Accessibility versus Action-Centeredness in the Representation of Cognitive Skills

Ron Sun (rsun@cecs.missouri.edu)
Xi Zhang (xzf73@mizzou.edu)
Department of CECS, University of Missouri, Columbia, MO 65211, USA

Abstract

We believe that the distinction between procedural and declarative knowledge unnecessarily confounds two issues: action-centeredness and accessibility, and can be made clearer through separating the two aspects. The work presents an integrated model of skill learning that takes into account both implicit and explicit processes and both action-centered and non-action-centered knowledge. We examine and simulate human data in the Letter Counting task. The work shows how the data may be captured using either the action-centered knowledge alone or the combined action-centered and non-action-centered knowledge. The results provide a new perspective on skill learning.

Introduction

There have been many categories of knowledge being proposed in cognitive skill acquisition. Among them, one enduring (albeit somewhat controversial) distinction is the distinction between procedural and declarative knowledge (Anderson 1983, 1993). In Anderson (1983, 1993), procedural knowledge is represented in an action-centered way (with production rules that can only be used in one direction— from conditions to actions), and declarative knowledge in a non-action-centered way (i.e., with knowledge chunks that can be used in any possible direction).

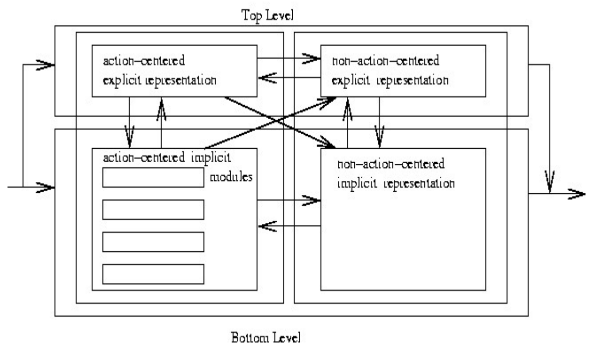
On the other hand, this distinction leads to another, equally intriguing distinction that has been hotly debated in the literature—the distinction between implicit and explicit knowledge (Reber 1989, Seger 1994, Destrebecqz and Cleeremans 2001). In Anderson (1983), declarative knowledge is assumed to be accessible: Subjects can easily access, manipulate, and report on such knowledge. Procedural knowledge is not: It leads to actions without much explicit subjective accessibility (Anderson 1993, Proctor and Dutta 1995, Sun et al 2001). Thus, in Anderson (1983), the two dichotomies are merged into one.

In ACT-R as described by Anderson and Lebiere (1998), however, each individual piece of knowledge, be it procedural or declarative, involves both implicit (subsymbolic) representation and explicit (symbolic) representation. Symbolic representation is used for denoting semantic labels and structural components of each concept, while subsymbolic representation is used for expressing its activation and other numeric measures. This view constitutes another perspective on the relation between the two dichotomies.

According to the first view above, the difference in action-centeredness seems the main factor in distinguishing the two types of knowledge, while accessibility a secondary factor (Anderson 1993). We believe that this view unnecessarily confounds two issues: action-centeredness and accessibility, and can be made clearer by separating the two issues. We have reason to believe that action-centeredness does not necessarily go with inaccessibility (see, e.g., Sun et al 2001), and non-action-centeredness does not necessarily go with accessibility either (see, e.g., Schacter 1987). Thus, we need to be aware of these two *separate* dimensions in theorizing about cognition.

The alternative view that each individual piece of knowledge, either procedural or declarative, involves both implicit and explicit processes is also problematic. The framework begs the question of the appropriateness of such close-coupling. The fact that each piece of knowledge has an explicit part contradicts the phenomenon that some knowledge may be completely implicit (Lewicki et al 1987, Destrebecqz and Cleeremans 2001). This contradiction raised the question of whether a tight coupling or a rather separate organization, for example, having these two types of knowledge in separate subsystems (Sun 2002, Sun et al 2001), makes better sense. At a minimum, more alternatives need to be explored.

