. A cue is a piece of information about an episode that can be compared to the episodes already in storage. Within the protected region of working memory, an agent constructs a cue..



A production rule fires and its accompanying action, which adds or deletes components from working memory, is carried out if the requirements of that production rule are met by the contents of procedural memory.

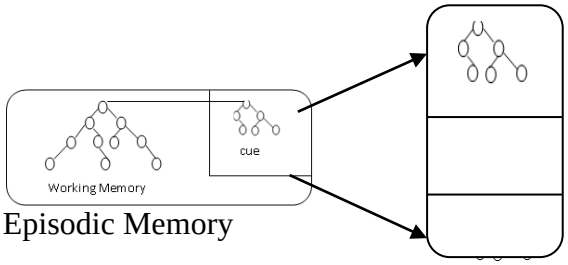


Fig 1.5 Episodic memory contains various episodes

Soar uses a method of knowledge transfer known as chunking to transform newly acquired information into creative outputs.stored in episodic memory that will help or fire in similar situation in future.

2.1 Chunking

Through the use of chunking, an agent may save computed processing as retrieved processing. In order to keep track of the outcomes of episodically-based choices, Agent employs chunking. Agent can get more done in less time because to chunking, which reduces the need for repeated episodic retrievals.

When a state produces an outcome, the 'chunking' process is activated. As a consequence, a brand-new regulation will go into effect. With chunking, we look at the components of working memory

that were used to generate the result. If any of the items in working memory are associated with super states, then those super states become conditions in the new rule. Chunking uses recursive backtracking to locate the originating rule and related working memory element for each remaining WM element, collecting all conditions along the way.

3. PROBLEM-SOLVING FUNCTIONS IN SOAR

We need to think of some operators that consciously alter internal/external state if we're going to tackle the challenge as given. Knowledge governs a sequence of operators that we call "activity"[4]. In order to choose an operator, one must apply their acquired knowledge; various changes are provided to symbolize these choices:-

- Knowledge that a certain operator is suitable for the circumstances at hand is what we mean by "Operator Proposal."
- Knowledge of how to evaluate potential operators (i.e., "Operator Comparison").
- Operator Selection — Acquiring the know-how to zero down on a single operator after conducting a series of comparisons. Ability to use an operator
- To apply an operator, one must understand how that operator affects the current state. In addition, Soar includes a fifth category of information that is related to operator choice and use in a roundabout way.
- Inferences about the state that can be drawn monotonically (state elaboration).

4. AN EXAMPLE TASK: THE WATER JUG PROBLEM

Task: Two water jugs have been provided to us, both of which are empty. The amount of water in jug A is five gallons, while the amount in jug B is three gallons. There is an endless supply of water from a nearby well, so we can fill all of our containers to capacity. We may also transfer water between jugs. The jugs are not graded at the intermediate level. The goal is to finally Jug A with 2 g water and jug B with n g waterSolution with the help of SOAR as an agent:

4.1 Initialization

- If there is currently no job, the water-jug problem should be started.
- Create a water jug identification and fill two containers with water.
- The contents and the capacity of each jug will remain the same.

4.1.2Water Jug Operator Proposals

It's important to fill in the blanks:
If the assignment calls for filling water jugs and one is empty, suggest doing so.
• Empty proposals
For the water-jug job, if a jug is found to be half full, it is recommended that it be emptied (if a jug has volume V and contents C, then it is empty (V-C)).
• Propose pour
If the water jug job requires two jugs, and one of the jugs is not full but the other is not empty, then you should suggest filling the first jug with water by pouring water from the second jug into it.

4.1.3 States

The "state" of an agent is the collection of up-to-date facts about its environment that it keeps in its active memory. The condition of the agent at the time an episodic memory is formed. Perception,

motor orders, and information from internal processing are all included in the contents of the episodic memory.

Jug A: Volume : 5 gallon
Contents : x gallon
Empty : y gallon
Jug B: Volume : 3 gallon
Contents : m gallon
Empty : n gallon

Water jug state structure

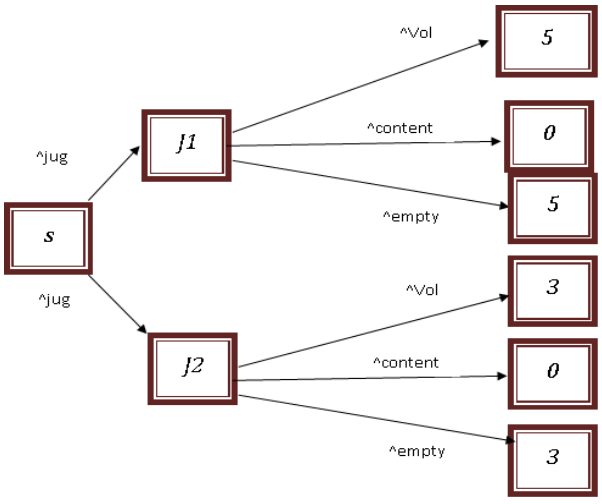
name water jug (<s> ^name water_jug)
(<s> ^jug <j1>)
(<s> ^jug <j2>)

name Jug A (<j1> ^vol 5)
(<j1> ^content 0)
(<j1> ^empty 5)

name Jug B (<j2> ^vol 3)
(<j2> ^content 0)
(<j2> ^empty 3)

