Collaboration and Modeling Support in CogLaborate

Reuben Cornel
Robert St. Amant
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
reuben.cornel@gmail.com, stamant@csc.ncsu.edu

Jeff Shrager
Stanford University Symbolic Systems Program (consulting)
Palo Alto, CA 94301
jshrager@stanford.edu

ABSTRACT: *This paper describes the CogLaborate system, a collaborative, tool-based environment for the ACT-R cognitive modeling community. CogLaborate is based on BioBike, which supports collaboration between biologists and computer scientists. This paper discusses how comparable benefits can be brought to cognitive modelers, and presents the design of CogLaborate, its frame-based representation for models, and a proof of concept in the form of an ACT-R module developed within the environment.*

1.   Introduction

Research on cognitive modeling has driven the formation of active, thriving communities. With ACT-R, for example, beyond the core group of researchers at Carnegie Mellon University, we have annual workshops, a summer school to introduce new researchers to the framework, a Web site, an active mailing list, and any number of small interdisciplinary groups of collaborators distributed throughout the world. The result has been a continuous stream of refinements and extensions to ACT-R, both the theory and the software architecture, as well as models, experiments, development tools, and the like.

In important ways the ACT-R research community is not unique *as* a community. Consider a vision of online communities that dates back to 1968 [Licklider and Taylor, 1968]:

> They will be communities not of common location, but of common interest. In each field, the overall community of interest will be large enough to support a comprehensive system of field-oriented programs and data.

A subfield of human-computer interaction, computer-supported collaborative work (CSCW), has produced a variety of concepts and tools based on this vision to help support collaboration between people and to foster online communities. The research described in this paper is an attempt to build a collaborative online environment for cognitive modelers, to explore the potential benefits of a CSCW approach to the field. We have developed a system called CogLaborate for this purpose.

In contrast to related research on extending the scope of modeling efforts beyond individual researchers and small teams (e.g. [Gluck et al., 2007]), the focus of CogLaborate is on model development rather than model execution. CogLaborate currently runs in prototype form on the Cyano server at the Carnegie Institution of Washington in Washington DC and client machines at the North Carolina State University. We have built CogLaborate to support the following:

- *Sharing of architecture extensions and running models.* Some extensions to ACT-R are more difficult to set up than others. In CogLaborate, such extensions can be tested and uploaded by modeling researchers to a shared environment for others to use immediately, saving repeated effort. Further, in contrast to a static model repository or a

conventional software configuration management system, CogLaborate can maintain models in a long-running Lisp environment, where they can be ready to execute, paused in their execution, or even executing in the long term.

- *Sharing of software and hardware resources* to support the development and dissemination of models and modeling software. Although CogLaborate does not approach the model execution capabilities of other systems (e.g. [Gluck et al., 2007]), it outmatches the performance of our local machines, even given network communication overhead.

- *Support for model analysis tools.* One important aspect of the CogLaborate project is the potential to support analysis of the structure and content of models. CogLaborate translates ACT-R models into a frame-based representation [Minsky, 1974], to support search and browsing by modelers. This means that procedures for analyzing models (currently under development) need not parse ACT-R code directly; instead they can rely on a slightly more abstract and uniformly structured representation.

CogLaborate is a new system, and we have not yet evaluated how and whether collaboration can benefit cognitive modeling research. Even in its prototype state, however, the promise of CogLaborate can be seen in two ways. First, we believe that a frame-based representation offers significant advantages for sharing and analyzing models, in comparison with their storage as modeling code. Second, we have exercised CogLaborate by building a specialized ACT-R module that relies on an existing extension to ACT-R (WN-Lexical) [Emond, 2006] and a model to test the new module. This experience exposed some of the procedural difficulties in carrying out such a task as well as the benefit that CogLaborate could provide the cognitive modeling community. We believe that our proof of concept—a new model running on an ACT-R extension that requires no more effort to install than logging into a remote server—demonstrates the value of our approach.