In this paper, as an alternative to the afore-mentioned views, we propose the separation of the two dichotomies—we treat them as orthogonal to each other (Sun 2002). We separate procedural vs. declarative knowledge based on representation. Procedural knowledge is represented in an action-centered way: It goes in only one direction—from condition to action. Declarative knowledge is represented in a non-action-centered way: It can go in any direction. Each of them resides in a separate subsystem. In a similar fashion but separately, implicitness/explicitness can also be distinguished based on representation. Implicit knowledge can be represented using distributed representation, which is relatively inaccessible (Sun et al 2001, Sun 2002), whereas explicit knowledge can be represented using symbolic/localist representation, which is relatively accessible. Implicit and explicit knowledge thus reside in two different components, respectively. Moreover, in this way, the two



Top Level
action–centered explicit representation
non–action–centered explicit representation
action–centered implicit modules
non–action–centered implicit representation
Bottom Level

Figure 1: The CLARION architecture.

dichotomies are completely orthogonal to each other. That is, there are both implicit and explicit procedural knowledge, and both implicit and explicit declarative knowledge. In this work, we test this view using the task of letter counting (Rabinowitz and Goldberg 1995 and Johnson 1998).

In the remainder of this paper, we first introduce the idea of action-centered vs. non-action-centered representation through a model, in the next section. In the section following that, we describe some simulations of the letter counting task. Discussions and concluding remarks complete this paper.

The CLARION Model

CLARION is an integrative model with a dual representational structure. It consists of two main subsystems: the action-centered subsystem (ACS) and the non-action-centered subsystem (NACS). Each subsystem consists in turn of two levels: the top level encodes explicit knowledge and the bottom level encodes implicit knowledge. See Figure 1 for a sketch of the model.

The Action-Centered Subsystem

Action decision making in the ACS is as follows:

1. Observe the current state x.
2. Compute in the bottom level (the IDN, or Implicit Decision Network) the "value" of each of the possible actions (a_i's) in the current state x: $Q(x, a_1)$, $Q(x, a_2)$, ..., $Q(x, a_n)$.
3. Find out all the possible actions (b_1 , b_2 , ..., b_m) at the top level (the ARN, or Action Rule Network), based on the current state information x (which goes up from the bottom level) and the existing rules in place at the top level.
4. Choose an appropriate action a, by stochastically selecting the outcome of the top level or the bottom level.
5. Perform the selected action a, and observe the next state y.
6. Update the bottom level in accordance with Q-learning (implemented within a back-propagation network).
7. Update the top level using an appropriate learning algorithm (not relevant to this simulation).
8. Go back to Step 1.

In step 4, the selection probabilities are determined based on the relative performance of the two levels. That is, the two levels compete against each other.

The learning of implicit action-centered knowledge at the bottom level can be done in a variety of ways consistent with the nature of distributed representations. In general, reinforcement learning can be used, especially Q-learning implemented in backpropagation networks. See Sun et al (2001) for details and cognitive justifications.

Action-centered explicit knowledge at the top level is encoded in the form of action rules. It can also be learned in a variety of ways or be given externally (Sun 2002).

Rule utility measures the effectiveness of an action rule in terms of cost and benefit:

$$U_j^r = benefit_j - v * \cos t_j \qquad (1)$$

where v is a parameter that balances the scales of *cost* and *benefit*. The benefit of a rule is calculated based on the positive match ratio— how many positive matches a rule produces within the context of all the possible matches by the rule (a positive match criterion needs to be set). The cost of a rule is set based on the execution time considerations. The utility of a rule is the comparison of its benefit and its cost, as commonly done.

The output from the top level is determined through a competition of all the rules matching the current input, the same way as the competition of the two levels. The competition among rules is based on utility U_j^r .

The base-level activation (BLA) of an action rule is

$$B_j^r = c * \sum_{l=1}^{n} t_l^{-d} \qquad (2)$$

where t_l is the lth recent encoding/use of the rule, and the default values of the parameters are $c = 2$, $d = 0.5$. This quantity specifies the odds of needing a particular rule based on the history of a rule (Anderson 1993), which decays gradually following a power law. It represents priming effects on a rule resulting from prior uses of that rule.

