

Implementing the Mind - Cognitive architectures

WA de Landgraaf

Stdnr: 1256033

Course: AI 2001, FAAI

Group A, docent Radu Serban

Department of Artificial Intelligence

Vrije Universiteit

De Boelelaan 1081a

1081 HV Amsterdam

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation.

Abstract

In this paper two of the main cognitive architectures, Soar and ACT-R, are examined and compared. From the perspective of perception, reasoning and learning we look at the differences and similarities of Soar and ACT-R. We also take a look at how these architectures compare to the Society of Mind, how well they have been maintained and developed in recent years and in which areas they are most widely used.

1 Introduction

For many years, psychologists have been experimenting with, studying and examining the human mind. They derive conclusions based on these experiments, sometimes attempt to localize the active regions using fMRI¹ or related techniques, publish a paper on the topic and then go after a related subject.

For many years, artificial intelligence researchers have been working in a similar fashion: for instance they create a theory for face recognition or derive a syntax for a particular expert system. Be it a neural network or a genetic algorithm, researchers specialize. Rather to derive a sound conclusion for a particular problem than to generalize this problem for more real-life situations and possibly have to generalize the conclusion as well.

His final gift to the artificial intelligence community, Newell's "Unified

¹ functional Magnetic Resonance Imaging, a technique for determining which part of the brain is active at a given point in time

"Theories of Cognition" was a milestone for AI. In this book, he argued not for a particular theory of cognition, but instead for the unification of theories into a all-encompassing cognitive architecture [Hayes-Roth, 1992]. He does push forward Soar as a candidate, but he reminds us that "I am much more concerned with figuring out how a theory could put it all together, and what could be the plausible yield from doing so, than I am for arguing that Soar is the best or even the favored choice".

This paper will look into a number of these cognitive architectures and their implementations. What are the differences between the most used cognitive architectures? In a previous paper [de Landgraaf, 2004] we have seen how the Society of Mind relates as a philosophical theory, but how does it compare to cognitive architectures? Does the combining of symbolism and connectionism work in practice, or was it just a novel idea not worth exploring further? How far are these architectures from being implemented and used in practice?

To answer these questions, first we will look at what cognitive architectures actually are, after which the Soar (section 3.1) and ACT-R (section 3.2) cognitive architectures will be reviewed. A comparison will be made between these two in section 4, where we also look at the Society of Mind, which was explained briefly in a previous paper [de Landgraaf, 2004]. For this comparison, we will focus on how these architectures are able to perceive, reason and learn.

2 Cognitive architectures?

As stated in the introduction, the focus of Newell was not one single solution but instead the 'unification of cognitive theories', instead of the separated results now being obtained. The sum of the parts count, the parts alone are but mere pieces of a puzzle. Newell's quest is greatly shared by Minsky in his 'Society of Mind', where connectionistic and symbolic ideas are used together in a similar fashion. Newell's focus is not on one discipline, separate *microtheories*². He is interested in using pieces that can fit together in order to complete the puzzle of cognition.

For Newell, cognition is an architecture, “a fixed computational structure that can process variable content to produce the desired cognitive behavior” [Hayes-Roth, 1992], so much is clear. However, only by producing the behaviors which are related to cognition can a theory claim to explain them. A sufficient architecture for Newell consists of more than a small number of laws, it would be a “carefully designed and coordinated system of mechanisms working in concert”.

A cognitive architecture strives to combine different AI and psychological theories together in a modular way in order to create intelligent systems able to cope with real-world problems. In other words, a cognitive architecture is a theory on how human cognition works. These architectures can then be compared in psychological experiments to how they differ from how a human being acts. They are not just useful for psychologists and AI researchers, on the contrary

² Theories that are but small pieces of the big picture, derived independently from each other and thus aren't able to fit in together.

even. In the field of Human-Computer Interaction, techniques are used that stem directly from Soar (GOMS, for instance). In education these models are used in order to estimate the difficulties for students in learning a particular subject. In neuropsychology they are used to help interpret fMRI data.

