AI Support for Building Cognitive Models

Robert St. Amant and Sean P. McBride
Department of Computer Science
North Carolina State University
Raleigh, NC 27695

Frank E. Ritter
College of Information Sciences and Technology,
The Pennsylvania State University
University Park, PA 16802

Abstract

Cognitive modeling techniques provide a way of evaluating user interface designs, based on what is known about human cognitive strengths and limitations. Cognitive modelers face a tradeoff, however: more detailed models require disproportionately more time and effort to develop than coarser models. In this paper we describe a system, G2A, that automatically produces translations from abstract GOMS models into more detailed ACT-R models. G2A demonstrates how even simple AI techniques can facilitate the construction of cognitive models and suggests new directions for improving modeling tools.

Introduction

In cognitive modeling, the concept of cognitive architectures has come to play a central role. A cognitive architecture is a theory of human cognition that can be realized as a program. Cognitive architectures commonly integrate theories of cognition, visual attention, and motor movement into a consistent framework. Models built on top of a given architecture provide a way of applying what is known about psychology to both explain and reproduce human behavior in specific tasks under the fixed constraints of the architecture. Models apply specific problem-solving strategies to tasks, reproducing aspects of human behavior such as durations for high- and low-level actions, occurrences of errors, and abstract strategies for exploration and learning.

There are several ways AI can be applied to cognitive modeling. Building cognitive models is a knowledge-intensive process that AI techniques might help manage. Cognitive models need to represent knowledge, and AI techniques can help with this representation. Lastly, model development and data fitting can involve search, which AI algorithms can facilitate.

Over the past few decades, useful collaborations have been established between researchers in cognitive modeling and human-computer interaction (HCI). Models can give insight into the behavior of users of interactive systems; interactive systems provide realistic, challenging tasks in which models can be validated. In some cases, a general cognitive architecture such as ACT-R [Anderson and Lebiere, 1998] has been the focus of collaboration, with extensions to the architecture being tailored to interaction with HCI environments. Other research has relied on HCI-specific architectures, such as the Model Human Processor [Card et al., 1980].

GOMS modeling for HCI evaluation [John and Kieras, 1996] is based on the latter architecture. GOMS (Goals, Operators, Methods, and Selection Rules) can be thought of as a high-level language in which interaction tasks can be expressed in a hierarchical form that reflects a decomposition of complex tasks into simpler ones.[1] GOMS operators include, for example, storage and retrieval of task items, goal establishment, and decision points. GOMS is a well-known formalism that has been applied with significant success in critical HCI domains [Gray et al., 1993].

An example of a general cognitive architecture that has been used in HCI research is ACT-R [Anderson and Lebiere, 1998]. The ACT-R architecture simulates internal cognitive processing, such as changes of attention and memory retrievals, as well as external behavior related to visual, auditory, and motor processing. We can think of ACT-R models as explaining behavior in cognitive terms at a more detailed level than GOMS models.

Because GOMS and ACT-R represent cognitive behavior at different levels of abstraction, a practical tradeoff arises: detailed models are much more difficult and time-consuming to build and test than coarser models. For very simple HCI tasks, such as traversing menus on a cell phone [St. Amant et al., 2004], a GOMS model might take hours to build, while an ACT-R model could take days or weeks to build. The ACT-R model, in compensation, can provide more insight into the cognitive processes operating during the task, including their timing, their learning, and their errors in execution.

In a paper presented at the 2004 International Conference on Cognitive Modeling [St. Amant & Ritter, 2004],[2] we asked the question, "To what extent can the translation between GOMS and ACT-R models be automated?" That is, given a coarse GOMS-level representation of human behavior, can we automatically build an ACT-R model that provides plausible additional detail? We presented a system called G2A with just such a capability.

[1] GOMS is actually a family of techniques; our description is specific to GOMSL [Kieras, 1999], a representative approach.

[2] This paper won the Best Applied Paper Prize at ICCM in 2004; at ICCM the work was viewed as an example of applied cognitive modeling, but it is also a plausible example of applied AI.

In the remainder of this paper we describe the representation of GOMS and ACT-R models in more detail, to provide context for our work. We then discuss the system G2A and its performance. We end with an account of the relevance of this work to the AI community, including opportunities for future work.