2. BioBike

CogLaborate is built on the Biobike platform. BioBike is an instantiation of KnowOS [Travers et al., 2005], a refinement of the concept of the operating system. Operating systems provide useful abstractions for users to work with the elements of a system. Files, for example, abstract away the details of how data is stored on hardware, and an OS provides functions for creating, managing, and manipulating data using this abstraction. The KnowOS vision extends this analogy to the realm of knowledge. An implementation of the KnowOS consists of the following layers [Travers et al., 2005]:

- A knowledge base, in a frame representation.
- An extensible programming language with appropriate abstractions for users to work with the system.
- A interface to the programming language and to other KnowOS services.

BioBike (originally known as BioLingua) provides biologists with the ability to perform computational biology operations on large data sets using a simple language [Massar et al., 2005]. BioBike ties a number of knowledge bases together transparently, using frames to represent organisms. As a KnowOS, it provides features customized for molecular biologists. These include

- A common framework to access genomic, metabolic, and experimental data.
- A general-purpose programming language (Lisp) customized for transparent access to the underlying knowledge bases.
- A highly interactive environment where code can be evaluated and its results displayed immediately.
- A number of general-purpose tools that help in analyzing interactions.
- A wiki through which scientists can collaborate and announce results.

BioBike provides biologists, in principle, with an environment in which they interact with the computer in the same terms as they would interact with their peers; with a uniform framework for accessing knowledge from a number of different knowledge bases; and with a common work area where data and results can be shared and external tools can be integrated. BioBike has been in place over a number years and has demonstrated benefits to collaborating teams of biologists and computer scientists during that time [Massar et al., 2005].

From a CSCW perspective [Rodden, 1991], the type of collaboration BioBike is designed to support is asyn-
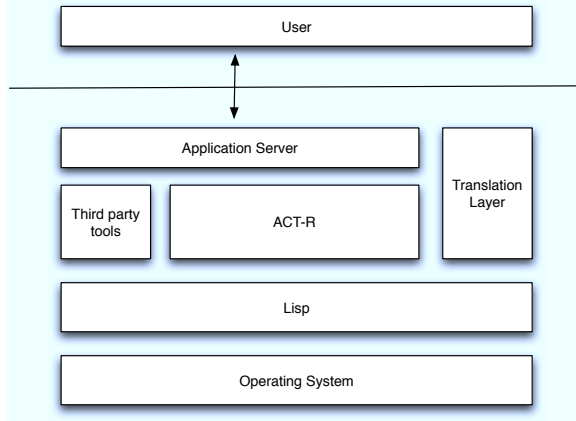
Figure 1: System Design

Figure 2: Frame representation for ACT-R models

chronous (not requiring collaborators to interact in real time) and geographically distributed (not requiring collaborators to be co-located). The synchronous/asynchronous and co-located/distributed distinctions do not create hard boundaries between categories of CSCW systems, but they do help us distinguish between message systems, conferencing systems, meeting rooms systems, and co-authoring systems. Of these categories, BioBike can be seen most naturally as an example of the last.

Figure 1 provides a high-level overview of CogLaborate, implemented on the BioBike chassis. Users interact through a Web-based application server with ACT-R and its third party extensions. The translation layer runs side by side with ACT-R, creating frame-based representations of ACT-R models when they are loaded and compiled; the user has access both to ACT-R and to these representations. These components are layered on top of a Lisp environment, which in turn runs on the operating system of the servers. This organization is fleshed out in more detail in Section 4.

The ACT-R component in CogLaborate replaces biology-specific functionality in BioBike; the modular structure of BioBike made this feasible. CogLaborate added only about 1,000 lines of new code to the existing code bases of ACT-R and BioBike.