3 Overview of Architectures

In this paper we are focusing on Soar and ACT-R. These aren't the only two cognitive architectures. In fact, quite a few working cognitive architectures have been developed, also including for example the EPIC and LICAL architectures. These architectures all (partially) fit the description Newell gives and all are heavily influenced by his work. However, we will focus on Soar and ACT-R as these are the oldest, most general, most widely-used and most complete implementations. EPIC is very focused on perception/motoric behavior (but lacks any form of learning), LICAL on the other hand is specialized for text comprehension and classification. EPIC has been integrated in both Soar (Soar-EPIC) and ACT-R (ACT-R/PM) in the last few years.

3.1 Soar

“Historically Soar stood for State, Operator And Result because of all problem solving in Soar is regarded as a search through a problem space in which you apply an operator to a state to get a result” [Soar FAQ]

Even though Newell wrote his book at the end of his life, he began

developing Soar in the early 1980's together with a few of his students. The focus of a cognitive architecture is to model cognitive behavior, therefore they had the following criteria [Lehman et al, 1993]. A cognitive architecture is:

- Goal-oriented. The implementations should focus on goals and subgoals in order to accomplish tasks.
- Able to describe complex environments. In order to accomplish goals, we must be able to perceive objects around us.
- Able to process large amounts of knowledge. In order to understand our environment, we must also be able to know what these objects are, what we can do with them and other properties.
- Able to generalize objects. What makes a chair a chair? We must be able to group objects with similar properties and identify other objects in the same groups.
- Flexible. When we run into problems, we should be able to deal with them.
- Able to learn. How other than through learning can a human gain knowledge?

3.1.1 Goals

In Soar, *problem spaces* are one of the key issues to solving goals. Each goal is composed of a number of subgoals and choices, this together creates a broad amount of ways to accomplish a goal. Even though these might not be recognizable on a biological level, Lehman et al argue that these constructs are available on a higher level in the mind. A cognitive architecture doesn't have to consider the smallest parts of the human brain.

When choosing different paths in the problem space, knowledge comes into place. This is called *the principle of rationality*. If you have knowledge that a certain path leads to a goal, then simply take that path.

3.1.2 Perception

Soar describes the *Perception/Motor Interface* for “defining mappings from the external world to the internal representation in working memory, and from the internal representation back out to action in the external world” [Lehman et al, 1993]. Thus, PMI is the interface between our working memory and the external world.

3.1.3 Knowledge

This leads us to the use of knowledge. Soar keeps semantic knowledge in the architecture's long-term memory (LTM). Knowledge applicable for the fulfilling of the current goal is in working memory. All data in working memory, all input from sensors, operators and goals together is called a single state of a Soar agent. Elements in working memory are the context of the problem at hand, these elements end up either through perception or associations acquired during the decision cycle. The decision cycle's purpose is to choose a viable operator related to the elements in working memory which leads to the fulfilling of a (sub) goal.

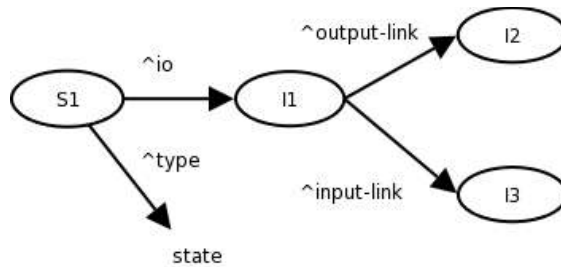


Figure 1, Soar working memory

In Figure 1, we have a part of the working memory. S1 is an *identifier* of type state, and has as *attribute* io (attributes in Soar are always prefixed with a ^). Each identifier-attribute value is called an element.

Knowledge rules, or productions, are implemented in Soar using if-then rules. How these rules are used to select operators is described in the next section.

A Soar rule uses the following syntax:

```

sp { rule*name
    (condition)
    (condition)
    ...
    -->
    (action)
    (action)
    ... }

```

Every rule has at least one condition and starts with sp (Soar production). When all conditions match (all conditions are true), all actions are executed. This is similar to (other) rule-based systems.

3.1.4 Reasoning

The decision cycle is split up into two parts: elaboration and decision. The former activates all associations which connects the working memory with long-term memory. Certain associations are stronger because they are activated sooner or multiple times, these preferred chunks are then placed in separate slots of the working memory and then evaluated during the actual decision cycle. Using these options, a number of possible operators (actions) are evaluated and the stronger the association between a slot and an operator (which also are in working memory), the more likely it is that this action takes place. A stronger association exists when there is a smaller distance in hops between the chunk and the operator compared with other operators. In the end the working memory of the single slot is altered via an operator; either a new goal, a new problem space, a new state or a new operator is placed in the working memory. A schematic overview of this process can be seen in Figure 2.

