Cogn Process (2005) 6: 142–146
DOI 10.1007/s10339-005-0052-4

LABORATORY NOTES

Lucio Inguscio · Frank E. Ritter

Applied Cognitive Science Laboratory at the Pennsylvania State University

Received: 13 December 2004 / Revised: 7 February 2005 / Accepted: 7 March 2005 / Published online: 10 May 2005

Introduction

The Applied Cognitive Science Laboratory of the School of Information Sciences and Technology at the Pennsylvania State University is situated in the town of State College, PA, just about halfway between Pittsburgh (Carnegie-Mellon University) and Philadelphia. The head of laboratory is prof. Frank Ritter, who is interested in cognitive science and in its applications to human–computer interaction. The focus of research in this lab is to understand human cognition by modeling its output and process, creating models that can explain its behavior, testing human–computer interfaces, and serving as colleagues and opponents in simulations. The projects are focused on models that learn, ranging from how to provide them access to interfaces to analyzing the effects of moderators (e.g., caffeine) on cognition. Projects have also been created to simulate children's cognitive development. Therefore, behind this general purpose, the research objectives are: (1) creating models that more accurately reflect human behavior, (2) supporting models to interact with a wider range of interfaces and more accurately (St. Amant et al. 2004), (3) summarizing human behavior in reviews through the gathering of additional data to support model building, (4) developing models that are easy to understand and that are reusable, and (5) building modeling tools. In this paper, the authors show the tools used in ACS Lab in order to summarize the benefits in their application and to point out how they might be reused. Further information is available at http://acs.ist.psu.edu/.

L. Inguscio (✉)
Department of Psychology, University of Rome "La Sapienza", via dei Marsi 78, 00185, Rome
E-mail: lucio.inguscio@uniroma1.it

F. E. Ritter
School of Information Sciences and Technology, Pennsylvania State University, University Park, PA 16802, USA
E-mail: frank.ritter@ist.psu.edu

Behavioral moderators and CafeNav suite

A behavioral moderator, broadly defined, is any substance or factor that causes a change to an individual's physiological state there by altering the behavior of an individual.

Current and forthcoming research have indicated that behavioral moderators play a significant role in almost every human behavior (Tomaka and Blascovich 1993). Since the Applied Cognitive Science Lab is very interested in both learning from and replicating human behavior, it is important to us to study and hypothesize about the effects of behavioral moderators.

Our current hypothesis is that behavioral moderators are quantifiable and that specific moderators affect cognition and behavior in a consistent relationship. Furthermore, we believe that we can derive the representational formula of this relationship and implement in a Cognitive Architecture Overlay (Ritter et al. 2004). The specific behavioral moderator that we are choosing to examine currently is caffeine.

Caffeine is the most popular psychoactive substance in the world; we are not yet able to model its effects on cognition. For this to be possible, cognitive architectures must advance by adding overlays. Caffeine's cognitive effects are entirely dependent on how much caffeine is currently in the body. Therefore, the pharmacokinetics of caffeine must be examined with particular attention on uptake and decay rates and dose-response curves. Cognitive effects of caffeine appear to include: faster reaction times, faster semantic processing, faster logical processing, and increased subjective alertness (Liebermam et al. 2002).We are using four specific tasks with corresponding ACT-R models to gather and analyze participants data.

1. Signal detection: In this task, we are using signal detection theory to analyze the participant's ability to detect and to respond to ambiguous stimuli. In this task, participants are asked to respond when they see

a circle. The circles appear for only 300 ms and they are displayed at quasi-random locations on the screen. Alternatively, squares can appear; however, participants are asked to ignore these squares as noise (false stimuli). We collect data throughout this process, recording accuracy and response time.
2. MODS task: In this task, developed by Lovett, Reder, and Lebiere, working memory capacity is tested by asking the participant to audibly repeat a list of letters that are displayed one at a time and then remember the digit presented at the end of the list. There are three to six letter lists presented on each trial, and thus the participant is asked to recall a three to six digit number per trial.
3. Driving task: In this task, we asked participants to play a video-game driving task (Fig. 1). Throughout this task, we collect participants' data monitoring the key presses that control the driving process.
4. Serial subtraction: In this task, we ask participants to start subtracting seven or thirteen from a given value. We assess their mood and emotional status in order to model the affects of emotion and worry. We have conducted and will continue to conduct research on the effects of behavioral moderators on a serial subtraction task. Currently, we are upgrading our cognitive model from the ACT-R 4.0 architecture to the ACT-R 5.0 architecture (Anderson 1998).

