J. of Japanese AI Research 24(2). 241-249.

招待論文

Invited Paper

Two Cognitive Modeling Frontiers

Emotions and Usability

Frank E. Ritter
College of IST, Penn State
frank.ritter @ psu.edu / www.frankritter.com

keywords: cognitive science, cognitive engineering, hybrid architectures, emotions, Soar, ACT-R, CoJACK

Summary

This paper reviews three hybrid cognitive architectures (Soar, ACT-R, and CoJACK) and how they can support including models of emotions. There remain problems creating models in these architectures, which is a research and engineering problem. Thus, the term cognitive science engineering is introduced as an area that would support making models easier to create, understand, and re-use.

1. Introduction

This paper reviews how to include emotions in hybrid cognitive architectures. Hybrid architectures combine the advantages of symbolic architectures and lower level, sub-symbolic architectures. The symbolic processing based on what is essentially an AI or agent architecture allows the models to perform relatively large and complicated tasks. The sub-symbolic components support modifying the performance in subtle but important ways. These architectures are increasingly available and usable.

There remain problems creating, reusing, and understanding these (and other models). This review is intended to inspire further research and to introduce the area of cognitive science engineering.

The problems noted here are large enough that others can pursue them as well. Indeed, they will require multiple researchers, there are plenty of applications, and we would appreciate the help.

This review represents my thinking, and draws too much on projects that I am working on. Longer and more complete reviews [Pew 98, Pew 07, Ritter 03b] include further work from a wider geographical and intellectual area.

2. Hybrid architectures

Cognitive architectures are an approach to describing the mechanisms of cognition that are fixed across users and across tasks [Anderson 07, Newell 90]. Cognitive architectures use cognitive psychology concepts like working memory and implement them in a computer program that takes in knowledge representations (typically rules, but it could include other types of task knowledge such as plans) to produce behavior. They are typically implemented as a computer program that applies task knowledge to produce behavior. Thus, the language has a representation for knowledge and a way to apply it. The application of knowledge in a cognitive architecture is unlike an AI system in that it is intended to take time, suffers from simulated limitations of human knowledge application, and often includes ways for errors to arise from not applying the correct knowledge or not applying the knowledge correctly.

Hybrid architectures can be defined as cognitive architectures with two types of knowledge representations that are at different levels or are of fundamentally different types. These architectures can be created in three ways. One way is by adding symbols to a connectionist representation (e.g., [Sun 94, Touretzky 86]). Work in this area provides a way to either create symbols or structures or to find them amongst the connectionist representation.

Another way is by adding sub-symbolic representations to a symbolic architecture, for example, declarative memory strength in ACT-R and reinforcement learning in Soar 9 [Laird 08]. Here, the weighting given to the rules and declarative memory elements is essentially a way to create a symbolic architecture that has changes that are small and gradual.

Most hybrid architectures are realized in these two ways, but there are also examples where one of the levels is a genetic algorithm, fuzzy logic, or other representation. Reviews of hybrid architectures (e.g., [Kandel 92]) and further examples of hybrid architectures are available at conferences.

I review three high-level hybrid architectures briefly to support readers not familiar with cognitive architectures or with these in particular.

2·1 Soar and ACT-R

Soar and ACT-R are two of the most commonly used cognitive architectures. They both provide a conceptual framework for creating models of how people perform tasks. This review draws upon and updates a previous description of Soar and ACT-R [Ritter 03b].

Both Soar and ACT-R are supported by computer programs that realize these theories of cognition. There are debates as to whether and how the theory is different from the computer program, but it is fair to say that they are highly related. It is generally acknowledged that the program implements the theory and that there are commitments in the program that must be made to create a running system that are not in the theory—places where the current theory does not say one thing or another.

As cognitive architectures, their designers intend for them to model the full breadth and width of human behavior. Cognitive architectures, including the ones discussed here, do so to a greater or lesser extent, usually with the areas covered increasing monotonically over time.

Further comparisons of Soar and ACT-R are available [Johnson 98, Kennedy 06, Ritter 03a], and online from the Soar and ACT-R home pages and FAQs, and the Soar Wiki.

§ 1 Background of Soar and ACT-R

Soar and ACT-R are based on sets of theoretical assumptions, reflecting, largely, their different conceptual origins. Soar was developed by combining three main elements: (a) the heuristic search approach of knowledge-lean and difficult tasks; (b) the procedural view of routine problem solving; and (c) a symbolic theory of bottom-up learning designed to produce the power law of learning. However, many of the constraints on Soar's theoretical assumptions consist of general characteristics of intelligent agents, rather than detailed human behavioral phenomena. Soar's outlook is more biased towards performance because it arose out of a more AI-based tradition.

