Soar and the Construction-Integration Model: Pressing a Button in Two Cognitive Architectures

Cathleen Wharton and Clayton Lewis*

CU-CS-466-90 March 1990

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

* The authors gratefully acknowledge fellowship support to the first author from Hewlett-Packard Laboratories, Palo Alto, California, and support to the second author from US West Advanced Technologies, Denver, Colorado, and the National Science Foundation.

Soar and the Construction-Integration Model: Pressing a Button in Two Cognitive Architectures†

Cathleen Wharton and Clayton Lewis

Institute of Cognitive Science
and Department of Computer Science
University of Colorado at Boulder
Campus Box 430
Boulder, Colorado 80309-0430
cwharton@boulder.colorado.edu

Abstract

Soar and the Construction-Integration Model are two cognitive architectures that have been used to model Human-Computer Interaction tasks. The architectures are in competing classes: one is symbolic, the other associational. To clarify what is at stake in this competition, we compare two models of a simple task, the pressing of a button on an Automatic Teller Machine. Results indicate that while overall processing in the models is similar, low level processing and the amount of structure required of each model is quite different. The Construction-Integration Model implies that interface cues which are associative in nature, e.g. adjacency cues, are fundamental to a good interface design. The Soar architecture on the other hand, does not directly indicate this.

Keywords

Soar, Construction-Integration Model, cognitive architecture, problem solving, associative, symbolic, Human-Computer Interaction

1. Introduction

Like Cognitive Scientists generally, researchers in Human-Computer Interaction (HCI) are weighing the merits of contrasting classes of cognitive architectures: those that feature highly structured information processed by rules and those that feature less structured information processed by propagation of activation. Newell and Card [NeC85] argue that the former class provides the description of choice in HCI, while Norman [Nor88] suggests that the latter kind of model will more fully illuminate the issues. We use the tags *symbolic* and *associational* to indicate these competing, though vaguely-specified, classes of model.

To clarify what is at stake in this competition, we chose an architecture from each class: Soar [LNR87], representing symbolic architectures, and Kintsch's Construction-Integration Model (CI) [Kin88], representing associational architectures. We then constructed within each architecture a model of the performance of a simple HCI task, pressing a button on an Automatic Teller Machine. As would be expected of any powerful, flexible architecture, both Soar and CI permit the task to be modelled. In fact, the overall picture of processing in the two models is much the same. However, the content of the models is quite different: Soar requires explicit knowledge to be supplied at two points in processing at which the CI model relies on general knowledge, implicit to the associative processes in the architecture. An accompanying difference is that the Soar model requires considerable prestructuring to perform the task, while the CI model relies largely on structure induced by situational cues. This contrast is suggestive with respect to arguments about the situated nature of cognition [Gre88, Suc87].

†The authors gratefully acknowledge fellowship support to the first author from Hewlett-Packard Laboratories, Palo Alto, California, and support to the second author from US West Advanced Technologies, Denver, Colorado, and the National Science Foundation. Please address correspondence to the first author.

Using an HCI task to compare architectures confers an advantage. Since cognitive models in HCI have the utilitarian purpose of supporting design, we can ask whether observed differences between models really *matter*, or whether one model could be viewed simply as an approximation to the other, differing from it in detail, but not in practical implications. We conclude that the differences we see here, do indeed matter.

The conclusions we suggest must carry a number of qualifications. First, Soar and CI do not exhaust the possible architectures in their respective classes, and there are other architectures, like ACT* [And83], which combine features of both classes. Second, there are many ways to model performance of any task within an architecture. Another analyst may find a way to model our task within Soar that relies less on explicit knowledge than our model does. Finally, our discussion is not grounded in a complete simulation for each model.

Our paper has the following format. First, we review the two cognitive architectures. This is done to both introduce key terminology and to give those who are unfamiliar with one architecture or the other, a brief overview. Then we describe the HCI task. Next, a Soar model for the task is presented, followed by a CI model for the same task. The models are then compared. The result is a discussion about processing and task prestructuring concerns. Implications of the differences are also presented.