We have started to collect data from 135 subjects performing most of these tasks, the CafeNav suite, under a 3×3 design (0, 200, 400 mg caffeine), (serial subtraction, Argus and the driving task).

Testing these models will be useful for at least three things. First, they will help to validate the models. Currently, moderators are included in models, but it is not clear that the changes lead to more accurate behavior, they may just lead to different behavior. Second, they will tell us where to improve the models. The test that shows that a modified model's behavior is similar to human's one is useful to validate the model. The tests that show the model's behavior is different will also show where to improve the model. However, this is more valuable in some sense, because it leads to progress. Finally, this will lead to a better science of human behavior.

dTank 2.0

The project in the previous section offers a significant advantage: including the effects of the behavioral moderators in models will lead to more believable and useful agents to train with, to train against, and to use within synthetic environments. Improving models in this way represents a move from multiple simple opponents to a smaller number of more intelligent opponents (e.g., the increasing complexity).

Modeling teams are becoming an increasingly active topic for the research communities of multi-agent systems, cognitive modeling, and decision making. Many research efforts have tried to use software-agents to model teamwork. However, we found little work has been done comparing different methods with common experiments. The agent-based modeling and behavior representation (AMBR) project has compared different cognitive process modeling in a military simulation environment (Gluck and Pew 2001), but it has only examined single agent behavior so far. In teamwork modeling, different architectures or models are often evaluated with different domains, which are not comparable. For example, STEAM (a Shell for Teamwork) used a helicopter combat scenario as a simulator (Tambe 1997), while collaborative agents for simulating teamwork (CAST) used a Wumpus grid-world as its test-bed (Yen et al. 2001). Consequently, it is difficult to evaluate teamwork performance between two architectures. As a first attempt to address the issue, we conducted an empirical study to compare team performance. First, we built agent teams with two different architectures: CAST and Soar. Then, we tested them within the dTank simulation.

dTank is a tank game simulator for agents in a distributed environment. dTank provides an architecture and platform neutral test-bed for adversarial real-time cognitive models, and a tool for teaching cognitive modeling, and agent creation. It was inspired by the Tank-Soar, and ModSAF systems.

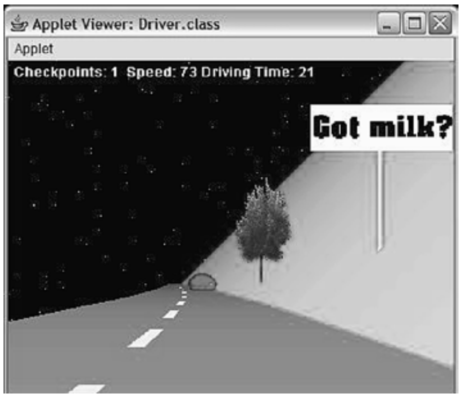
Applet Viewer: Driver.class
Applet
Checkpoints: 1 Speed: 73 Driving Time: 21
Got milk?

Fig. 1 The screen displayed during the driving task

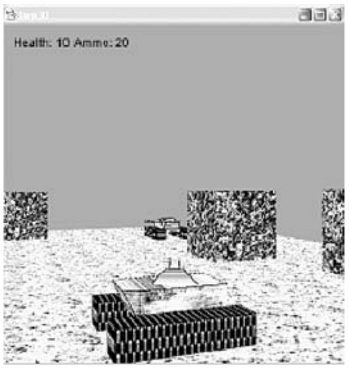

Fig. 2 The 3D interface used by the human

dTank 2.0, like Tank-Soar, is a simulation game, created by Isaac Councill, where a user's tank wages battle with one to many agent-controlled tanks or other users. A first-person view of the game is also available, developed by Alexander Wood (in http://acs.ist.psu.edu/dTank). Agents to play dTank are available in Java, Jess, Soar, and Herbal (Soar).