In contrast, ACT-R grew out of detailed phenomena from memory, learning, and problem solving [Anderson 83, Anderson 07, Anderson 73]. ACT-R is thus suited more for slightly lower-level phenomena, and is more suited for predicting reaction times, particularly for tasks under 10 s. ACT-R's outlook is more biased towards predicting reaction time means and distributions because it arose out of a more experimental psychology tradition.

These differences are relative; both architectures have been used for both high and low level models, with attention paid to both performance and time predictions. Plenty of examples are provided on their home pages.

§ 2 Similarities between Soar and ACT-R

Soar and ACT-R can be seen as similar in numerous ways. They both have two kinds of memory, declarative (facts) and procedural (rules), although they represent these items differently. Typical instantiations of them now have input provided through a model of perception and output buffered through a model of motor behavior [Byrne 01, Chong 01, Ritter 00].

Both Soar and ACT-R model behavior by reducing much of human behavior to problem solving. Soar does this rather explicitly, being based upon Newell's theory of problem solving, the Problem Space Computational Model (PS CM), whereas ACT-R merely implies it by being goal directed.

In both architectures memories are conceptually infinite, with no provision being made for the full removal of any memory item in ACT-R, although elements become less active with time and lack of use. Both ACT-R and Soar remove declarative memory associated with goals, which therefore can be seen as a type of short-term memory. For procedural memory, rules may only be added to both architectures but not removed.

The course of processing involves moving from an initial state to a specified goal state. ACT-R keeps only one goal active (as a memory structure, there may be multiple internal and external situations where the goal is satisfied), whereas Soar may have several of them arranged in a stack.

Both ACT-R and Soar maintain a goal hierarchy where each subsequent sub-goal becomes the focus of the system. In ACT-R, these must be satisfied in a serial manner, and in the reverse of the order they appear in the hierarchy. Soar generally proceeds in a serial way as well, but is capable of removing (or solving) intermediate subgoals should the current problem solving resolve a subgoal that is much higher in the goal hierarchy. This difference makes ACT-R potentially less reactive, although work is in progress to make ACT-R more reactive [Lebiere 01, Salvucci 06].

§ 3 Differences between Soar and ACT-R

There are also fundamental differences between Soar and ACT-R. Soar only moves between states through changing the state as part of a decision procedure, which rules can vote on but cannot directly cause. In Soar, when no more productions can fire, an operator is selected or a state is modified. This whole process is called a decision cycle. Where an operator cannot be selected (e.g., none are proposed), a sub-goal is created with a goal to choose the next operator. Movement between states is done in ACT-R by firing productions, which may directly change the state and goal stack. Where there is no matching rule, many versions of ACT-R just stop.

Learning in earlier versions of Soar only occurred for production memory. New rules are created by the architecture whenever a sub-goal is resolved, so that when next encountering the same situation, the new production fires without the need to enter the sub-goal. This type of information can include which operator to select or how to implement an operator. These rules tend to be atomic, and in nearly all cases until recently can be seen as immediately fully learned. This learning mechanism (procedural chunking) can implement a wide range of learning effects, including long-term declarative memory learning. Soar 9, the latest version, includes reinforcement learning, a sub-symbolic learning mechanism.

ACT-R learning involves both declarative and procedural memory. When rules fire they can become stronger, and as declarative memories are used more they are strengthened. Each potential production also has an expected gain based on its probability of success, its cost, and the current goal's value. The production with the highest expected gain is selected when several productions are matched. The more often the production meets with later success, the higher this probability for the rule will become.

A rule learning mechanism is less often used in ACT-R models, and when it has been used, the resulting rules are typically created in a nascent state such that they have to be created several times before they are fully learned (e.g., [Jones 00, Taatgen 03]).

2·2 *CoJACK*

CoJACK combines BDI agents' high-level knowledge representation and usability with several aspects of low level cognitive architectures, including traceability, processing time predictions, and errors [Evertsz 07]. CoJACK attempts to include lessons from ACT-R and from Soar [Norling 04]. CoJACK also includes important aspects of macrocognition, including variability in its performance of a long-term task. Its behavior and the effects of moderators on performance have been demonstrated in a simple adversarial environment. CoJACK provides lessons for other architectures, including how to define, measure, and control variability that arises from individual and temporal aspects of cognition; the importance of situation awareness (SA) and knowledge representations necessary for complex SA; and some of the complexities that will arise when we try to add SA to other cognitive architectures.

§ 1 JACK

CoJACK is based on JACK, the Java Agent Construction Kit [Busetta 99]. JACK uses the so-called Belief-Desires-Intentions (BDI) model, a paradigm that explicates rational decision-making about actions. The BDI paradigm was developed in response to a perceived problem with existing AI approaches to planning. Agents are typically situated in a dynamic environment and must constantly review their goals and activities.

The key programming constructs of JACK are:

Event—the central motivating factor in agents that are generated in response to external stimuli or as a result of internal computation.