2. The Two Cognitive Architectures: Soar and CI

2.1. Soar

Soar [LNR87] is an architecture designed to perform a wide variety of cognitive tasks, using problem spaces as the primary organizational unit. That is, Soar embodies the *Problem Space Hypothesis* [New80]. The problem solving behavior is dictated by *explicitly coded,* general and task specific problem solving knowledge. Some knowledge is defined by the architecture, while the rest is initially defined by productions. Combined, the task and general knowledge sets allow for the creation, retrieval, and selection of problem spaces, goals, states, and operators, as well as for operator application.

Soar uses an *organizational* approach to memory: the representation, acquisition, retention, and retrieval processes of memory are defined in terms of the organization set forth by a symbolic production system. As such, the architecture has the components indicative of a standard production system: a long-term production (recognition) memory, a short-term working memory, a mechanism for creating new productions through the technique of chunking, a working-memory manager to remove outdated information, and a decision procedure to handle conflict resolution.

The interaction between the components is dictated by the behavior of a Soar decision cycle. The result of decision cycle is to take an action as defined in Newell and Simon's seminal work on Human Problem Solving [NeS72]. A Soar decision cycle may thus result in state or operator selection, subgoal creation due to an impasse, etc. A Soar decision cycle is defined as follows. First, there is a syntactically monotonic elaboration phase, during which all possible productions held in the long-term memory store are fired until quiescence is reached. The productions serve to add information to working memory (WM). Next, the decision procedure determines what action to take, by examining the contents of WM. The decision is based upon the semantic interpretation of discrete preference values that were placed into WM during the elaboration phase. If the decision procedure is successful (it determines that a change is needed given the current context), then the change is made. Learning also occurs should any subgoals subsequently be terminated. If the decision procedure fails (we are at an impasse), then a subgoal is set up in an attempt to resolve the impasse. In either case the decision cycle is now complete and we begin another cycle, with the elaboration phase.

2.2. Construction-Integration Model

The Construction-Integration Model [Kin88] was originally designed to handle discourse comprehension tasks, but has recently been extended to the area of problem solving [Man89]. Problem solving behavior in CI, is primarily based on task specific knowledge. Unlike Soar, CI does not incorporate explicit top-down knowledge of problem spaces or their generation. Instead, the problem solving process is *implicit*, and basically bottom-up. The problem solving process relies on the implicit semantic relationships between nodes in the network, instead of explicit knowledge as does Soar. Consequently, CI behavior is not viewed as a path through a tree of explicitly defined subgoals, instead it can be viewed as a dynamic topographical map. The highest point of elevation on this map changes with each decision cycle, defining the current goal context and action to take. The highest point refers to the most active node in an associative network.

In contrast to the Soar approach to memory, CI relies upon the principles of *associationism*: the memory processes are based on an associative network. In this network, nodes represent domain specific knowledge and links specify node interrelationships. A localist representation is used and two primary types of nodes exist: propositional nodes and action nodes. There are two link types, excitatory and inhibitory. Two common excitatory links are *argument overlap* and *propositional embedding*. In CI the similarity of two nodes is captured intrinsically by a link. Whereas in Soar, explicit rules are required to indicate such similarity. A long-term store contains all knowledge the system is aware of (world, general task, and problem solving), and a short-term store holds all of the knowledge that is activated once a task description is encountered. Knowledge in the short-term store goes through a process of network *construction*, and then the resulting network is *integrated* and acted on by the decision procedure. A formal learning mechanism does not yet exist.

Like Soar, CI's activity revolves around a decision cycle, the result of which is to "fire" an action node. This firing causes action to be taken, effectively changing the "problem solver's current conception of the world". For instance, propositional nodes indicating a state change may be added to this "world" from visual processes. A CI decision cycle consists of the following. First, the contents of the short-term store are *constructed* in the form of an associative network: all old and new nodes have their links established in light of the new information (gained from the previous decision cycle or an encounter with the task description). Next, the new network is activated using an activation source such as a specific task description, goal, or request. The network is then *integrated* using a spreading activation technique, until a stable state is reached. At this point the decision procedure determines which node is most active and fires it, once again changing the state of the "world". A new cycle then begins with the reconstruction of a new network.

3. The Task: Pressing a Button

The task modelled is that of pressing a button on a Automatic Teller Machine (ATM). The button pressing is in response to a first time user's goal to *"determine the balance of his checking account."* The particular button press addressed here, is the one which indicates that the transaction selected is "--checking balance", as prompted for on the display screen in Figure 1.