The Non-Action-Centered Subsystem

At the bottom level of the non-action-centered subsystem, an "associative memory" network (AMN for short) encodes non-action-centered implicit knowledge. Since our experiments in this work do not involve this component, we will not get into the details.

At the top level of the non-action-centered subsystem, a general knowledge network (GKN) encodes explicit, non-action-centered knowledge (cf. Sun 1995). In this network, chunks are encoded through specifying attribute values. Links between chunks encode associations between pairs of chunks.

The basic form of a chunk is as follows: *chunk-id$_i$* : $(dim_{i_1}, val_{i_1})(dim_{i_2}, val_{i_2})\ldots\ldots (dim_{i_n}, val_{i_n})$ where *dim* denotes a particular state/output dimension, and *val* specifies its corresponding value. A node is set up in the GKN to represent a chunk. The node connects to its corresponding features in the bottom level (see Sun 1995 for further details).

Each chunk has a base-level activation (BLA), similar to what was described earlier regarding rules. It represents priming effects on a chunk resulting from prior uses of that chunk.

Response Times

The total response time by an agent is the sum of its decision time, perceptual time, and actuation (motor) time:

$$RT_{BL} = PT_{BL} + DT_{BL} + AT_{BL}$$
$$RT_{TL} = PT_{TL} + DT_{TL} + AT_{TL}$$

where *RT* is the total response time, *PT* is the perceptual time, *AT* is the motor action time, and *DT* is the decision time. RTs are determined separately for each level. Whichever level is selected decides the overall RT (which is either RT_{BL} or RT_{TL}). Normally, the RT of the bottom level is shorter than that of the top level. RT is partially determined by the history of use of rules and chunks (priming as represented by BLAs). See Sun (2002) for detailed justifications of these calculations.

Experiments

The Letter Counting Task

The setting of the letter counting task was as follows (Rabinowitz and Goldberg 1995, Johnson 1998): Subjects were asked to solve alphabetic arithmetic problems of the forms: *letter1* + *number* = *letter2* or *letter1* - *number* = *letter2*, where *letter2* is *number* positions up or down from *letter1*, depending on whether + or - was used. They were given *letter1* and *number*, and asked to produce *letter2*.

In experiment 1, during the training phase, one group of subjects (the consistent group) received 36 blocks of training, in which each block consisted of the same 12 addition problems. Another group (the varied group) received 6 blocks of training, in which each block consisted of the same 72 addition problems. While both groups received 432 trials, the consistent group practiced on each problem 36 times, but the varied group only 6 times. The addends ranged from 1 to 6. The consistent group experienced 2 occurrences of each addend, while the varied group experienced 12 occurrences of each addend. In the transfer phase, each group received 12 new addition problems, repeated 3 times.

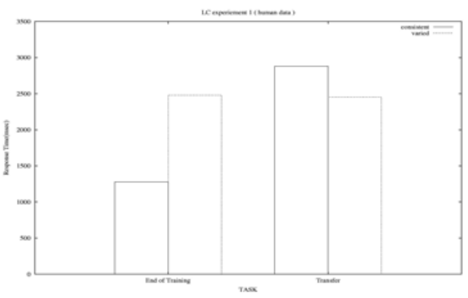


Figure 2: Experiment 1 of the letter counting task.

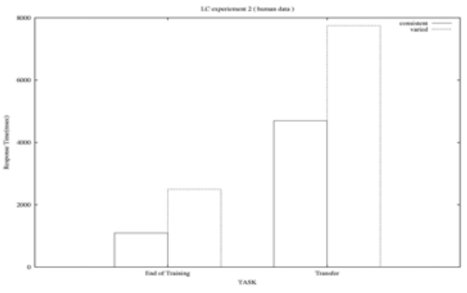


Figure 3: Experiment 2 of the letter counting task.

The findings were that, at the end of training, the consistent group performed far better than the varied group. However, during transfer, the consistent group performed worse than the varied group. The varied group showed perfect transfer, while the consistent group showed considerable slow-down. See Figures 2. (Note that only correct responses were used in the analysis; the same below.)