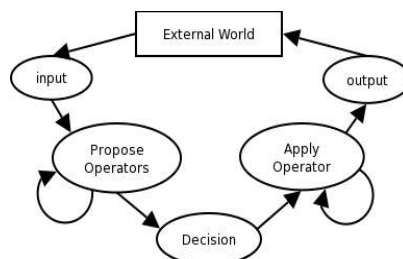


Figure 2, Soar decision making model

An example:

```
sp { propose*hello-world
    (state <s> ^ type state)
    -->
    (<s> ^ operator <o> +)
    (<o> ^ name hello-world)
```

```

}

sp { apply*hello-world
    (state <s> ^ operator <o>)
    (<o> ^ name hello-world)
-->
    (write |Hello World|)
    (halt)
}^3

```

These two productions have the following result: propose*hello-world is executed because currently there is a state identifier in working memory. This production proposes an acceptable operator, who's name attribute is hello-world. The decision procedure identifies this preference and finds a rule which describes an operator and has as name attribute hello-world. Coincidentally, our second rule apply*hello-world fits this description, fires and writes 'Hello World' to the screen and exits. It would not be valid to use <o> instead of 'Hello World' in the first action of this example, <o> by itself is an operator, however it would be valid to use <o> ^name, as this is the same as 'hello-world'.

What happens when a decision can't be made? What happens when there is a tie between operators? In that case an *Impasse* structure comes into place, with the goal of resolving which of the two have been most successful. It would for example take into account the history (Did that work last time I used it?).

3.1.5 Learning

³ This example has been derived from the Soar tutorial. See the Soar website in the references section for further details and the complete syntax of Soar

In Soar, learning is simply the adding of new associations to LTM from perceptions and derivations. It eases the amount of episodic memory necessary to derive new conclusions. Learning generalizes, combining what happens now and what happened in the past. These combinations are called *chunks*, and the process creating chunks is called *chunking*. Chunking occurs when impasses are solved: we have derived a solution for considering an option, this option is then stored in LTM. In Soar, chunking is the basis of all other types of learning. Besides learning, Soar is also able to forget chunks by using destructive operations.

3.2 ACT-R

Initially, ACT stood for 'adaptive control of thought' and stretches back to the ACT Theory [Anderson, 1976], however many different models have been derived from this theory. Another acronym specifically for ACT-R is 'Atomic Components of Thought – Rational' based on a recent book [ACT-R FAQ]. The ACT theory and the resulting architectures have always been based on psychological and biological foundations. As such ACT*, one specific framework designed by Anderson in 1983, is still actively used in HCI⁴ studies up to this day.

ACT-R 5.0 is the latest cognitive architecture in a long line of ACT-R versions.

Even though Newell pushed forward Soar as a possible candidate as a cognitive architecture, ACT-R has been heavily influenced by the 'Unified Theories of Cognition'. As Soar, it is made up of separate interconnected components called modules. [Anderson et al, 2004]

⁴ Human-computer interaction

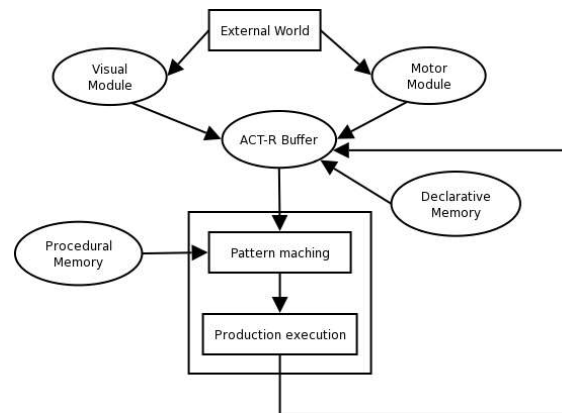


Figure 3, overview of a number of ACT-R modules

3.2.1 Goals

The Goal module is, similar to the Soar version, where the system keeps goals directly accessible to the working memory (in ACT-R, the working memory is referred to as the buffer). As in Soar, subgoals are created and stored when a goal can't be accomplished directly. As stated in [Anderson, 2004], from recent psychological findings it appears that different parts of goal-orientated activation (areas that keep track of goals, representations of problems) are located in different locations, as such there is discussion to split this module into separate parts. Goals are placed upon one single stack and have attributes and slots, which are occupied by chunks defining the nature of the goal. If a goal has been fulfilled, it is removed from the stack and the next goal is ready to be met.