Both agents and the server interpret the game like a 2D world. The 3D interface used by the human player is an interpretation of the 2D world (Fig. 2).

Although agents live in a 2D world, they have a vision component and are limited to seeing only what a human could see with the 3D interface (a small step toward what has been called the fog of war). Therefore, although the 2D plan view display is available to the user, its use should be limited to demonstration work with agents, and to debugging; it should not (and has not been enabled to) be used for modeling human behavior. This 2D interface eases debugging by allowing the modeler to see tank actions and determine if those actions make sense or if an agent needs revision.

Results

As individual models become aggregated into teams of models, comparing architectures for teamwork modeling will become more important for cognitive engineers making decisions on choosing tools and implementing models. Through a set of experiments, we compared two architectures and their behaviors. Compared with Soar, CAST has more features that are designed specifically for modeling teams, and performed well where increased communication was useful. Furthermore, we grouped the knowledge into domain-dependent knowledge and domain-independent knowledge. The domain- independent knowledge is needed to compliment certain features from the architecture. In the Soar team, for example, knowledge about how to aim a gun at an enemy is domain-dependent; knowledge on how to compose messages is domain-independent. In CAST, communication is a part of the architecture. Therefore, the boundary between domain-independent knowledge and architecture can be blurred. Different architectures may be able to capture or use different domain-dependent knowledge. When we compare the knowledge coded for CAST and Soar, we find that Soar can incorporate more knowledge for making decisions. Although the above findings are not conclusive, we have learned lessons on the relations between architecture and knowledge: (1) by including equivalent knowledge, a Soar team can perform collaborative behavior that is partially similar to a CAST team; (2) some team behavioral differences are difficult to resolve with knowledge alone; (3) the experiments suggest that capturing teamwork behaviors as a part of the architecture or as a part of the agent's knowledge is important and perhaps equivalent decision choice for team modeling; (4) human teams may vary on the teamwork behavior. Is it also affected by differences in knowledge? The question will motivate us to collect human data and compare with the models.

Purpose of dTank

To summarize the usefulness of this project, dTank is intended to serve in three distinct but related roles, teaching, cognitive modeling, and agent creation.

1 *As a teaching tool.* dTank provides a simple, well-documented environment for experimenting with agent programming. A sample API for Soar agents is included with the dTank distribution, and instructions exist within this manual for creating interfaces for other types of agents. dTank supports protocol capture from human players. Students can use dTank to study cognitive modeling techniques including model-to-data fitting, in addition to general AI applications.
2 *As a modeling tool.* Due to the protocol capture facility mentioned above, as well as dTank's communication layer, dTank provides a good environment for modeling both individual and team phenomena. Any cognitive architecture that can be made to support socket communication can interact with dTank. The communication layer is general enough that virtually any theory of multi-agent communication can be tested. One experimental setting could be, for example, studying how a user works with a social environment defined by a set of agents with known characteristics, knowledge, and behavior.
3 *As a developmental test-bed for advanced AI applications.* The primary reason for the development of dTank was to create a tool to investigate the usability of distributed AI (multiagent) systems. In particular, the ACS Lab is using dTank to inform the development of tools that help to explain the behavior (both actual and intended) of complex cognitive models to that of the modeler and human opponents of these models. This effort is aimed to improve the usability and usefulness of applied agent technologies, as well as to provide improved facilities for the validation of model behavior.

Choosing and getting started with a cognitive architecture

Cognitive architectures are useful to a wide variety of users. Computer scientists, psychologists, cognitive scientists, and various domain experts can all use cognitive architectures. Unfortunately, designing, implementing, and using cognitive architectures can be a difficult task considering that the background and expertise of this diverse set of users varies from novice to expert. In addition, the tasks performed by users of such architectures can vary considerably, and can include a wide variety of tasks. It is essential that development environments are created to allow the modeler to focus more on the problem domain, and less on the details of a particular architecture, in order to accommodate the wide range of users and to promote the use of cognitive systems. An example answer is herbal.