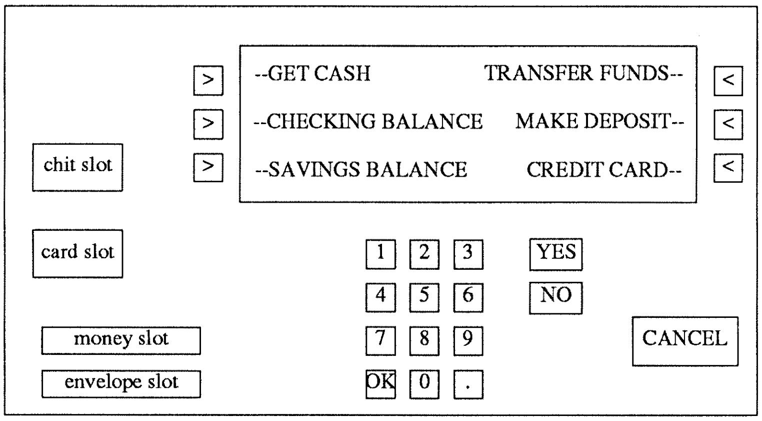

Figure 1: ATM Interface.

The knowledge the user is assumed to have is not overly rich. First, it is assumed that he understands standard banking terminology (e.g. checking, savings, account balance, etc.). Second, it is assumed that he knows that buttons afford pressing and how to press a button. But, it is important to note that when the user comes up the ATM for the first time, he does *not* know exactly which physical actions will be required or what specific choices the machine will offer. For instance, with regard to physical actions, he does not know whether he will need to press buttons, use a touch screen, or use a keyboard.

But, why bother with such a seemingly trivial task? The interest centers on the knowledge the user is assumed *not* to have. Since the user does not know before confronting the machine what kind of action to take, or even what choices will be offered, the appropriate behavior must be organized in response to cues presented by the machine. This is in contrast to having the behavior being determined only on the basis of prior knowledge. The dynamic organization required by this trivial task challenges existing cognitive models in HCI, such as GOMS [CMN83] and Cognitive Complexity Theory [KiP85], in which the skilled users being modelled are assumed to possess detailed knowledge of the actions appropriate to given conditions. In the same way, this task challenges cognitive architectures to provide more than a framework which a human analyst can use to program desired performance.

3.1. A Soar Model for the Task

As of 1987, many tasks performed by Soar were AI-related, e.g. blocks world, eight puzzle, monkey and bananas, and R1-Soar [LNR87]. A few tasks were more psychologically oriented, e.g. syllogistic reasoning, and task transfer. Since then, Soar has been used to evaluate more complex tasks, e.g. natural language understanding with NL-Soar [New90]. Recent work has also been done in the domain of HCI, as exemplified by Browser-Soar [JNC90], and work on immediate reasoning can be seen in IR-Soar [PNL89], which is also used in a Soar Theory of Taking Instructions [LNP89].

Given our assumptions about the user's background knowledge, we need a model that allows us to extract information in a new situation, which can further be used to organize Soar to do a task. Consequently, our model is based on both IR-Soar and the Instruction Taking Soar, mentioned above. The focus of the model is on how the user discovers which arrow button needs to be pressed to select the transaction "--checking balance". Thus, the user must comprehend the display screen's instructions along with the interface (the choice labels and arrow buttons to the side of each), and then the user must select the proper choice and press the appropriate button.