3. Model representation

ACT-R models are essentially Lisp data structures. One plausible representation of models in CogLaborate is simply the Lisp code that defines models at the top level. This approach has a few disadvantages, how-
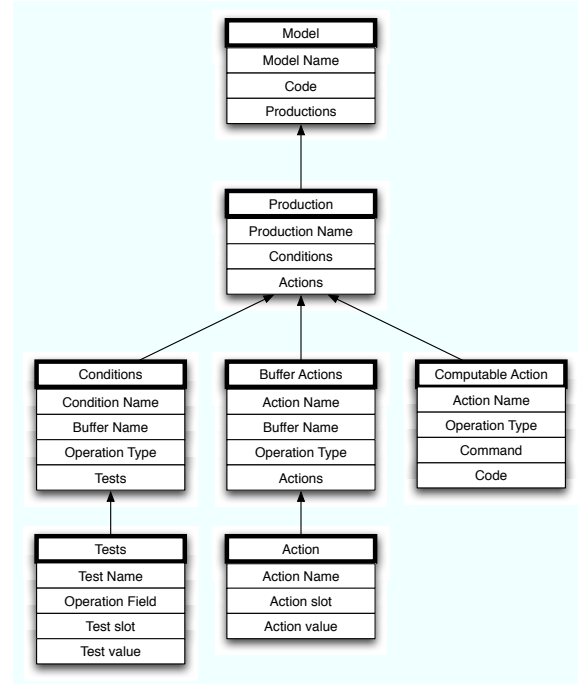
ever. A direct representation exposes search, browsing, and analysis tools to the syntax and structure of models, in some cases requiring parsing at the textual level. (For example, forms such as =goal> and +goal> are related—they access the goal buffer—but they are not tokenized as such by the Lisp reader.) Other software engineering issues arise as well in the context of collaboration, such as the difficulty of managing meta-data associated with models and knowledge structures (e.g., for version control).

Instead, CogLaborate adopts a frame representation. Frames were introduced by Marvin Minsky [Minsky, 1974] in a seminal paper on knowledge representation. Frames are structures that can represent objects, situations, and concepts. Frames are arranged in a parent-child hierarchical taxonomy, with child frames representing specializations of their parents [Karp, 1993]. A frame contains slots that define the properties of the object being represented by the frame. Slots can also represent relationships between two frames.

CogLaborate provides translation between the Lisp source code of models and a frame representation, in both directions. Descriptions of the frames for representing models are given below; their structure is shown in Figure 2.

- *The model frame* represents an ACT-R model. It consists of a code slot that holds all the code that is required by the model, including code for initialization of the model, chunk definitions for the model, and miscellaneous utility functions that may be required by the model. It also has a slot for productions.

- *Production frames* contain a conditions slot, which defines the tests that are required for the condition to fire, and an actions slot, which lists all the actions that will be executed if that production is fired.

- *Test frames* capture the buffer tests that are part of conditions in a production. Each test frame represents one such test. A test frame has a slot to represent individual clauses within the test.

- *Conditions frames* represent an individual clause consisting of a test field and a value field for comparison of a buffer slot and a value. The value field can also hold variables, as is common in ACT-R productions.

- *Buffer actions frames* hold actions that can modify, clear, or retrieve a chunk in a buffer.

- *Action frames* represent individual clauses for modifications to a buffer.

- *Computable Action frames* specify actions executed by the ACT-R architecture that have side-effects, such as printing information to the screen.

The AllegroServe Web Application server acts as a front end for interaction with CogLaborate. When a model is evaluated in CogLaborate, it is compiled by ACT-R, running on the server. CogLaborate code is plugged into the ACT-R compiler to allow access to the internal data structures generated as the model is parsed. This model representation is then converted into frames as described above. The frame-based representation thus exists side by side with the source model code (as well as with the running model).

4. Using CogLaborate

Briefly, cognitive modelers using CogLaborate for ACT-R development rely on a Lisp listener in a Web browser, where code can be evaluated; a structured representation for models in frames; and mechanisms for sharing and examining models at different levels of detail.

The user interacts with the CogLaborate sys-

tem through a Web interface. On logging in, users are put into the ACT-R package. Models are submitted through the Web interface in their source code representation, with code wrapped in a `with-user-meta-process` form. This macro creates a new meta-process for each user and allows models to be run without conflict with other users of the system, who may be running their own models at the same time. No other changes to model code are required for use in CogLaborate.

Development on CogLaborate up to the present has focused on basic functionality, which means that the Web interface does not provide as rich an environment as the graphical user interface to ACT-R. The workflow of using CogLaborate in its current state means building and testing models and architecture extensions locally before uploading the work to the server. Even though it is possible to build models completely from scratch in CogLaborate, a more efficient workflow for model development must await further work on the front end.