Herbal

We are working on an integrated development environment called Herbal that acts as a first step toward creating development tools that support the wide range of users of cognitive models. Users can create models graphically using it, and they can have these models compiled into Soar productions. The productions will work as any Soar model does. The structure of the model can be passed to an associated tool that can help to display and to explain the model and how the model runs. Last fall, in IST at Penn State University, 35 students used it in a senior-level course on cognitive modeling.

The design of herbal is based on question that users ask about a model (Council et al. 2003).

A tutorial exists for the herbal integrated development environment (IDE) (version 0.9) and Herbal Viewer (version 0.8); the latest versions are available for download at http://acs.ist.psu.edu/projects/herbal (Cohen 2005).

A short note on the Dismal spreadsheet

Sequential data

Dismal was developed to support sequential data (protocol) analysis. It provides typical support for exploratory sequential data analysis, such as the ability to compute word counts, to search for lines matching given patterns, and to semi-automatically assign codes to segments. Furthermore, it supports aligning predictions with sequential data, for example, a model trace and verbal utterances. When a complete description of correspondences can be provided such as keystrokes by the subject and the model, the two sequences can be automatically aligned. When the comparison is less clear, such as between a subject's verbal utterances and the representations in the model, semi-automatic commands allow an analyst to align items by pointing. Together, these commands substantially reduce the work of testing cognitive models with protocols by up to a factor of five (Ritter and Larkin 1994), allowing such analyses to be performed more often and with more insight.

HCI experimentation

Dismal is useful in teaching principles of HCI because it is instrumented—it is possible to automatically record each user action and the time it occurred. We have found that undergraduates are able to use this in 5 week practical to gather realistic user data and test HCI theories [e.g., the keystroke model of Card et al. (1983)]. As the source code is provided, dismal can be used itself as a test bed for evaluating various interface designs, and comparisons can be based on actual user data.

Extendibility through implementation in GNU Emacs

Dismal is implemented as part of and it is now included in the distribution for the GNU-Emacs editor, which leads to four relatively novel features. (1) Dismal is mainly keystroke driven although it can be partially mouse and menu driven. For example, the key binding to cut a region of text now cuts a range of cells. This leads to ease of use for expert users, a high transfer of skill for Emacs users, and a large potential user base. (2) Dismal is directly extendible because its Lisp source code is provided. This allows users to write their own custom commands, for example, to count the occurrences of sequences in a column. (3) Its architecture is designed to allow it to be given commands from an associated program. This permits models of human behavior to use it as an interactive task environment. These models can either be written in the native Lisp code or called as an associated Unix process.

More about GNU and dismal at http://www.gnu.ai.mit.edu/ (Ritter and Wood 2005).

Categorical data display—CaDaDis

Categorical data display is a tool that shows the inner workings of a cognitive model through a visualization. The screenshot is an example of the Waterjug demo running on Soar (Fig. 3).

It is our belief that these multiple views into the model help in learning how to develop cognitive models, debugging models, comparing models, and comparing architectures. Agent displays that are generated automatically are not uncommon in the area of cognitive

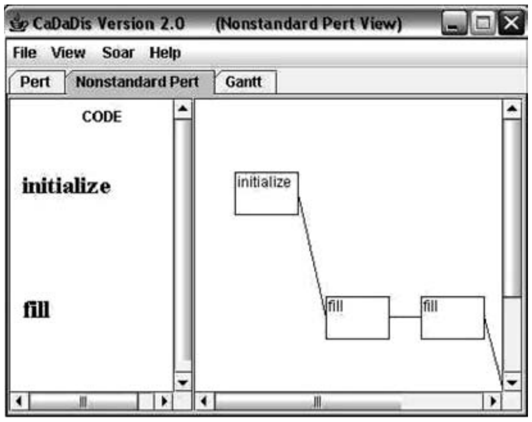

Fig. 3 A Soar model for waterjug task

science. For example, there were similar displays in CPM/GOMS, APEX, and the TSI.

Our display offers two Pert Charts and one Gantt Chart. Other displays can and will be included in the future. The two Pert Charts are the standard Pert Chart and the variation based on the work of Bonnie John et al. (2002) where the action occurrence is displayed, but not the length. The CaDaDis can be used in conjunction with different cognitive architectures since the graphics stem from messages. As long as messages can be pulled from the agent, a display can be generated (Tor et al. 2004).