Cognitive Modeling and G2A

We can only give here a very approximate description of GOMS and ACT-R; in particular, most of the details of model execution are neglected.

GOMS models are usually expressed in procedural form. Figure 1 shows a sample GOMS method for editing a document [Kieras, 1999]. The method breaks down into a sequence of steps. Some steps are primitive operators such as storing an item (First, in this case, is just a symbol) in a named memory location, <current task name>; looking at the screen, pressing keys, and moving the mouse are also possibilities. Steps can also involve execution control, such as branching (in Decide forms) and method invocation (in Accomplish goal forms). The duration of operators is general across all GOMS models; for example, key presses take 280 milliseconds. As a model is executed, durations of operators and tasks are accumulated.

ACT-R takes a production system approach to modeling, as is obvious from the syntax of ACT-R models. Figure 2 shows a sample production, the equivalent of the Get-task-item-whose step in the GOMS method in Figure 1. This production describes operations on two buffers, goal and match. Each of these buffers has a set of slots whose values can be constrained in a standard way; slot names prefixed by the '=' sign are variables. On the left hand side of the production, the goal buffer is constrained to hold a goal with a given task name and state, and a task item matching the name must be found in memory. On the right hand side, the current task and state slots of the goal buffer are updated. As with GOMS models, the durations of low-level actions are specified by the ACT-R architecture rather than being specialized to a given model; durations are collected as productions fire.

In translating between these representations, there are three main issues to be addressed. First, we expect some correspondence between low-level operations in both representations; hand movements and button presses, for example, are based on similar models of motor behavior, and basic memory retrievals should be comparable across the representations. These correspondences must be mapped out explicitly. Second, the procedural structure in GOMS model execution must be reproduced in production form, managed by buffers that maintain appropriate control information. Third, and most important, some GOMS operations underconstrain equivalent behavior in ACT-R. For example, ACT-R represents visual processing at a level that allows different visual search strategies to be carried out through the firing of several productions, while in GOMSL such processing is encapsulated in a single primitive. Different possible translations must be identified and evaluated.

As discussed in the next section, G2A translates between the representation of GOMS and ACT-R models via translation rules, with results produced by a simple hill-climbing search. We evaluated G2A in two ways. First, by comparing the execution times of translated ACT-R models with their source GOMS models, we can show that existing predictive accuracy of a GOMS model is not lost in the translation. Second, by comparing the resulting ACT-R models to actual user performance, we can show that the translation gives information that is usable in practice. For both parts of the evaluation, we used a text editing task represented by the most complex model given in Kieras's GOMS manual (Kieras 1999). This model takes up about four pages—190 lines of code containing 15 mental object definitions, 11 methods, and two sets of selection rules.

Our evaluation resulted in a few promising findings:

- We used G2A to generate an ACT-R model that matched the duration of the methods in the original GOMS model with less than 5% error (less than 1% for the duration of the entire task). The ACT-R model contained five chunk definitions, 23 chunks, 79 productions, and various auxiliary constructs (over 1,500 lines of formatted model code).

```
Method for goal: Edit Document
Step. Store First under <current task name>.
Step Check for done.
Decide: If <current task name> is None, Then
   Delete <current task>;
   Delete <current task name>;
   Return with goal accomplished.
Step. Get task item whose
   Name is <current task name>
   and store under <current task>.
Step. Accomplish goal: Perform Unit task.
Step. Store Next of <current task>
   under <current task name>;
Goto Check for done.
```

Figure 1. Sample GOMS method

```
;;; Get-task-item-whose ((is name [current-task-name]))
;;; [current-task]
  (p production86
    =goal>
      isa goal
      [current-task-name] =[current-task-name]
      %state edit-document-3
    =match88>
      isa task-item
      %id =temp87
      name =[current-task-name]
    ==>
    =goal>
      [current-task] =temp87
      %state edit-document-4)
```

Figure 2. Sample ACT-R production

- We used the performance of a single user in a pilot study to steer G2A's translation process, in order to predict the performance of six other users. We found that the translated model (of comparable size to the translation above, but of different structure) generated by G2A matched user performance at the level of primitive operators with an error rate of around 35%. This accuracy is about average for automatically generated models in such tasks (John et al, 2004; Salvucci and Lee, 2003).