Let's consider a slightly more detailed scenario to illustrate the use of the system. A user creates a model and evaluates it in CogLaborate. This is done by entering a model into the Lisp listener displayed in the Web interface, as shown in Figure 3. The Lisp listener has two text boxes. The larger text area is used to enter complete models; the smaller text box to enter individual commands.

Once a model is entered into CogLaborate, it can be accessed (via its name) by any other user of the system, through a simple search. The model resulting from the search is displayed in its frame representation. The model can be navigated by active links corresponding to the slots of the current frame, whether at the level of models, productions, or lower in the frame hierarchy. To see the source code of the model, users can click the Frame→Listener link on the index page of the model. The result is shown in Figure 4.

5. A proof of concept

To evaluate the capabilities of CogLaborate we built a simple, medium-scale model. The point of this exercise is twofold. First, it shows that the system is capable of supporting a non-trivial cognitive modeling effort. Second, it demonstrates the level of maturity of the system. This section discusses the problem description, the approach we took to solving the problem,

Figure 3: A user creates and evaluates a model

and what we learned through the exercise.

In a crossword puzzle, words or phrases are positioned in an interlocking grid, horizontally and vertically. The words are to be guessed by a set of clues that define the words or phrases. Our proof-of-concept problem is a crossword puzzle where the clues and the solutions are synonyms of each other.

This problem is appropriate for the following reasons: it demonstrates that the system is ready to solve practical problems; it shows that the system can be used to write and test an ACT-R module, with the environment acting as a sandbox; finally, it places considerable demands on the hardware of the computer, in terms of memory and CPU.

The crosswords are generated by a new *Crossword* module for ACT-R. This module relies on information from the WNLexical module [Emond, 2006], which enables ACT-R to make use of the WordNet lexical database. WordNet is [Miller, 1995] "an online lexical database designed for use under program control. English nouns, verbs, adjectives, and adverbs are organized into sets of synonyms [synsets], each representing a lexicalized concept. Semantic relations link the synonym sets."

Each clue is represented as a list that consists of the starting co-ordinates of the word, the direction (across or down), the clue string, a location to put in a solution, and the actual solution. These data structures are manipulated by the crossword module, which translates clues into chunks. It can also set words in specific locations, verify that the crossword solution under construction respects the constraints of the puzzle, and return results from queries about the parameters of a specific clue. The module maintains the current state of the crossword solution, with some entries filled in and others empty.
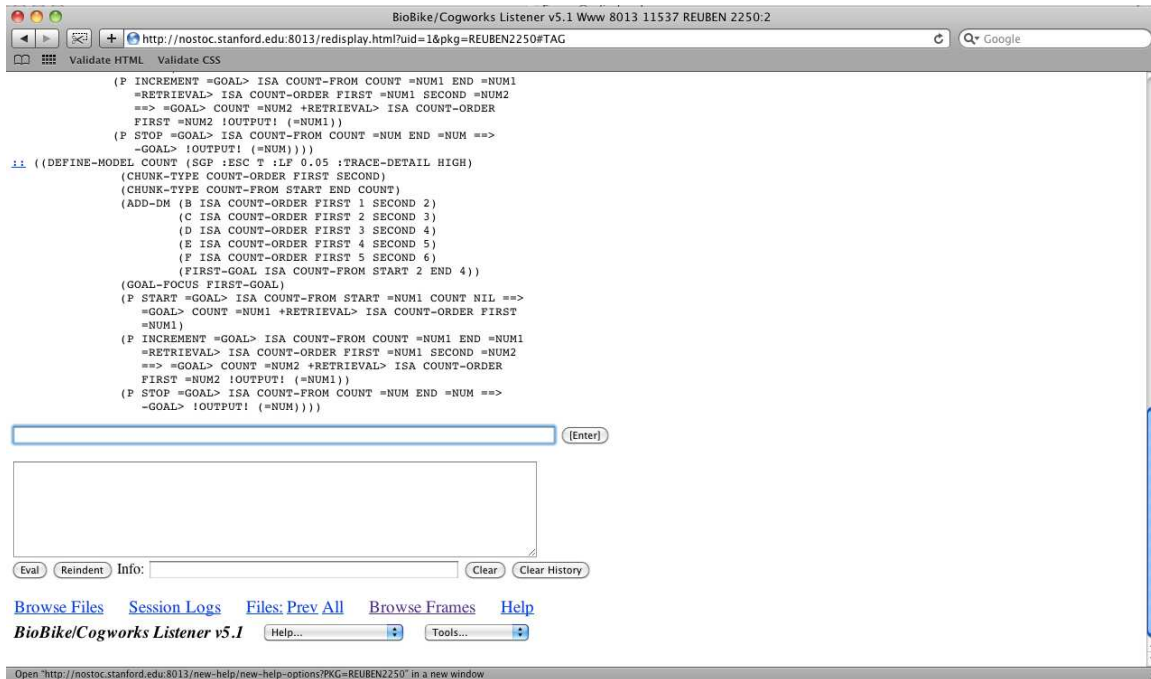
Figure 4: A user displays the source of a model

