A Preliminary ACT-R Compiler in Herbal

Jaehyon Paik (jaehyon.paik@psu.edu)
Department of Industrial and Manufacturing Engineering
The Pennsylvania State University, University Park, PA 16802 USA

Jong W. Kim (jongkim@psu.edu)
Frank E. Ritter (frank.ritter@psu.edu)
College of Information Science and Technology
The Pennsylvania State University, University Park, PA 16802 USA

Herbal (Haynes, Cohen, & Ritter, 2009) represents human behavior based on the Problem Space Computational Model (Newell, Yost, Laird, Rosenbloom, & Altmann, 1991). The PSCM is a theory of cognition that defines human behavior as movement through problem spaces using operators. To represent models, Herbal uses XML as basis forms of high-level language, and Herbal compiles them into low-level rule-based representations that execute within a cognitive architecture, Soar, and intelligent agent architecture, Jess. Users can create them with a GUI or directly in XML.

Herbal is implemented as an Eclipse plug-in, which provides a popular graphical development environment. It uses Eclipse's powerful functions for creating and maintaining agents, so it enables model creators to make models more easily.

We have started to create an ACT-R compiler in Herbal because of these features. Although several easy to use frameworks exists to develop ACT-R models, such as ACT-Simple (Salvucci & Lee, 2003), and G2A (St. Amant & Ritter, 2004), they cannot represent more than KLM-GOMS or GOMS models.

Matching the Herbal Components with ACT-R Components

We developed the ACT-R compiler based on the Jess compiler, because the Jess compiler compiles into declarative knowledge and procedural knowledge, and its output has a Lisp-like syntax similar to ACT-R. The current version of Herbal has 6 basic components, Agent, Problem Space, Operator, Condition, Action, and Type. These can all be mapped onto ACT-R components as shown in Table 1, which includes their Jess correspondences as well.

Table 1: Herbal components and their implementation in Jess and ACT-R.

Herbal	Jess	ACT-R
Agent	Agent	Model
Problem Space	Defmodule	Slot of Goal buffer
Operator	Defrule	Production
Condition	Condition of defrule	Condition of rule
Action	Action of defrule	Action of rule
Type	Deftemplete	Chunk-type
- Field	- Slot	- Slot

We added an additional component, called *Declarative Memory,* in the Herbal environment. With this component, users can represent hierarchical or sequential tasks in an ACT-R model easily. The *Declarative Memory* consists of 6 components: library, element name, parent name, first child name, next sibling name, and action name. Through these components, users can layout their whole task hierarchically or sequentially, and the relations among tasks are shown in a tree form in the bottom of the user interface. Based on these relationships, the productions are made by ACT-R compiler.

To explore the flexibility of this high-level compiler approach, we added a user expertise compiler flag to Herbal. It leads to compiling either a novice or an expert user model. The expert model does not retrieve declarative memory items when it executes subtasks. However, the novice model retrieves declarative memory items to move to the next task step according to the goal hierarchy in declarative memory. Figure 1 shows the difference between Expert and Novice model.

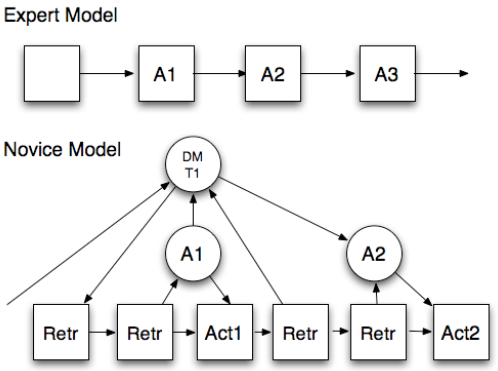

Figure 1: The Structure of the Expert and Novice model.

Experiment

We use a simple dialing number task to show a simple model. This task is decomposed into a set of hierarchical subtasks to dial each component and then the numbers in each component. It consists of four subtasks: Long Distance Code, Area Code, Exchange, and Extension. Each subtask has its own subtasks (the buttons to press), and all of these subtasks are related with other tasks and subtasks as a parent, child, or sibling.

We chose to dial 1 (814) 865–4455, so the Long Distance has a subtask, press-1, and the Area Code has subtasks, press-8, press-1, and press-4. The Exchange and Extension have similar subtasks. So, the total number of tasks is 11 leaf nodes, 4 sub nodes, and one task node, and these tasks are stored as 16 declarative memory type nodes in the Herbal environment.