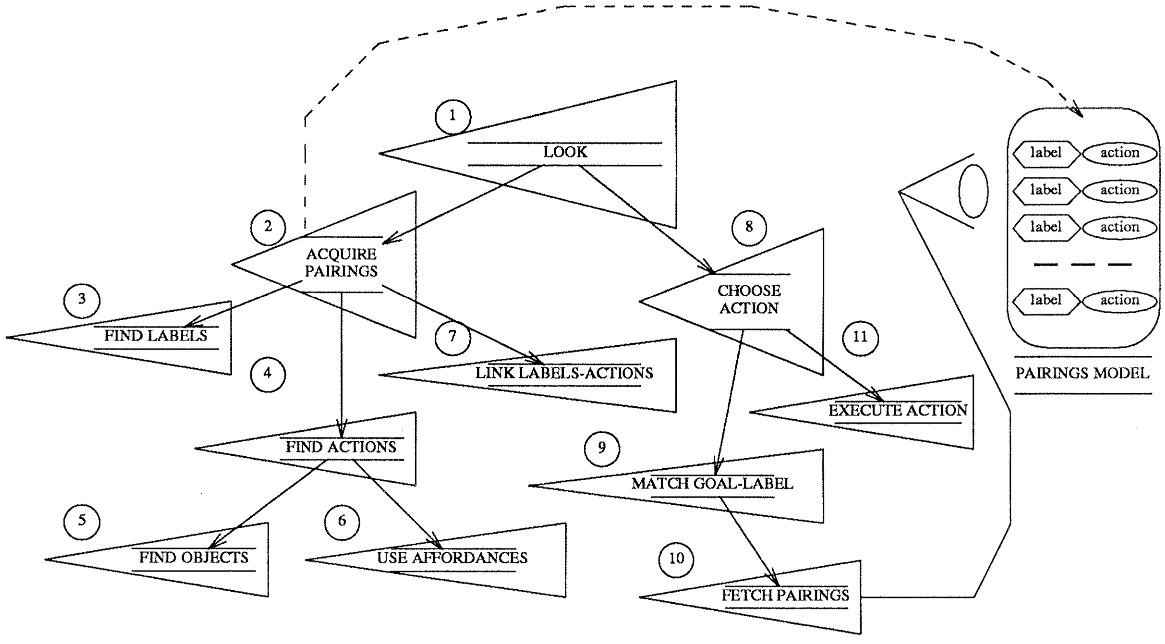

Figure 2: Behavior in the Soar Model

Figure 2 shows the behavior of the Soar system as it performs this button pressing task. A discussion of the figure follows, where each (#) corresponds to the space being visited. (1) For this model, the system begins in a *look* space. Here a series of matching operators are applied to each incoming visual stimulus. Each matching operator serves to acquire one label-action pairing, each of which is added to the *pairings model*. When all pairings have been made, then the user would need to do the task (choose and press one button). (2) The matching operator is implemented in an *acquire pairings* space (similar to the comprehend space used in IR-Soar). Here the pairings are built by firing operators which find labels, find actions, and then link them. (3) The label finding operators are implemented in the *find labels* space. These operators serve to produce label location information of the form (label at location). For example, we might have ("--checking balance" at left-arrow-2). (4) The action finding operators would be implemented in a *finding actions* space. To find actions we need to first find objects which will help us to attain our goal, then we need to combine our knowledge of affordances with the object. (5) To get objects, we apply the find objects operators, to be implemented in a *find objects* space. Each operator application serves to generate information similar to (button at location). For example, we might get (button-2 at left-arrow-2). (6) The affordance operators (to be defined by the *use affordances* space), would take our general knowledge of buttons, i.e. (affords button press), and combine it with the information about an object in (5). The result of the application would be something like (press button-2 at left-arrow-2). (7) Next, link operators would be used to link the labels and actions from (3) and (4). The result would be a label-action pairing, e.g. (label at location press button). For our example we would have ("--checking balance" at left-arrow-2 press button-2). Such label-action pairings are in effect the content of the pairings model (produced by the acquire pairings comprehension process). (Note, this pairings model only roughly corresponds to IR-Soar's behavioral model. Instead of the model being one of future behavior, it is a model that contains all possible actions the user can take. Both are interpretable structures. The space of possible actions is limited in size; the size defined by the number of pairings acquired during successive look operations.)

Once the label-action pairings are acquired (as indicated by the dashed line from (2) to the pairings model), Soar must choose and press one button. Thus, a pairing must be selected. The selection must be based on the user's goal to determine the balance of his checking account. (8) After all pairings have been placed into the model, Soar would not know what action to take and consequently drop into the *choose action* space. However, in order to choose an action, the goal must be matched to a label-action pairing, and once matched, the button must be pressed. To do these tasks requires the use of a match goal to label operator and an execute action operator. (9) The match goal to label operator is defined by the *match goal-label* space. To get this match, Soar can use a form of browsing behavior; employ directive looking. In this case, the model takes on a solution akin to Browser-Soar, where the user's search for and finding of a particular pairing's label would be accomplished by applying search and evaluation criteria operators. (These may need to be defined by additional problem spaces). However, to have directive looking (and to press the button), the label-action pairings knowledge must be in recognition memory. To resolve this impasse, the pairings model must be consulted. (10) The *fetch pairings* space serves to locate a pairing in the pairings model and interpret it as a possible action. The directive looking behavior ceases when a pairing matches the current goal. (11) Finally after the match has occurred, the *execute action* operator does the specified action to the particular object interpreted from the pairing information obtained by (9)-(10). In this case, Soar would press button-2 at location left-arrow-2, assuming the goal to determine the checking account balance was matched to the label "--checking balance".