3.2.2 Perception

An additional module extension to ACT-R, ACT-R/PM splits perception into two phases: 'where' and 'what'. In the 'where' phase, locations and primitive features of all objects are gathered. In the 'what' phase, specific features are collected. This is consistent with psychological

theories on perception where 'where' is located in the dorsal area of the brain and 'what' is located in the ventral system. ACT-R also uses the term *chunk*⁵ to describe a piece of declarative information, all these chunks together form the current perceptions available to the system. 'Where' is pre-attentive and provides pop-out features, 'What' is controlled by attention: only by focusing attention can the system derive detailed information on objects and identify them.

3.2.3 Declarative Memory

ACT-R splits memory up into a declarative and a procedural module. In the declarative memory module chunks are the way knowledge is stored. Chunks in ACT-R are specified in the following way:

(b ISA count-order first 1 second 2)

(c ISA count-order first 2 second 3)

(d ISA count-order first 3 second 4)

(e ISA count-order first 4 second 5)

(f ISA count-order first 5 second 6)

These chunks specify a count-order, linking together two numbers at a time. The first item in each chunk is its name, which can be chosen arbitrarily. After that, each pair is of the type slot-fact.

(p1 ISA property object shark attribute dangerous value true)

(p2 ISA property object shark attribute locomotion value swimming)

(p3 ISA property object shark attribute category value fish)

⁵ Please note that the use of chunks is different in Soar and ACT-R. This is a cause of confusion in some other articles, however having a common vocabulary is something the author finds admirable and is an indication that there is a will to work and think along similar lines

These types of chunks are of a more semantical nature, adding attributes to a shark object.

(first-goal ISA count-from start 2 end 5 step start)

This is a special type of goal chunk which is used in the goal module described in the previous section. Note that this is a possible goal, where the objective would be to count-from 2 until 5. The step slot is included to limit the states in which this goal is in. Later on, we can set this goal to counting, or stop it once it has fulfilled its goal.

Chunks are activated using the following activation equation:

$$A_i = B_i + \sum_j W_j S_{ji}$$

The activation of chunk i is A , where B is the base activation of the chunk, W are the attentional weights of elements j and S being the strength of association between the elements j and the chunk i .

The attentional weights W are calculated using $(1/n)$, depending on the number n of sources of activation. S depends on the number of facts associated to j . The base level of activation depends on practice and decay (time since the last practice) of the activation.⁶ The probability of activation of a chunk depends on a threshold calculated using a sigmoid equation:

$$P_i = 1/(1+e^{-(A_i-r)/s})$$

where r is a fixed threshold and s is a noise cancellation constant.

⁶ For further details on the activation function (and references to alternatives) see [Anderson, 2004]

Using these two equations, one chunk is selected and activated amongst many alternatives.

3.2.4 Procedural Memory

Like Soar, an ACT-R model has a large number of production rules (or productions). Like declarative memory, they are subject to change. Each rule is evaluated according to the context with the following equation:

$$U_i = P_i G - C_i$$

where P_i is the probabilistic chance of rule i being able to achieve the goal, G being the value of that goal. C is the cost of using this rule. Both P and C are learned for varying experiences with former usage of the production rule, similar to Bayesian chances (probabilities which depend on previous accuracy). However, depending on the goal at hand rules will be chosen, reflected in $P_i G$, more randomly in some situations than others. These exact rules aren't fixed in the ACT-R architecture, however the choice for this type of rule selection is largely derived from the psychological results that people base their actions on how successful they were in former situations. Another related detail is that the splitting of procedural and declarative memory is based on neurological findings; similar actions like production rule selection appears to be taking place in the Basal Ganglia and is especially involved when learning new production rules [Anderson, 2004].

The learning of new production rules is still a topic under development, with ACT-R learning derived from Soar's chunking mechanism.