Using the ACT-R compiler with the user expertise compiler flags, we generated expert and novice models. The expert model produces the next task without retrieving a declarative memory using 17 rules and 16 chunks, however, the novice model retrieves a declarative memory to produce the next task using 8 rules and 16 chunks.

We simulated 10 trials per model to get mean prediction times. The default ACT-R parameters were used (these are carried in the compiler). The expert model's predicted times, shown in Table 2, did not have any variance with the ACT-R default values, however, there exist differences among the trials in the novice model. In addition to the ACT-R cognitive modules' times, we added keystroke times (typing random letter) to each model to get total predicted time (we do not yet use ACT-R/PM for motor output).

For comparison, this task was analyzed using the KLM-GOMS theory (Card, Moran, & Newell, 1983). For the keystroke operator, we use the same time of "typing random letter", 0.50 s. The number of keystrokes is 11. Thus, the total time spent in key stroking is 5.5 s (as used above). For each mental operator, we use the default preparation time (T_M) of 1.35 s. A user mentally prepares what numbers to press, what to retrieve from memory, and what to do for the next step. In this task, a mental preparation for each subtask was counted: Long Distance Code, Area Code, Exchange, and Extension. Thus, the total time spent in mental preparation is 5.4 s. Therefore, the total execution time from the KLM is 10.9 s ($T_{execute} = T_K + T_M = 5.5 + 5.4$). Table 2 shows above result with respect to prediction time, and the number of rule firings.

Table 2: The mean, standard deviations of prediction time, and the number of rule firings in each model, and KLM model for the simple dialing number task (N=10).

	KLM	Novice	Expert
Mean	10.9 s	13.48 s	6.35 s
SD	0 s	0.79 s	0 s
Rule firings	-	20	16

The Herbal/ACT-R novice model is a bit slow compared to the KLM predictions, as it should be. The Herbal/ACT-R expert model is a bit fast. It is the case, that the Herbal ACT-R compiler makes different predictions across the expert and novice models, and it may be the case that subjects when they perform this task are best represented by a model between these two extremes, or by a distribution of user models, as John (1996) proposed .

Discussion and Conclusion

We started to develop an ACT-R compiler and declarative memory component in the Herbal environment. This compiler takes knowledge represented as a PSCM model in Herbal, and in addition to compiling it in Soar and in Jess, compiles it into ACT-R. This compilation process was tested and appears to show some promise for creating more sophisticated models more easily.

We added a declarative memory pane for representing hierarchical task analyses. This representation is not currently pretty, but allows users to represent tasks in a GOMS-like language. As part of this component, we included a way (a compiler flag) to generate both novice and expert models from the same knowledge set. The novice model accesses the declarative memory elements to generate behavior. The expert model is compiled so that the rules apply directly and keep the state on the goal. (This compiler flag is not yet used by the Soar or Jess compilers.)

The model of simple dialing number task was compared with a GOMS model with respect to predicted time. The GOMS model's prediction time is located between our expert and novice model. The novice model of this task fired 20 rules, and the expert model fired 16. If a task has more hierarchical levels in it, the number of rule firings between these two model types will be more different. Because this task has a hierarchical structure (3 levels), there was a noticeable difference.

References

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum.

Haynes, S. R., Cohen, M. A., & Ritter, F. E. (2009). A design for explaining intelligent agents. *International Journal of Human-Computer Studies, 67*(1), 99-110.

John, B. E. (1996). TYPIST: A theory of performance in skilled typing. *Human-Computer Interaction, 11*(4), 321-355.

Newell, A., Yost, G. R., Laird, J. E., Rosenbloom, P. S., & Altmann, E. (1991). Formulating the problem space computational model. In R. F. Rashid (Ed.), *Carnegie Mellon Computer Science: A 25-Year commemorative* (pp. 255-293). Reading, MA: ACM-Press (Addison-Wesley).

Salvucci, D. D., & Lee, F. J. (2003). Simple cognitive modeling in a complex cognitive architecture. In *Human Factors in Computing Systems: CHI 2003 Conference Proceedings*, 265-272. ACM: New York, NY.

St. Amant, R., & Ritter, F. E. (2004). Automated GOMS to ACT-R model generation. In *Proceedings of the International Conference on Cognitive Modeling*, 26-31. Erlbaum: Mahwah, NJ.