Plan—a procedure that defines how to respond to an event. When an event is generated, JACK computes plans that are applicable to the event. The agent selects the plan that will form its next intention. Plans have a body that defines the steps to be executed. Non-deterministic choice allows the agent to try alternative plans.

Beliefset—the agent's declarative beliefs in a first order, tuple-based relational form. Beliefsets are analogous to working memory.

Intention—the currently active plan instantiations, i.e., the plan instances that the agent is committed to. A plan becomes an intention when the agent instantiates it with symbolic references.

JACK represents and executes plans in a way that maps well to SMEs' (Subject Matter Experts) introspections about their own reasoning processes. Figure 1 shows that modellers can specify behavior graphically, which is useful for visualizing the logical structure of tactics and discussing them with SMEs. Therefore, a key design goal for CoJACK was to retain JACK's high-level representation and ease-of-use.

§ 2 CoJACK Extends JACK with Timing, Errors, and Moderators

CoJACK augments JACK with a set of constraints and parameters, as well as a moderator layer. The major activities in the JACK architecture, such as adding a belief or instantiating a plan have a (simulated) time cost associated with them that is parameterised.

CoJACK adds noise to the decision processes, which then affects the beliefs retrieved; this implements effects such as failure to retrieve a matching belief and retrieval of a belief that only partially matches. A similar mecha-

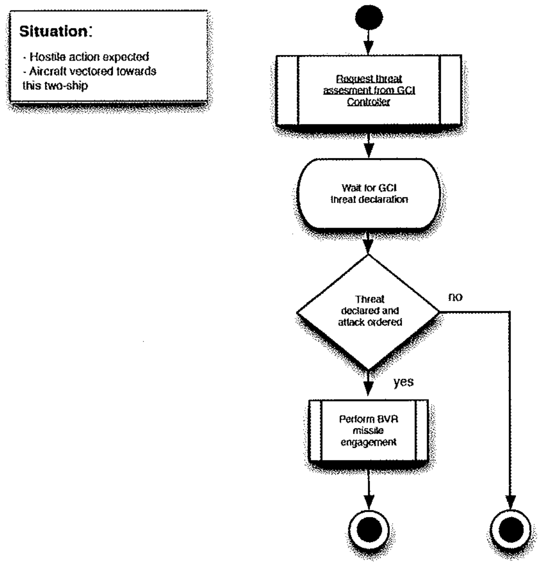
Situation:
- Hostile action expected
- Aircraft vectored towards this two-ship
Request threat assesment from GCI Controller
Wait for GCI threat declaration
Threat declared and attack ordered
no
yes
Perform BVR missile engagement

Fig. 1 A simple JACK/CoJACK plan.

nism affects the selection of the next intention to execute. Thus, the agent can choose the wrong intention or even fail to retrieve no intention. The adjustments mean that CoJACK also learns, which is somewhat unusual for an agent architecture.

The cognitive parameters can be moderated at runtime, leading to systematic variations in behaviour. For example, the caffeine moderator decreases the time taken to perform reasoning steps, leading to shorter response times.

CoJACK cognitive models have been created for three tasks: serial subtraction, how Rules of Engagement will influence and be influenced by behavioural moderators [Evertsz 07], and to play a very simple tank game [Evertsz 08].

3. Models of emotions

Cognitive science and cognitive modeling are increasingly interested in creating models that have emotions in them as a way to define and understand emotions. These models provide useful definitions of emotions and provide a way to explore what emotions are, how they may be related, and how they influence cognition and performance.

I would like to note two ways emotions have been created in cognitive architectures (out of several possible ways [Ritter 93]). They have been created by modifying architectural parameters (overlays) and by incorporating emotions directly into the architecture (drives).

3·1 Overlays

Representing emotion and other moderators of behavior as parameter changes is a very direct way to include their effects in a cognitive architecture. Architectures that include multiple parameters, making the changes easy to represent.

§1 Soar

There have been several attempts to include emotions in Soar (e.g., [Chong 98, Marinier 07]). The largest attempt is to include appraisal theory [Gratch 04], which is a nice example of how an emotional theory can be added to an architecture.

Gratch and Marsella found that additional types of information were required to create their models with emotions. They found that emotional content could be useful when generating interactions (e.g., answering the question 'what happened?' can be informed by the situation's emotional aspects). They found that the appraisal process, behavior, and the physiology of the agent interact. This interaction has been noted before, but their work shows that having a running model will make the work more interesting. Modeling this interaction remains a very important problem.

§2 ACT-R

ACT-R has been used numerous times to create overlays of emotions and behavioral moderators related to emotions. These include overlays representing the effects of development on cognition [Jones 00], fatigue [Jongman 98], stress [Belavkin 00], arousal [Cochran 06], sleep loss [Gunzelmann], and stress [Ritter 07].