When the model is run it defines three chunk types, one for clues and two for maintenance of the state of the crossword problem as it is being solved. The basic problem-solving strategy the model follows is to check memory for clues that have not been added to the puzzle representation. If one is found, it is used to retrieve all the synsets of the clue word via the `wn-lexical` buffer. (A single word may have more than one synset.) For every synset found a chunk is created with the `imaginal` buffer. If the word is not found, this results in an error. For each synset chunk, its corresponding words are tested against the constraints of the puzzle by the crossword module, which also marks the clue as being solved. This process repeats until all the clues have been solved or have been marked as being unsolvable.

This is not intended to be a cognitively plausible model of crossword puzzle solving, but rather to exercise CogLaborate. The model consists of sixteen productions with a total of about four hundred and sixty lines of code, which can be fairly described as medium-sized. The source for the model and a sample execution trace, as well as the Crossword module, are publicly available but are not given here due to space limitations [Cornel, 2009].

During the development of the Crossword module, a difficulty arose when an older version of the WNLexical was used; we were not aware that a newer version was available that contained a bug fix we needed. This caused us some wasted time. The conventional lesson learned is that developers should consider such possible sources of problems, but another possibility is that dissemination of modules (along with models and other software to support modeling) might be improved with a centralized resource for modeling such as CogLaborate.

6. Discussion

The concept of repositories for cognitive models is not new, and there has been continuing interest in establishing such shared resources.[1] Such resources can have obvious benefits: improved access to computational capabilities, a stable and growing body of explicitly expressed knowledge about a domain, and so forth. Our work on CogLaborate explores a new dimension of potential benefits for cognitive modeling research: collaboration.

On creating a frame-based abstraction for ACT-R

---

[1] A panel at the Biologically Inspired Cognitive Architectures symposium, at the 2009 AAAAI Fall Symposium Series, was devoted to this topic.

models it quickly became clear that this representation could be used to explore a number of other possibilities beyond our original conception of CogLaborate. As observed by Langley et al. [Langley et al., 2009], an important issue facing cognitive modeling is support for software reuse. This project promotes reuse of models in the sense that the representation allows for models to be represented, analyzed, and distributed in a more transparent fashion than in their current representation as Lisp code. Today, it is impossible to determine the similarity between two ACT-R models except through code inspection and ad hoc judgments. The frame-based representation introduced in this research makes more sophisticated analysis possible: comparison of the use of buffers across productions, for example. Such analyses remain for future work.

It's important to point the limitations of this approach, which may not be obvious given the current prototype state of CogLaborate. The frame representation provides no significant level of abstraction over the structure of an ACT-R model—there is a one-to-one correspondence between ACT-R knowledge structures and frames. A critical question that remains to be answered is whether meaningful commonality and contrasts between models can be made at this detailed level. For example, consider an ACT-R model that solves a problem such as the Tower of Hanoi (e.g. [Anderson and Douglass, 2001]). Such models are sometimes described in terms of the problem-solving strategies that they capture, but strategies can be explicit, emergent, or somewhere in between. Analysis of the knowledge structures explicit in a model may possibly give insight into higher-level cognitive abstractions that are present, but there are no guarantees. It will be worthwhile for us to consider recent research on high-level behavior representation languages [Ritter et al., 2006], where issues of abstraction are considered explicitly.