3.2. A CI Model for the Task

As for CI, less diverse tasks have been investigated. Much work has revolved around discourse comprehension tasks such as word sense disambiguation, understanding of arithmetic word problems [Kin88], and memory for sentence recognition [Wel89]. Some HCI related work has also been done: this on electronic mail system tasks [Man89], [1]UNIX is a trademark of AT&T Bell Laboratories. and phone answering systems [LPW90].

The CI model discussed here, is based on the HCI work mentioned above, particularly the phone answering systems work. The focus of this model, too, is on how the user makes the "connection" between the labels, buttons, and actions, selecting and pressing only the correct one. Unlike Soar, all of the knowledge required for CI does not have to be as explicitly stated. Instead, in two places, the model relies on the natural, associational machinery of CI: the first is in connecting the labels and the actions, the second is in matching the goal to a given label so that an action can be chosen. A simplified network diagram given in Figure 3, demonstrates how the model makes use of this machinery.

When the user encounters the ATM, it is assumed that he has the knowledge of affordances and buttons as mentioned earlier. He also has the goal of determining the balance in his checking account. This background knowledge (of actions, affordances, and goals) is represented in Figure 3 with three nodes: an action node (represented by the nested ovals), a goal node (at the top of the diagram), and an affordance node (near the bottom of the diagram).

[1]UNIX is a trademark of AT&T Bell Laboratories.

Whereas in the Soar model labels and actions were linked through a complicated pairing process, in CI these links are established automatically using the principle of *argument overlap* for nodes. When the user looks at the display screen, he generates the nodes within the boxed portions of Figure 3. These nodes indicate the location of a particular label, the location of some object, and that the object is a button. (Locations are indicated by ARROW-L# or ARROW-R#, where L stands for left, R for right, ARROW for the arrow buttons adjacent to the display screen, and # for the top-down ordering of these arrow buttons.) For each label and object the user looks at, one boxed structure is created. Thus, if the user looked only at the first two, leftmost labels on the screen ("--get cash" and "--checking balance") then the network in Figure 3 would be generated.

After this information is put into working memory by a "looking action", the argument overlap links are established. The arguments that overlap between nodes, are underlined. In effect these links are established by the property of visual contiguity. The links are indicated by the arrows in the diagram. When the network is integrated during the next decision cycle, the fact that the goal node is linked to only one boxed structure, will influence the decision reached. In this case, the network path from the goal node to the action node, will cause CI to press the button associated with the "--checking balance" option. The fact that the goal node is not linked to other boxed structures (associated with other choices), causes those ATM choices to be lessened in activation.