Typically, they modify a few parameters within the existing architecture. In a few cases they modify the equations that the architecture uses to select a rule to fire or how declarative memories are retrieved.

This approach is incremental, but it provides a way for theories of emotions and behavioral moderators to be formally represented. The resulting theories will work with other models, for example, a driving model, to provide predictions of how driving is influenced by fatigue.

§3 CoJACK

CoJACK uses nearly all of the equations in ACT-R to modify the use of BDI constructs. Figure 2 shows how overlays interact with JACK and CoJACK. The overlays modify parameters to represent changes such as working memory capacity and access speed, processing speed, and noise in the decision process. These overlays include stress, the effects of caffeine (submitted), and fear [Evertsz 07].

§4 Summary

Overlays have been created in each of these architectures. These overlays have been used to implement a wide

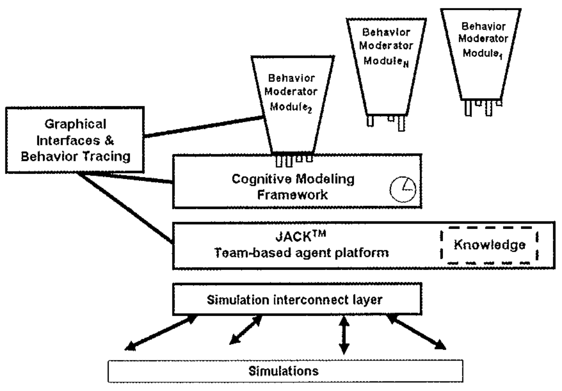
Behavior Moderator Module$_N$
Behavior Moderator Module$_1$
Behavior Moderator Module$_2$
Graphical Interfaces & Behavior Tracing
Cognitive Modeling Framework
JACK™ Team-based agent platform
Knowledge
Simulation interconnect layer
Simulations

Fig. 2 The relationship of JACK, CoJACK, and moderators.

range of theories and suggest that most emotions and moderators can be explored in this way.

These overlays can be broken into types similar to the way that architectural components [Gray 07] can be represented: Changes can be made to (local) parameters, changes can be made to architectural components, and changes can be made to strategies or knowledge representations.

These overlays have raised some interesting questions. For example, the work on modeling sleep deprivation in ACT-R [Gunzelmann] has raised questions about how to generalize results across tasks and across populations. Modeling caffeine has left us slightly stumped without additional data-yes, people with caffeine get faster, but how to apportion the speed-up across mechanisms?

Initial work has started with relatively simple and in most cases static overlays. In time, the overlays will have to represent a more dynamic system that is being modeled — emotions interaction with cognition, and behavior moderators influence strategy choices, which in turn influence further changes to cognition.

Similar work is possible in other architectures. It would be straight forward to implement similar overlays to EPIC [Kieras 97], Clarion [Sun 06b], R-CAST [Fan 07], and even GOMS [Kieras 98]!

3·2 PSI and other systems with drives

Dietrich Dörner proposed the PSI architecture (e.g., [Dörner 03]). This architecture includes drives that interact with most of its cognitive mechanisms. The drives help clarify how emotions could arise and change the way the architecture works on a fundamental level, providing an architecture more suited for behaving autonomously in a simulated world, which it does.

PSI includes three types of drives, physiological (e.g., hunger), social (i.e., affiliation needs), and cognitive (e.g., reduction of uncertainty). These drives influence all goal formation and knowledge selection and application. The resulting architecture generates new kinds of behaviors, including context dependent memories, socially motivated behavior, and internally motivated task switching. Joscha Bach has recently implemented this theory as the MicroPSI architecture [Bach].

While there have been tests of the architecture's predictions to human data, the resulting PSI models are generally not presented as validated cognitive models, but as theoretical explorations in the space of architectures for generating behavior. The sweep of the architecture can thus be larger—it presents a new cognitive architecture attempting to provide a unified theory of cognition.

§ 1 Summary and open problems

PSI and similar systems that include emotions directly, including PMServe [Silverman 04], Clarion [Sun 06a], and MAMID [Hudlicka 02], have several lessons for other architectures and models. They suggest that the mechanisms related to drives are important aspects of human behavior. For simple systems, not including them is not a problem. As models attempt to cover a wider range of behavior and over longer time periods, drives will become more important, and simple overlays will eventually not be able to cope with the breadth of effects.

In time, models will come to include the effects on cognition of emotions and behavioral moderators like lack of sleep that most experimental methods have attempted to remove. Architectures will need to include through overlays simple changes to information processing that arise from these factors. They will also have to include drives to represent long term goals and needs.

Including behavioral moderators are important for applications because we would like to predict actual not just ideal behavior. While many users are fresh and alert, not all are. It will also be important for simulations to have models that do not do the right thing, as these represent opponents to take advantage of and colleagues to support.