Recently 'production compilation' has been developed for ACT-R which works in a similar fashion to derive a new production rule from two old rules, combining these two while having the same results. There have been related procedural learning methods in the past, however most often production rules were specified beforehand.

An ACT-R production rule is similar to Soar:

```
(p name
  buffer conditions
==>
  buffer changes
)
```

The buffer conditions are formatted in the following way:

```
= goal>
  ISA      count-from
  number   =num1
= retrieval>
  ISA      count-order
  first    =num1
  second   =num2
```

The variables goal and retrieval are bound by the goal and retrieval slots in the buffer. num1 and num2 are local variables, set for this rule only. In this case, our goal is to count-from a certain number, namely num1. We have in our declarative memory count-order chunks and this production rule fires when we have the same number in the 'first' slot of a count-order.

The buffer changes, or actions:

```
= goal>  
  start    =num2  
+retrieval>  
  ISA     count-order  
  first   =num2
```

Here we have the changes our rule makes to the buffer. Of the goal buffer, we change the start slot to num2. In our retrieval buffer we request (note the plus instead of the equals sign, which is the default) a chunk of type count-order that has in its first slot a number equal to num2.⁷

4 Comparison of architectures

As is clear, there are quite a lot of ideas which have been exchanged between Soar and ACT-R. The same has been happening between other architectures, this alone is an indication of a different mentality compared to, for example, pure connectionistic or symbolic projects. However, there are clearly large differences between these two architectures which will be highlighted below.

4.1 Foundations

When comparing architectures, be they ancient buildings or ones of a more cognitive nature, we have to start at the foundations. A choice of foundation frequently decides the outcome, as when constructing a

⁷ This example is meant to illustrate how ACT-R structures its memory types and is not a complete working example. Please see the ACT-R tutorial, from which these examples have been derived, for further details.

building even the highest peak has to be supported by an underlying structure.

In this aspect, it is clear that Soar and ACT-R are both built upon different foundations. Soar is ultimately a symbolic system, strongly related to semantical connections and tree-searching techniques. Problem spaces and knowledge are the basis of Soar, with a number of modules active in searching goals and solving uncertain decisions using the vast amount of knowledge.

This is in stark contrast to ACT-R, which is primarily based on connectionistic-related foundations. The adaption and learning phases of ACT-R resemble neural networks, and this is no coincidence. ACT-R is strongly biased towards (neuro)psychological findings and as such emphasizes the need for comparable computational neural activity. ACT-R too is goal-orientated however, but this and the use of chunks are the two of only few similarities between these architectures. It is clear that the similarities in syntax and the use of Common LISP as a means of describing both knowledge and rules stems from the fact that this language was heavily used during the early years of AI; current efforts in the defining of semantics is focused on markup languages like XML, RDF and OWL. The adoption of one standard could be the cause of better integration of Soar and ACT-R, at least on a level of declarative knowledge.

4.2 Learning

Learning in Soar is actually only a by-product, it is only done when decisions can't be made and a stand-off (impasse) occurs. Again, this is in stark comparison with ACT-R, where the tuning of probabilities is done after each production rule. Applying a certain rule has a consequence: it's success or failure will depend the likelihood of it

being used next time. Knowledge in ACT-R is not merely the placement of symbols, but consists of a number of additional variables added to simulate the brain at the cellular level.

4.3 Perception

On the area of perception, both architectures only had rudimentary support until EPIC was developed. Originally developed for human-systems interaction, EPIC's superior visual, auditory, vocal and manual motor processors have more or less become the default P/M interface for both Soar and ACT-R.

4.4 Development

Both projects are under quite a large amount of development, with the latest Soar release from November 2003 and the latest ACT-R 5.0 release from April 2004. Both projects have stood the test of time, in that after 20-30 years they still are being expanded. From a technical point of view Soar has been keeping up better than ACT-R; Soar's internal kernel has been reimplemented in more modern languages than Common LISP although it continues to use a similar syntax from the users perspective. There is a fork of ACT-R in the Java language called jACT-R, however it has yet to be pushed forward as a replacement of ACT-R and is in an early state of development. Both ACT-R and Soar are available freely under similar liberal licenses (ACT-R is licensed under the GNU LGPL, Soar is licensed under a BSD license). Both architectures have a large amount of researchers and institutes behind them, especially in the psychological field, but less in the AI community itself. Many publications from a large variety of

research fields have been devoted to either projects, this together with ample manuals and tutorials provides a large body of documentation one can use to work on similar or extended projects.