G2A as an AI Application

Much of the processing in G2A is based on simple techniques for interpreting and compiling programming languages. Nevertheless, as mentioned in the previous section, a GOMS model is at best an incomplete specification of an ACT-R model. The high-level abstractions provided by a GOMS model can be accomplished by different lower-level productions or different sets of productions in ACT-R. In other words, for any given GOMS model, there may be a large number of possible translations into ACT-R models. To find the "best" ACT-R model, we face a search problem.

G2A maintains a set of rules for the translation process. Simple GOMS operators may have a single translation rule (e.g., storing a value in a memory location can be handled by a single GOMS primitive and a single production). Other GOMS operators, however, may hide a good deal of complexity: a Look-for-object-whose operator could translate into a fixed sequence of productions that focus attention on a specific location in the visual environment, or a larger set of productions that govern an opportunistic visual search of objects in view, among other possibilities.

In G2A, the translation of a given GOMS model is treated as a search problem. To guide the search process, a target set of durations is required either for tasks or primitive operators. This target can be provided by the standardized durations for GOMS operators. Alternatively, when pilot user data is available, durations can be taken from actual user performance. A state in the search space is a set of translation rules, one for each GOMS operator. Evaluating a state involves applying the state's translation rules to the given GOMS model in order to generate an ACT-R model. The durations of ACT-R productions that correspond to primitive GOMS operators or task boundaries, as generated by execution of the ACT-R model, are compared with the target durations. G2A performs hill-climbing through this space, with state transitions corresponding to single substitutions of individual translation rules. Mean squared error over the duration comparisons is minimized. In general, a few hundred iterations are sufficient to identify good translations.

G2A brings some benefits to the cognitive modeling process. The ACT-R models automatically produced by G2A reflect straightforward decisions that a human modeler might make to capture human performance in this domain. Returning to our example of visual processing above, if an ACT-R modeler found that observable actions such as key presses were of very short duration, but that such actions depended on information on the screen, he or she would conclude that an exhaustive visual search of the screen is not being carried out and devise appropriately efficient ACT-R productions for the activity. The predictions of the translated ACT-R model, as generated by the search, can be much more accurate for predicting actual behavior than the standardized predictions of the source GOMS model. Generating the same ACT-R models by hand would be both time-consuming and error-prone, even in such a well-understood domain; in one case study [St. Amant et al., 2004] we found a difference of several hours versus two weeks for the construction of a GOMS model and an ACT-R model, respectively.

Discussion

While G2A represents perhaps the simplest possible application of a well-known AI technique to the problem of cognitive model translation, it is nevertheless interesting from an AI perspective: an important contribution of our work was to identify and formalize this problem in the first place. One view of the elaboration of cognitive models to greater levels of detail was that the process would be largely deterministic, with little freedom in mapping operations from one representation into another. Our experience with G2A showed that the differences between models at different levels of abstraction were greater than expected.

Along with Howes et al. [2005] and others [Ritter et al., 2006], our work shows something that is relatively obvious but that had not been widely acknowledged earlier: moving from an abstract model to a more detailed model of human performance, even if both models are based on a largely consistent set of assumptions about cognition, involves traversing a search space of significant size, and empirical results only weakly constrain the search. As AI researchers, we find that this area allows us to explore interesting and important questions for cognitive modeling, including what makes a good theory, how to represent and compute differences between strategies, and how to represent the complexities of human behavior. In future work, we would like to answer the following questions in more general terms:

To what extent can cognitive model development be supported by automated search? We have found that the model translation process can be managed by an automated search process, with many solutions covered by better solutions. Still, there are different "best" answers. To paraphrase Hamming [1962], the purpose of modeling is insight, not performance measurements. G2A allows cognitive modelers to move through a larger space of modeling decisions than is feasible without assistance, but it remains an open question exactly how much G2A contributes to the understanding of behavior.

Do general heuristics apply to the transformation between modeling architectures? In GOMS, visual and cognitive operators have the longest durations. This means that when encoding these operators as ACT-R productions their translation has the greatest potential for differing predictions. There are many choice points when going from GOMS specifications to ACT-R implications, and yet they are modeling the same mechanisms and behavior. The question of which theoretical level is most appropriate is likely to be a cause of difficulties.