Including these effects are important for theoretical reasons. They will help codify theories and can help generate more accurate predictions.

In the end, this approach will require creating a simulated body to hold the physiology of the mechanisms, including fatigue, neurotransmitters, and other systems. As embodying cognition progresses it will offer a way to unify the life sciences. Before that happens, we will need summaries of how specific moderators influence cognition on multiple tasks. These summaries will take time, but be invaluable for creating overlays to architectures.

4. Usable models

Models of emotional effects need larger models to represent strategies to choose between and larger knowledge sets to implement long term behavior. It has been noted a few times that cognitive models are not always easy to create and use [Pew 98, Pew 07, Ritter 03b]. It is the case that they are not as easy to use as we would like them to be. Whenever theory cannot be applied because it is too difficult, it might be viewable as an engineering problem.

I would like to introduce the term cognitive science engineering. This is a new area for studying how to make and apply cognitive science. (This area cannot be called cognitive engineering, because that name is already taken.) Cognitive science tends to reject the engineering of models as cognitive science in the same way that psychology tends to reject software for psychology as psychology (although tools for physics and chemistry are physics and chemistry).

The usability of models and the ability to create and understand large models appear to be the first and perhaps the most important type of cognitive science engineering. Other examples of cognitive science engineering are now clearly visible in a previous review [Ritter 03b], such as optimizing the fit of models to data [Kase 08] and connecting models to simulations, where both engineering and science interact to create plausible, accurate connections between a model and a simulated or real world [Byrne 01, Ohno 99, Ritter 00]. These are not problems directly in the science of cognitive science, but they are problems that stop cognitive science from progressing in a particular direction.

The usability of models has been addressed by the three architectures differently, and provide lessons for each other and for other architectures.

4·1 Soar

There have been several attempts to provide programming interfaces for Soar. The Developmental Soar Interface (DSI) was one of the first [Ritter 94]. Since then, there have been several console based interfaces (e.g., [Ritter 98]). The current version of Soar (9) has a pretty nice interface on this level.

More recently, structured rule editors have been provided. These, such as Visual-Soar, are becoming more sophisticated and complex, and start to organize the rules as parts of operators, and provide a datamap of the structures used across operators. These tools help with relatively low-level programming, and are becoming more high-level as time progresses.

There are two more promising approaches that propose higher level languages that compile into Soar. The first high-level language for Soar was TAQL [Yost 93]. A small study suggested that it allowed users to write Soar models about three times faster. HLSR is a more recent attempt to create a high level language for Soar and ACT-R [Jones 06].

Herbal is another attempt [Haynes 09]. It implements a version of the Problem Space Computational Model [Newell 90] as a language that compiles into Soar and Jess. It has been used by over 100 people. Our current best estimate is that it allows undergraduate psychology majors to create (small) cognitive models more than 3 times faster than computer science graduate students.

4·2 ACT-R

ACT-R currently has a nice rule-level interface, but it does not have a general high-level language. However, there are several projects on how to make simplified ACT-R models more quickly [Ritter 06]. These include special purpose languages like ACT-Simple, which creates simple models; G2A, which creates ACT-R models from GOMS models; CogTool, a system to build simple models automatically from visual descriptions, and a language for menu interaction.

ACT-R has an increasing range of theories that can and are being reused across models. These do not provide a general language, but they help build models, and may provide lessons on how to get model reuse or may provide a type of module-based high-level language.

4·3 CoJACK

CoJACK was partially created based on a review that argued that it would be easier to create an interface and extension to JACK to make it cognitive than it would be to use Soar [Shakir 02]. This report had flaws, for example, a limited sample size of researchers were contacted. It was, however, encouraging that usability was considered in this decision. In the several years that I have worked with CoJACK, users who knew Java did not complain about CoJACK's usability.

There are probably several reasons that CoJACK appears or is usable (which might be the same thing). The high-level BDI constructs that it inherits are easy to use because they use folk psychology terms that programmers and experts themselves use. There are questions we are still exploring of whether these are the constructions that people use internally to generate behavior, but the translation from BDI constructs to cognitive science constructs, if they are different, might be supported later. Also, a comprehensive interface, broadly defined, was carefully designed and created. Finally, it could be argued that Java and the constructions are complex enough that only elite programmers have used it.

The next step for CoJACK is to use the interface to create larger, more complex models, and then, to validate them. That will be real proof that the architecture is usable.

4·4 Open problems

There is a need to create larger models, that is, with more knowledge in them, and this need will occur in every architecture. This is particularly true when modeling how emotions lead to strategy choices and changes in performance. All of these architectures are headed towards providing high-level languages. A recent review [Ritter 06] found that there were multiple groups working in this area. This work will be important for some time to come.