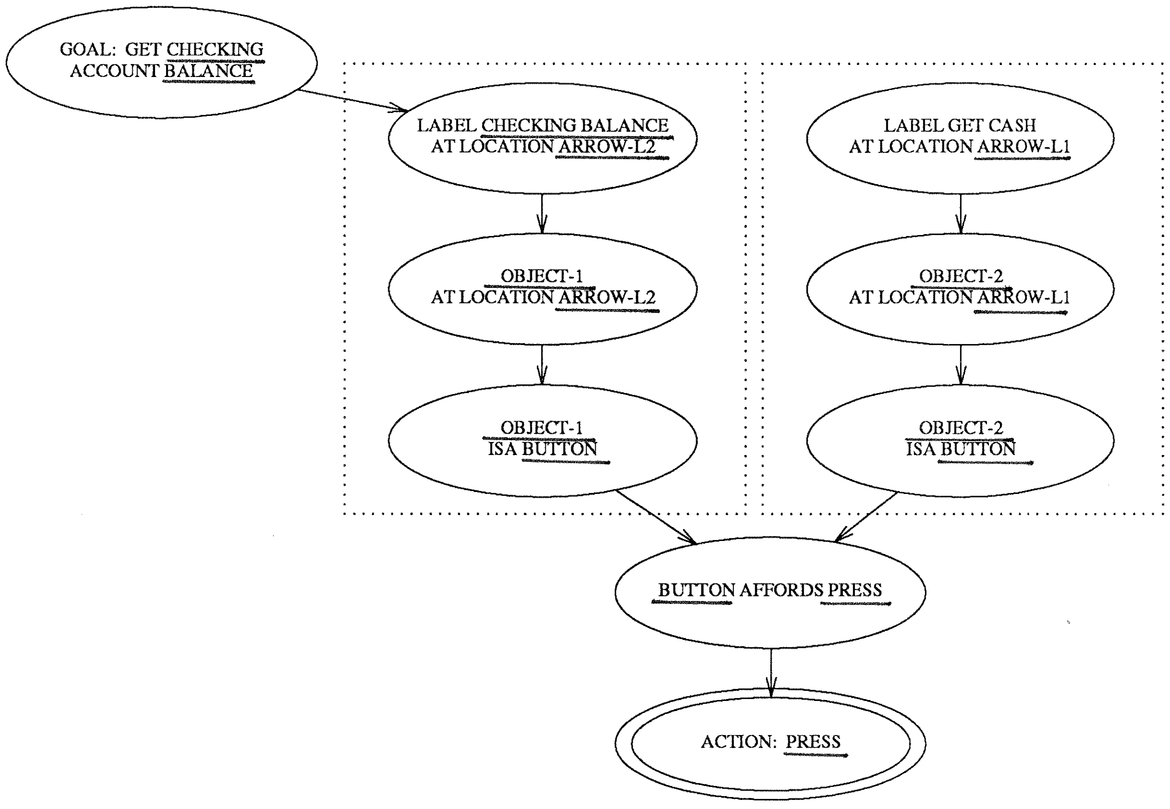
GOAL: GET CHECKING ACCOUNT BALANCE
LABEL CHECKING BALANCE AT LOCATION ARROW-L2
OBJECT-1 AT LOCATION ARROW-L2
OBJECT-1 ISA BUTTON
LABEL GET CASH AT LOCATION ARROW-L1
OBJECT-2 AT LOCATION ARROW-L1
OBJECT-2 ISA BUTTON
BUTTON AFFORDS PRESS
ACTION: PRESS

Figure 3: Behavior in the CI Model.

4. Comparing the Models

We now several address issues raised by the above models. First we give the issue, then the discussion.

• *High level similarities in processing: architectural and model concerns.*

Soar and CI are quite similar at a high level. Consider the architectural descriptions presented earlier. Both systems' behavior is based on a decision cycle that serves to introduce new knowledge into the world. In CI (and Soar), knowledge may be introduced through node (or production) firings, construction (or elaboration) phases, and mechanisms for knowledge integration (or interpretation of the preference semantics). Decision procedures are used to capitalize on the integrated knowledge by doing a specified action, e.g. action node firing (or state selection). When the models are viewed at a high level of processing, here too, similarities exist. For instance, both models allow for self organization (discovering of knowledge and actions) by extracting information from the environment. Both models allow for the acquisition of all label-action pairings as needed (or dictated by our looking). And finally, both models allow us to interpet the pairings in such a way that the models select the proper ATM option and press the appropriate button.

• *Low level differences in processing: tying labels to actions and choosing an action.*

In the Soar model, we were required to use explicit knowledge about how the labels and actions should be paired (linked). Similarly, explicit knowledge is required to ensure that the action should be done because somehow the goal matches the label. The CI model, however, all of this linking is "free". The model simply relies upon the associative mechanism built into the architecture. Thus, CI gives us links based on implicit semantic relationships. Soar in contrast requires specific rules to make the same links.

• *Task prestructuring requirements.*

Soar requires a number of appropriately interconnected problem spaces to exist and be recruited for the task. The resulting structure is not specific to the task, but it is not of general utility either. So how this structuring occurs must be accounted for. This appears to be an area of research within the Soar framework. CI in comparison, does not use a structure other than that induced on the task description and actions by general background knowledge and perception of the situation.

• *What do the differences mean?*

CI argues that use of cues in the interface, of the type that lead to associative connections, are of fundamental importance in interface design. These include similarity of position, as between label and button in our example, and similarity of wording between goal and label. In Soar the use of cues, if indicated at all, rests on convention and not architecture: Soar can be instructed to use them but is not architecturally constrained to use them. There is a testable contrast here: according to CI, but not Soar, cues like adjacency should be effective even for people with no experience with artifacts that embody them. The same analysis suggests that in Soar, conventions which contradict similarity cues should be readily accepted and used, whereas in CI such conventions would be expected to be very difficult to use.