To what extent can AI representations support cognitive modeling? Roughly speaking, a cognitive architecture can be characterized by its modeling language and its behavior in processing models expressed in that language. What this means in practice is that most GOMS modelers can ignore the internal representation of cognitive processing that is used by the GOMS architecture, focusing instead on models written in the external language, whereas ACT-R users more often need to keep the mechanisms in mind (this observation is reminiscent of the resolved debate about the value of higher-level languages versus assembly language).

In order to build G2A, we developed an ad hoc internal, intermediate representation that captures some of the commonalities between the architectures. In our current work we are revising the system to rely instead on PDDL [http://zeus.ing.unibs.it/ipc-5/]. We expect that a formal representation of cognitive modeling operations in planning terms will facilitate formal analysis of cognitive models to answer questions about, for example, the size of the production rule space traversed in principle by a cognitive architecture during the execution of a given model, and the changes to this space that arise from changes to the model.

Can AI techniques improve tools for cognitive modeling? Current environments for developing cognitive models are comparable to software development environments, supporting a tight loop of development, execution, and testing. They tend to work at a lower level of programming than modelers find desirable, and at a lower and slower level than most AI programmers would tolerate. This may be inherent in the task of modeling detailed human behavior, but the modeling process could plausibly be improved if AI tools were to migrate to modeling environments. Given an internal PDDL representation of a model, for example, a modeler could ask whether a given set of primitive operators could be combined so as to reach a specific goal, something that ACT-R modelers must currently evaluate without automated assistance.

In general, we believe that there are interesting opportunities for the application of AI concepts to the field of cognitive modeling, in particular to aid cognitive modelers in their development. Given the shared historical background and interests of the cognitive modeling and AI communities, this appears to be a natural area for collaboration.

Acknowledgements

This research was supported by the National Science Foundation (IIS-0083281 and ITR-046852) and the Office of Naval Research (contract N00014-06-1-0164).

References

Anderson, J. R., & Lebiere, C. 1998. *The Atomic Components of Thought.* Lawrence Erlbaum. Mahwah, NJ.

Card, S., Moran, T., & Newell, A. 1983. *The psychology of human-computer interaction*. Hillsdale, NJ: LEA.

Gobet, F., & Baxter, G. D. 2003. *Techniques for modeling human performance in synthetic environments: A supplementary review*. Wright Patterson AFB: Human Systems Information Analysis Center.

Gray, W. D., John, B. E., & Atwood, M. E. 1993. Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction, 8*(3), 237-309.

Hamming, R. W. 1962. *Numerical methods for scientists and engineers*. Dover, New York.

Howes, A., Lewis, R. L., Vera, A., & Richardson, J. (2005). Information-Requirements Grammar: A theory of the structure of competence for interaction. In *Proceedings of the 27th Annual Meeting of the Cognitive Science Society*, 977-983. Hillsdale, NJ: Lawrence Erlbaum.

John, B. E., & Kieras, D. E. 1996. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction, 3*(4), 320-351.

John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. 2004. Predictive human performance modeling made easy. *Proceedings of Chi '04*, 455–462. ACM.

Kieras, D. 1999. A Guide to GOMS Model Usability Evaluation using GOMSL and GLEAN3. University of Michigan, Ann Arbor, MI.

Ritter, F. E., Haynes, S. R., Cohen, M. Howes, A., John, B., Best, B., Lebiere, C., Jones, R. M., Crossman, J., Lewis, R. L., St. Amant, R., McBride, S. P., Urbas, L., Leuchter, S., Vera, A. (in press). High-level Behavior Representation Languages Revisited. In *Proceedings of ICCM - 2006- Seventh International Conference on Cognitive Modeling.*

Salvucci, D. D. and Lee, F. J. 2003. Simple Cognitive modeling in a complex cognitive architecture. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 265-272.

St. Amant, R., Freed A., and Ritter, F. E. (2004). Specifying ACT-R models of user interaction with a GOMS language. *Cognitive Systems Research. 6*(1): 71-88.

St. Amant, R., Horton, T. E., and Ritter, F. E. (2004). Model-based evaluation of cell phone menu interaction. In *Proceedings of CHI'04*. 343-350. NY, NY: ACM.

St. Amant, R. and Ritter, F. E. 2004. Automated GOMS to ACT-R model translation. *Proceedings of the Sixth International Conference on Cognitive Modeling,* 26-31. Mahwah, NJ: LEA.