5. Conclusions

This brief review has described several hybrid cognitive architectures. Their extension with a sub-symbolic representation provides them with more accurate timing predictions (particularly including variability in performance), errors (because the symbolic processes become less accurate at applying the knowledge that they have), and the ability to include moderators where basic processes become less accurate.

This paper also introduced cognitive science engineering, the engineering of cognitive science systems. Creating the various aspects of the cognitive theories as software takes time and effort. Creating this label may provide a way to describe this type of work, help practitioners find each other, and provide a way to label commonly occurring problems and solutions.

It is worth pointing out that this work noted here does not require much technology for progress. Much of the work can be accomplished with a stopwatch, a way to make audio and perhaps video recordings, and any common computer. However, this work will use technology if it is available, for example, high-performance computing. Thus, it can be taken up in many research settings.

This review has noted several interesting problems in modeling emotions. These projects are examples of how computational modeling of cognition can be used to unify psychology and to make increasingly accurate predictions of human behavior. Similar projects are available applying cognitive models to other areas of human behavior. As such, they offer an excellent way to build collaborations across areas of psychology and the social sciences, and to support applications in virtual worlds and to support interface design.

These research problems are not unique to the architectures discussed—the same problems are general and will arise in other architectures. So, they will keep us busy for quite a while, and can be attacked by a range of researchers. The research agenda is modest in that we have only just begun to accomplish work in this area. It is not modest at all in that the research agenda is quite large.

Acknowledgments

Support was provided by ONR Contract N00014-03-1-0248 and N00014-06-1-0164, and by the UK MOD's Directorate of Analysis, Experimentation and Simulation corporate research programme (Project No: RT / COM / 3 / 006). Olivier Georgeon, Kazuhisa Miwa, and Colin Sheppard provided useful comments that improved this paper.

◇ References ◇

[Anderson 73] Anderson, J. R. and Bower, G. H.: *Human associative memory*, Erlbaum, Hillsdale (1973)

[Anderson 83] Anderson, J. R.: *The architecture of cognition*, Harvard University Press, Cambridge, MA (1983)

[Anderson 07] Anderson, J. R.: *How can the human mind exist in the physical universe?*, Oxford University Press, New York, NY (2007)

[Bach] Bach, J.: *Principles of synthetic intelligence: Building blocks for an architecture of motivated cognition*, Oxford University Press, New York, NY (in press)

[Belavkin 00] Belavkin, R. and Ritter, F. E.: Adding a theory of motivation to ACT-R, in *Proceedings of the Seventh Annual ACT-R Workshop*, pp. 133–139, Pittsburgh, PA (2000), Department of Psychology, CMU

[Busetta 99] Busetta, P., Rönnquist, R., Hodgson, A., and Lucas, A.: JACK intelligent agents - Components for intelligent agents in JAVA, *AgentLink News Letter, 2(Jan)* (1999), www.agent-software.com/white-paper.pdf

[Byrne 01] Byrne, M. D.: ACT-R/PM and menu selection: Applying a cognitive architecture to HCI, *International Journal of Human-Computer Studies*, Vol. 55, No. 1, pp. 41–84 (2001)

[Chong 98] Chong, R. S.: *Modeling dual-task performance improvement: Casting executive process knowledge acquisition as strategy refinement*, Unpublished PhD thesis. CSE-TR-378-98, The University of Michigan (1998)

[Chong 01] Chong, R. S.: Low-level behavioral modeling and the HLA: An EPIC-Soar model of an enroute air-traffic control task, in *Proceedings of the 10th Computer Generated Forces and Behavioral Representation Conference*, pp. 27–35, Orlando, FL (2001), Division of Continuing Education, University of Central Florida

[Cochran 06] Cochran, R. E., Lee, F. J., and Chown, E.: Modeling emotion: Arousal's impact on memory, in *Proceedings of the 28th Annual Conference of the Cognitive Science Society*, pp. 1133–1138, Mahwah, NJ (2006), Erlbaum

[Dörner 03] Dörner, D.: The mathematics of emotions, in *Proceedings of the Fifth International Conference on Cognitive Modelling*, pp. 75–80, Bamberg, Germany (2003), Universitäts-Verlag Bamberg

[Evertsz 07] Evertsz, R., Ritter, F. E., Russell, S., and Shepherdson, D.: Modeling rules of engagement in computer generated forces, in *Proceedings of the 16th Conference on Behavior Representation in Modeling and Simulation*, 07-BRIMS-021, Norfolk, VA (2007), U. of Central Florida

[Evertsz 08] Evertsz, R., Busetta, P., Pedrotti, M., Ritter, F. E., and Bittner, J. L.: CoJACK-Achieving principled behaviour variation in a moderated cognitive architecture, in *Proceedings of the 17th Conference on Behavior Representation in Modeling and Simulation*, 08-BRIMS-025, pp. 80–89, Orlando, FL (2008), U. of Central Florida