Another interesting research direction is to investigate software reuse as provided by object-oriented programming environments. That is, we can develop features such that models can inherit behavior from other more general models. This way we should be able identify general patterns that emerge from human cognition. A third and obvious possibility is the investigation of user interfaces that allow cognitive scientists to create models without having to learn Lisp; the issue of cognitive modeling languages and ease of modeling is a continuing concern in the field [Ritter et al., 2006].

Some of the core features of CogLaborate are par-

tially supported by other systems. For example, conventional systems for source control provide some of the same benefits as CogLaborate, as do model repositories such as the ACT-R Web site (http://act-r.psy.cmu.edu/models/), which even includes a few Web-based simulations. We believe that CogLaborate demonstrates new possibilities. The most interesting for us are the following:

- CogLaborate can be used as a collaborative sandbox for learning and exploration in modeling. Access to a shared environment in which models and even modeling processes can exist for long periods of time provides continuity and a persistent context for the exchange of ideas. We expect this to be most useful for remote collaborations.

- CogLaborate, with its frame representation of models, supports the development of new techniques for development, analysis, and comparison. Does my new model share structure with any existing models already in the environment? How different are two models for the same task, developed for different versions of the ACT-R architecture?

We are actively building on these ideas.

7. Acknowledgments

References

[Anderson and Douglass, 2001] Anderson, J. R. and Douglass, S. (2001). Tower of hanoi: Evidence for the cost of goal retrieval. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27(6):1331–1346.

[Cornel, 2009] Cornel, R. (2009). Coglaborate - an environment for collaborative cognitive modeling. Master's thesis, Department of Computer Science, North Carolina State University.

[Emond, 2006] Emond, B. (2006). WN-LEXICAL: An ACT-R module built from the WordNet lexical

database. In *Proceedings of the Seventh International Conference on Cognitive Modeling*.

[Gluck et al., 2007] Gluck, K., Scheutz, M., Gunzelmann, G., Harris, J., and Kershner, J. (2007). Combinatorics meets processing power: Large-scale computational resources for BRIMS. In *Proceedings of the Sixteenth Conference on Behavior Representation in Modeling and Simulation*, pages 73–83. Citeseer.

[Karp, 1993] Karp, P. D. (1993). The design space of frame knowledge representation systems. Technical Note 520, SRI International, Menlo Park (CA), USA.

[Langley et al., 2009] Langley, P., Laird, J. E., and Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160.

[Licklider and Taylor, 1968] Licklider, J. and Taylor, R. (1968). The computer as a communication device. *Science and technology*, 76:21–31.

[Massar et al., 2005] Massar, J. P., Travers, M., Elhai, J., and Shrager, J. (2005). Biolingua: a programmable knowledge environment for biologists. *Bioinformatics*, 21(2):199–207.

[Miller, 1995] Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

[Minsky, 1974] Minsky, M. (1974). A framework for representing knowledge. Technical report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA, USA.

[Ritter et al., 2006] 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., and Vera, A. (2006). High-level behavior representation languages revisited (group symposium). In Fum, D., del Missier, F., and Stocco, A., editors, *Proceedings of the Fifth International Conference on Cognitive Modeling (ICCM)*, pages 404–407, Trieste, Italy. Edizioni Goliardiche.

[Rodden, 1991] Rodden, T. (1991). A survey of CSCW systems. *Interacting with Computers*, 3(3):319–353.

[Travers et al., 2005] Travers, M., Massar, J. P., and Shrager, J. (2005). KnowOS: The (re)birth of the knowledge operating system. In *The International Lisp Conference*.

Author Biographies

REUBEN CORNEL, M.S., is a recent graduate of the Knowledge Discovery Laboratory at North Carolina State University. His research interests are in intelligent systems and cognitive science.

ROBERT ST. AMANT, Ph.D., is an Associate Professor in the Department of Computer Science at North Carolina State University. His research explores models of interaction, drawing on concepts in artificial intelligence, human-computer interaction, and cognitive science.

JEFF SHRAGER, Ph.D, is the CTO of CollabRx, and a consulting associate professor in the Symbolic Systems program at Stanford University. As a computational psychologist of science, Dr. Shrager seeks to understand how science works, and to build human-computer networks that facilitate discovery.