Categorical Data Display is written in Java using the Vista package created by SoarTech. Currently, there are three cognitive architectures that are able to create displays, Soar, ACT-R, and JESS. JACK and CAST are the next two to be included.

Preliminary results show that CaDaDis is successful in showing model behavior. It can create unique displays showing information with more clarity than textual traces. It provides nice displays of model activity in two different architecture. Furthermore, it can prove useful in debugging cognitive models by analyzing rule usage, whether certain operators fire, and so on. The reuse of these displays with cognitive architectures (Soar and ACT-R) suggest that the first major reuse of cognitive modeling and agent behavior may be in interface design and not in the knowledge. This might not be that surprising, given that the interface code looks more like the code that gets reused now. Interfaces make up about 50% of most systems (Meyers and Rossen 1992). If this is true, which we believe, it can be for cognitive models and agents using CaDaDis, this is a very worthwhile result.

Acknowledgments The authors wish to thank Andrew L. Reifers and Geoffrey P. Morgan for material from their web sites that we used.

References

Anderson JR, Lebiere C (1998) The atomic components of thought. Lawrence Erlbaum, Mahwah, NJ

Card S, Moran T, Newell A (1983) The psychological human-computer interaction. Lawrence Erlbaum Associates, Hillsdale, NJ

Cohen M (2005) Version 0.9 of the Herbal IDE/Viewer Tutorial, from http://acs.ist.psu.edu/projects/herbal/index.html

Council IG, Haynes SR, Ritter F (2003) Explaining soar: analysis of existing tools and user information requirements. In: Proceedings of the 5th international conference on cognitive modeling, Bamberg

Gluck KA, Pew RW (2001) Overview of the agent-based modeling and behavior representation (AMRB) model comparison project. In: Proceedings of the 10th computer generated forces and behavioral representation conference

John B, Vera A, Matessa M (2002) Automating CPM-GOMS. In: Proceedings of the CHI'02 conference on human factors in computer systems, New York

Liebermam HR, Thairon WJ, Shukitt-Hale B, Speckman KL, Tulley R (2002) Effects of caffeine, sleep loss, and stress on cognitive performance and mood during U.S Navy SEAL training, from http://link.springer.de/link/eservice/journals/00213/contents/02/01217/paper/s00213-002-12179ch000c.html

Meyers BA, Rossen MB (1992) Survey on user interface programming. In: Proceedings of CHI'92, New York

Ritter FE, Larkin JH (1994) Using process models to summarize sequences of human actions. Hum-Comput Interact 9:345–383

Ritter FE, Wood A (1991, revised every 6 months since 1991). Dismal, a spreadsheet for GNU Emacs, from http://www.gnu.org/software/dismal

Ritter FE, Wood AB (2005) Dismal: a spreadsheet for sequential data analysis and HCI experimentation. Behavior Research Methods, Instruments, and Computers

Ritter FE, Reifers A, Klein L, Quigley K, Schoelles M (2004) Using cognitive modeling to study behavior moderators: pre-task appraisal and anxiety. In: Proceedings of the human factors and ergonomics society, Santa Monica, CA

St. Amant R, Horton TE, Ritter, FE (2004) Model-based evaluation of cell phone menu interaction. In: Proceedings of the CHI'04 conference on human factors in computers systems, New York

Tambe M (1997) Towards flexible teamwork. J Artif Intell 33(1): 1–64

Tomaka J, Blascovich J, Kelsey RM, Leitten CL (1993) Subjective, physiological and behavioral effects of threat and challenge appraisal. J Pers Soc Psychol 65:248–260

Tor K, Haynes SR, Ritter FE, Cohen MA (2004) Categorical data displays generated from three cognitive architectures illustrate their behavior. In: Proceedings of the 17th international joint conference on artificial intelligence (IJCAI-01), Los Altos

Yen J, Yin J, Ioerger TR, Miller MS, Xu D (2001) CAST: collaborative agent for simulating teamwork. In: Proceedings of the 17th international joint conference on artificial intelligence (IJCAI-01), Los Altos