4.1.3 Working Memory

An abstract view of working memory for Water Jug Problem through SOAR.



All WMEs must have some kind of connection to a state, either immediately or indirectly. The proposal imprints the operator structure (often including name and parameter) on the mind.)

4.1.3.1Initialize Water-Jug

Proposal

If no jobs are chosen, the initialize Water-jug operator will be suggested.

Application

Create an empty 5-gallon and 3-gallon water container if the initialize-Water-jug operator is used.

With every application of any operator a new WME has produced.

4.1.3.2 Operator fill jug

Proposal

Put out the fill operator if there is a jug, i.e. one that is empty.

Application

If the fill operator is chosen for a jug, the contents of that jug will be adjusted so that they fit properly.

4.1.3.3 Operator empty jug

Proposal

In the event that the jug is not empty, i.e., there is anything in it, the empty operator should be proposed.

Application

If the empty operator is chosen as the jug, the value in that jug must be set to 0.

4.2Goal Detection

For example, if you find a jug that holds 5 liters of liquid but only has 2 of them inside, you may mark the issue as solved and call it a day.

There are two basic kinds of soar design: Every time a choice has to be made, Soar's episodic memory will encode the contents of working memory as a directed acyclic graph (DAG). Such data, along with its encoding time, is permanently maintained in episodic memory.

In order to recover an episode, rules are activated to build an episodic cue, which is a directed, linked, acyclic graph that describes task-relevant linkages and properties. The "best" matching episode is determined by the cue-matching process; this episode is the most recent one that has the most structural similarities with cue leaf nodes. [6] In the next step, the episodic memory recreates the event in the active memory.

5. ENHANCEMENTS OR FUTURE WORK

- Every detail of an event is stored in an episodic memory system in an automated fashion.
- • The episodic memory system is capable of learning when to draw from and how to employ conjunctive cues.
- Learning to skip forward in episodic memory at the appropriate times allows it to make use of the memory format's temporal structure.

6. CONCLUSION

In this version of the cue matching process, episodes are retrieved if they have at least one characteristic with a cue leaf node. Additionally, the mechanism provides the "best" episode in terms of cue structure, leaf nodes, and temporal regency. Given these guarantees, encoding, storing, and retrieving all scale at least linearly with the total number of transitions between states, even in the worst scenario. To mitigate the occurrence of this worst-case scenario, the system makes use of regularities in state representations and dynamics.

In most cases, the cue [5] is a rather limited collection of traits that the agent is deliberately seeking to rediscover from a prior experience. If Agent knows the exact start or end time of an episode, they may get either episode from before or after that time. Immediately after an event, the episodic memory systems kick in to retrieve the results. Following a thorough review of all potential courses of action, the agent will choose the one with the greatest score.

Based on their present condition, agents decide what actions to do by accessing episodic memory and recalling sequences of previous states and acts to assess the likelihood of a candidate action's success in the current state. If the agent is unable to get the object, he or she will try to find an other, more favorable way, or make one up. If the recovery is successful, the agent will remember this route and continue along it.

REFERENCES

[1]John laird ,A Soar's eyeview of ACT-R,24th soar workshop ,june 2004.

[2]Michael James ,Working memory element (WME) Decay in Soar : University of Michigan.

[3] Nuxoll, Laird and james (2004), Comprehensive working memory activation in SOAR. Pp226-230.

[4]J-Y. Donnart and J.-A. Meyer, “A hierarchical classifier system implementing a motivationally autonomous animat,” in Proceedings of the 3rd Int. Conf. on Simulation of Adaptive Behavior. MIT Press/Bradford Books, 1994, pp. 144–153.

[5]A.S Maida , A Uniform architecture for rule-based meta-reasoning and representation :- case study . , IEEE Computer Society Press.

[6]Artificial Intelligence: Modern Approach by Stuart J. Russell , Peter Norvig Practice Hall Series in Artificial Intelligence.

[7] D. McFarland and T. Boesser, Intelligent Behavior in Animals and Robots. Cambridge, MA: MIT Press, 1993.