[Fan 07] Fan, X. and Yen, J.: R-CAST: Integrating team intelligence for human-centered teamwork, in *Proc. of the Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*, pp. 1535–1541 (2007)

[Gratch 04] Gratch, J. and Marsella, S.: A domain-independent framework for modeling emotion, *Journal of Cognitive Systems Research*, Vol. 5, No. 4, pp. 269–306 (2004)

[Gray 07] Gray, W. D.: Composition and control of integrated cognitive systems, in Gray, W. D. ed., *Integrated models of cognitive systems*, pp. 3–12, Oxford University Press, New York, NY (2007)

[Gunzelmann] Gunzelmann, G., Gross, J. B., Gluck, K. A., and Dinges, D. F.: Sleep deprivation and sustained attention performance: Integrating mathematical and cognitive modeling, *Cognitive Science* (in press)

[Haynes 09] Haynes, S. R., Cohen, M. A., and Ritter, F. E.: A design for explaining intelligent agents, *International Journal of Human-Computer Studies*, Vol. 67, No. 1, pp. 99–110 (2009)

[Hudlicka 02] Hudlicka, E. and Pfautz, J.: Architecture and representation requirements for modeling effects of behavior moderators, in *Proceedings of the Eleventh Conference on Computer Generated Forces and Behavioral Representation*, 02-CGF-085, pp. 9–20, Orlando, FL (2002), Division of Continuing Education, University of Central Florida

[Johnson 98] Johnson, T. R.: A comparison of ACT-R and Soar, in Schmid, U., Krems, J., and Wysotzki, F. eds., *Mind modeling - A cognitive science approach to reasoning, learning and discovery*, pp. 17–38, Pabst Scientific Publishing, Lengerich, Germany (1998)

[Jones 00] Jones, G., Ritter, F. E., and Wood, D. J.: Using a cognitive architecture to examine what develops, *Psychological Science*, Vol. 11, No. 2, pp. 93–100 (2000)

[Jones 06] Jones, R. M., Crossman, J. A. L., Lebiere, C., and Best, B. J.: An abstract language for cognitive modeling, in *Proceedings of the 7th International Conference on Cognitive Modeling*, pp. 160–165, Mahwah, NJ (2006), Erlbaum

[Jongman 98] Jongman, G. M. G.: How to fatigue ACT-R?, in *Proceedings of the Second European Conference on Cognitive Modelling*, pp. 52–57, Nottingham (1998), Nottingham University Press

[Kandel 92] Kandel, A. and Langholz, G.: *Hybrid architectures for intelligent systems*, CRC Press, Boca Raton, FL (1992)

[Kase 08] Kase, S. E.: *HPC and PGA optimization of a cognitive model: Investigating performance on a math stressor task*, Unpublished PhD thesis, College of IST, Penn State University, University Park, PA (2008)

[Kennedy 06] Kennedy, W. G. and Trafton, J. G.: Long-term learning in Soar and ACT-R, in *Proceedings of the Seventh International Conference on Cognitive Modeling*, pp. 162–168, Trieste, Italy (2006), Edizioni Goliardiche

[Kieras 97] Kieras, D. E., Wood, S. D., and Meyer, D. E.: Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task, *Transactions on Computer-Human Interaction*, Vol. 4, No. 3, pp. 230–275 (1997)

[Kieras 98] Kieras, D. E.: A guide to GOMS model usability evaluation using NGOMSL, in Helander, M. ed., *Handbook of human-computer interaction*, Vol. 12, pp. 391–438, Elsevier, Amsterdam (1998)

[Laird 08] Laird, J. E.: Extending the Soar cognitive architecture, in *Artificial General Intelligence Conference*, Memphis, TN (2008)

[Lebiere 01] Lebiere, C.: Multi-tasking and cognitive workload in an ACT-R model of a simplified air traffic control task, in *Proceedings of the 10th Computer Generated Forces and Behavioral Representation Conference*, pp. 91–98, Orlando, FL (2001), Division of Continuing Education, University of Central Florida

[Marinier 07] Marinier, R. P. and Laird, J. E.: Proceedings of Computational modeling of mood and feeling from emotion, in *Proceedings of the 29th Annual Conference of the Cognitive Science Society*, Austin, TX (2007), Cognitive Science Society

[Newell 90] Newell, A.: *Unified Theories of Cognition*, Harvard University Press, Cambridge, MA (1990)

[Norling 04] Norling, E. and Ritter, F. E.: *COJACK Software Specification*, Agent Oriented Software Limited (2004)

[Ohno 99] Ohno, T. and Ogasawara, H.: Information acquisition model of highly interactive tasks, in *Proceedings of ICCS/JCSS-99*, pp. 288–293, Waseda University (1999), Japanese Cognitive Science Society