5. Conclusions

As the discussion section indicates, a simple HCI task, can bring out both similarities and differences between two cognitive architectures. CI relies less on explicit knowledge than does Soar, and instead more on patterns of similarity within the environment to structure its behavior. For HCI, CI more than Soar, suggests an emphasis on environmental cues for interface design.

6. Acknowledgements

The authors thank Stephanie Doane, Bonnie John, Peter Polson, and John Rieman for helpful discussions.

7. References

[And83] J. R. Anderson, *The Architecture of Cognition*, Harvard University Press, Cambridge, MA, 1983.

[CMN83] S. K. Card, T. P. Moran and A. Newell, *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1983.

[DKP89] S. M. Doane, W. Kintsch and P. Polson, ''Action Planning: Producing UNIX Commands'', *Proceedings of Eleventh Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, August 16-19, 1989, 458-465. Conference held at University of Michigan, Ann Arbor, MI.

[Gre88] J. G. Greeno, ''Situations, Mental Models, and Generative Knowledge'', in *Complex Information Processing: The Impact of Herbert A. Simon*, D. Klahr and K. Kotovsky (editor), Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1988.

[JNC90] B. E. John, A. Newell and S. Card, ''Browser-Soar: A GOMS-like Model of a Highly Interactive Task'', Work in Progress, February 10-13, 1990. Talk given at Human-Computer Interaction Consortium, San Diego, CA.

[KiP85] D. E. Kieras and P. G. Polson, ''An Approach to the Formal Analysis of User Complexity'', *International Journal of Man-Machine Studies 22* (1985), 365-394.

[Kin88] W. Kintsch, ''The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model'', *Psychological Review 95*, 2 (1988), 163-182, American Psychological Association, Inc.

[LNR87] J. E. Laird, A. Newell and P. S. Rosenbloom, ''SOAR: An Architecture for General Intelligence'', *Artificial Intelligence 33*, 1 (1987), 1-64, Elsevier Science Publishers B.V. (North-Holland).

[LNP89] R. L. Lewis, A. Newell and T. A. Polk, ''Toward a Soar Theory of Taking Instructions for Immediate Reasoning Tasks'', *Proceedings of Eleventh Annual Conference of the Cognitive Science Society*, Hillsdale NJ, August 16-19, 1989, 514-521. Conference held at University of Michigan, Ann Arbor, MI.

[LPW90] C. Lewis, P. Polson and C. Wharton, ''Applying the Construction-Integration Model to Phone Answering Systems'', Work in Progress, University of Colorado at Boulder, Boulder, CO, 1990.

[Man89] S. M. Mannes, ''Problem-Solving as Text Comprehension: A Unitary Approach'', (Unpublished PhD Dissertation), University of Colorado at Boulder, Boulder, CO, 1989.

[NeS72] A. Newell and H. A. Simon, *Human Problem Solving*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972.

[New80] A. Newell, ''Reasoning, Problem Solving, and Decision Processes: The Problem Space as a Fundamental Category'', in *Attention and Performance VIII*, R. Nickerson (editor), Erlbaum, Hillsdale, NJ, 1980, 693-718. Chapter 35.

[NeC85] A. Newell and S. K. Card, ''The Prospects for Psychological Science in Human-Computer Interaction'', *Human-Computer Interaction 1* (1985), 209-242.

[New90] A. Newell, *The 1987 William James Lectures: Unified Theories of Cognition*, In press, 1990.

[Nor88] D. A. Norman, *The Psychology of Everyday Things*, Basic Books, Inc., Publishers, New York, 1988.

[PNL89] T. A. Polk, A. Newell and R. L. Lewis, ''Toward a Unified Theory of Immediate Reasoning in Soar'', *Proceedings of Eleventh Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, August 16-19, 1989, 506-513. Conference held at University of Michigan, Ann Arbor, MI.

[Suc87] L. Suchman, *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge, New York, 1987.

[Wel89] D. M. Welsch, ''A Connectionist Approach to Recognition Memory for Sentences'', (Unpublished Masters Thesis), University of Colorado at Boulder, Boulder, CO, 1989.