4.5 Society of Mind?

How do our two main architectures compare to the Society of Mind? The Society of Mind had as its foundations the use of K(nowledge)-lines, which activate other societies of agents, polynomes, which link a multitude of agencies together in order to produce or perceive a single notion of an object, and pronouns which in effect create new links for the working memory. Both Soar and ACT-R provide the activating of other chunks of knowledge, however this is far more elaborate in ACT-R. Soar on the other hand does have a very strong knowledge-based slot system, which is very like the notion of (uni)frames in the Society of Mind, however this is to a large degree comparable with recent versions of ACT-R.

5 Conclusion

What are the differences between the most used architectures? How does the Society of Mind compare to cognitive architectures? Does the combining of symbolism and connectionism work in practice, or was it just a novel idea not worth exploring further? How far are these architectures from being implemented and used in practice?

On the whole, ACT-R seems more like the Society of Mind in its mixing of symbolic and connectionistic techniques for the describing of

knowledge and manners of activating linked knowledge. The Society of Mind argued for a better look at how humans learn and act: ACT-R is the implementation of many of these arguments.

Both architectures however haven't truly explored the idea of growing societies of agents, in this sense the creating of modules by the system itself. ACT-R is very flexible when it comes to adapting rules and Soar's learning module derives new production rules, however both architectures are still fairly static. This is no surprise, as the growing of extra modules is a very complicated task indeed, yet this possibly is an area where a lot can be gained. When given a new interface to deal with, the cognitive architectures are able to deal with this given the right rules. Yet, an architecture will not, by itself, attempt to use this interface if it doesn't have a carefully constructed module which makes the interface available to the rest of the architecture. Experimenting and growing are key components of the Society of Mind, the current cognitive architectures fall short in this area. Also, there is a large duplication of efforts. Having both architectures cooperate on a semantic level would be an interesting way forward and maybe make hybrid architectures possible.

ACT-R has a number of advantages in theory over Soar, being that it is more in contact with the biological nature of cognition. Both architectures are used in HCI research in order to model human nature, but again ACT-R seems to be the best option for research in this field because of the close connections to our neural building blocks. Both architectures however are also showing their age; semantic knowledge has gained a lot of terrain in the last 10 years but this doesn't show in both implementations. Neural networks are progressing rapidly. Genetic algorithms are continuing to make sense out of chaos (or the other way around). Cognitive architectures have come a long way to

date, however without renewed interest and expansion they could quickly become redundant. For psychological and human-computer interface researchers the current cognitive architecture's implementations offer unique possibilities, however widespread use in the AI community is still far off. Why? The author suspects it has to do with the detailed nature of AI research now underway. The time for theorizing and separate implementations is over. Let's start integrating.

Bibliography

Anderson, 2004

J. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, Y. Qin (2004) An Integrated Theory of the Mind. Psychological Review.

de Landgraaf, 2004

W.A. de Landgraaf (2004), Artificial Consciousness, the search for what makes us human.

Hayes-Roth, 1992

B. Hayes-Roth (1992), On Building Integrated Cognitive Agents: A Review of Allen Newell's Unified Theories of Cognition. Knowledge Systems Laboratory.

Laird, 1996

J. Laird, & P. Rosenbloom (1996). The evolution of the Soar cognitive architecture. In Steier, D. M., & Mitchell, T. M. (Eds.), Mind Matters: A tribute to Allen Newell, pp. 1--50. Erlbaum, Mahwah, NJ.

Lewis, 2001

Lewis, R.L. (2001) Cognitive theory, Soar. In International Encyclopedia of the Social and Behavioral Sciences. Amsterdam: Pergamon (Elsevier Science).

Pollack, 1992

J. Pollack (1992) On Wings of Knowledge: A review of Newell's Unified Theories of Cognition, Artificial Intelligence. 59, 355-369 .

Soar website & FAQ

<http://sitemaker.umich.edu/soar>

ACT-R website & FAQ

<http://act-r.psy.cmu.edu/>