[Pew 98] Pew, R. W. and Mavor, A. S. eds.: *Modeling human and organizational behavior: Application to military simulations*, National Academy Press, Washington, DC (1998), books.nap.edu/catalog/6173.html

[Pew 07] Pew, R. W. and Mavor, A. S. eds.: *Human-system integration in the system development process: A new look*, National Academy Press, Washington, DC (2007), books.nap.edu/catalog.php?record_id=11893

[Ritter 93] Ritter, F. E.: Three types of emotional effects that will occur in cognitive architectures, in *Workshop on architectures underlying motivation and emotion (WAUME93)*, School of Computer Science and Centre for Research in Cognitive Science, University of Birmingham, UK (1993)

[Ritter 94] Ritter, F. E. and Larkin, J. H.: Developing process models as summaries of HCI action sequences, *Human-Computer Interaction*, Vol. 9, No. 3, pp. 345–383 (1994)

[Ritter 98] Ritter, F. E., Jones, R. M., and Baxter, G. D.: Reusable models and graphical interfaces: Realising the potential of a unified theory of cognition, in Schmid, U., Krems, J., and Wysotzki, F. eds., *Mind modeling-A cognitive science approach to reasoning, learning and discovery*, pp. 83–109, Pabst Scientific, Lengerich, Germany (1998)

[Ritter 00] Ritter, F. E., Baxter, G. D., Jones, G., and Young, R. M.: Supporting cognitive models as users, *ACM Transactions on Computer-Human Interaction*, Vol. 7, No. 2, pp. 141–173 (2000)

[Ritter 03a] Ritter, F. E.: Soar, in Nadel, L. ed., *Encyclopedia of cognitive science*, Vol. 4, pp. 60–65, Nature Publishing Group, London (2003)

[Ritter 03b] Ritter, F. E., Shadbolt, N. R., Elliman, D., Young, R. M., Gobet, F., and Baxter, G. D.: *Techniques for modeling human performance in synthetic environments: A supplementary review*, Human Systems Information Analysis Center (HSIAC), Wright-Patterson Air Force Base, OH (2003)

[Ritter 06] Ritter, F. E., Haynes, S. R., Cohen, M., Howes, A., John, B., Best, B., et al.: High-level behavior representation languages revisited, in *ICCM - 2006- Seventh International Conference on Cognitive Modeling*, pp. 404–407, Trieste, Italy (2006), Edizioni Goliardiche

[Ritter 07] Ritter, F. E., Reifers, A. L., Schoelles, M., and Klein, L. C.: Lessons from defining theories of stress for architectures, in Gray, W. ed., *Integrated models of cognitive systems*, pp. 254–262, Oxford University Press, New York, NY (2007)

[Salvucci 06] Salvucci, D. D.: Modeling driver behavior in a cognitive architecture, *Human Factors*, Vol. 48, pp. 362–380 (2006)

[Shakir 02] Shakir, A.: Assessment of models of human decision-making for air combat analysis, Unpublished technical report (2002), Abstract, put on the web with permission, at http://acs.ist.psu.edu/papers/shakir02-abstract.pdf

[Silverman 04] Silverman, B. G.: Human performance simulation, in J. W. Ness, D. R. R. and Tepe, V. eds., *The science and simulation of human performance*, pp. 469–498, Elsevier, Amsterdam (2004)

[Sun 94] Sun, R. and Bookman, L. A. eds.: *Computational architectures integrating neural and symbolic processes: A perspective on the state of the art*, Kluwer, Norwell, MA (1994)

[Sun 06a] Sun, R.: The Clarion cognitive architecture: Extending cognitive modeling to social simulation, in Sun, R. ed., *Cognition and multi-agent interaction*, Cambridge University Press, New York (2006)

[Sun 06b] Sun, R. and Zhang, X.: Accounting for a variety of reasoning data within a cognitive architecture, *Journal of Experimental and Theoretical Artificial Intelligence*, Vol. 18, No. 2, pp. 169–191 (2006)

[Taatgen 03] Taatgen, N. A. and Lee, F. J.: Production compilation: A

simple mechanism to model complex skill acquisition, *Human Factors*, Vol. 45, No. 1, pp. 61–76 (2003)
[Touretzky 86] Touretzky, D. S.: A distributed connectionist production system, (technical report No. CMU-CS-86-172), Computer Science Department, CMU (1986)
[Yost 93] Yost, G. R.: Acquiring knowledge in Soar, *IEEE Expert*, Vol. 8, No. 3, pp. 26–34 (1993)

〔担当委員：三輪 和久〕

Received October 8, 2008.

Author's Profile

Ritter, Frank E.

Frank Ritter researches the development, application, and methodology of cognitive models, particularly as applied to understanding learning, predicting the effect of behavioral moderators, and interface design. He currently edits the Oxford series on cognitive models and architectures for Oxford University Press.