In experiment 2, the training phase was identical to that of experiment 1. However, during the transfer phase, both groups received 12 subtraction (not addition) problems, which were the reverse of the original addition problems, repeated 3 times. The findings were that, in contrast to experiment 1, during transfer, the consistent group actually performed better than the varied group. Both groups performed worse than their corresponding performance at the end of training, but the varied group showed worse performance than the consistent group. See Figure 3.

Simulation 1

Model Setup. The simulation was based on "top-down" learning: that is, we encoded a set of a priori rules for capturing prior knowledge concerning counting letters at the top level of the ACS. Then, on the basis of these rules, performance was carried out and implicit learning at the bottom level of the ACS took place. The set of rules included the following:

If goal=addition-counting, start-letter=*x*, number=*n*, then starting with *x*, repeat *n* times: count-up

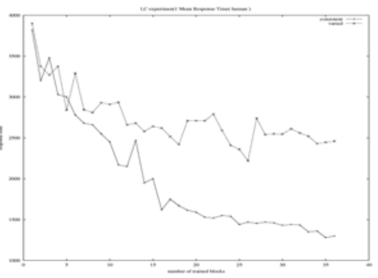


Figure 4: The learning curves of the letter counting task.

If goal=subtraction-counting, start-letter=*x*, number=*n*, then starting with *x*, repeat *n* times: count-down

If goal=addition-counting, start-letter=*x*, number=*n*, then retrieve chunks with (*dim*1 = *x*, *dim*2 = +, *dim*3 = *n*, *dim*4 =?) and report *dim*4.

If goal=addition-counting, start-letter=*x*, number=*n*, then retrieve chunks with (*dim*1 = ?, *dim*2 = -, *dim*3 = *n*, *dim*4 = *x*) and report *dim*1 (retrieve and reverse).

If goal=subtraction-counting, start-letter=*x*, number=*n*, then retrieve chunks with (*dim*1 = *x*, *dim*2 = -, *dim*3 = *n*, *dim*4 =?) and report *dim*4.

If goal=subtraction-counting, start-letter=*x*, number=*n*, then retrieve chunks with (*dim*1 =?, *dim*2 = +, *dim*3 = *n*, *dim*4 = *x*) and report *dim*1 (retrieve and reverse).

In these rules, "?" represented "don't care" conditions.

Rules competed based on their utility. In calculating the utility (i.e., the cost and benefit of each rule), the benefit was set equal to the positive match ratio of a rule (the default function for benefit). The rule condition must match the current state to be counted as a "match". A "positive match" was further determined by the outcome of the matching rule being correct.

Three inputs were provided: starting letter, arithmetic operator, and number. In addition, the goal from the goal structure was also input to both levels. There were 26 possible output values.

One IDN network was involved. Its output indicated the (guessed) target letter. There were 35 input units (26 letters + 6 numbers + 2 signs + 1 goal), 30 hidden units, and 26 output units.

The NACS was used for storing and retrieving experienced instances (in the form of chunks). In the NACS, each question and answer pair encountered was encoded as a chunk.

A retrieval rule from the ACS triggered all the chunks that overlapped with the retrieval cue (as indicated by the retrieval rule). Actual retrieval was limited to one chunk at each step. All triggered chunks (those partially or fully matching the retrieval cue) competed to be the one retrieved, based on chunk strengths.

The response time for chunk retrieval was determined by both the BLA of the chunk retrieved (in the NACS) and the BLA of the retrieval action rule applied (in the ACS). The total response time of chunk retrieval was the sum of the perceptual time (PT_{TL}), the decision time of the retrieval rule (in part determined by the BLA of the action rule: $t_2 + t_0 / BLA(rule)$), the retrieval time in the NACS (in part determined by the BLA of the chunk retrieved ($t_1 / BLA(chunk)$), and the verbal answer time.

Simulation Results. First of all, during the training phase of experiment 1, the simulation matched the response time difference between the consistent and the varied group. See the simulation data in Figure 5, which is to be compared with Figure 2. The simulated consistent group had a lower response time because it had more practice on a smaller number of instances, which led to the better performing IDN (the bottom level in the ACS), as well as better performing instance retrieval from the NACS. Because they were better performing, the IDN and the NACS were more likely to be used in determining the overall outcome of the simulated consistent group, due to the competition among different components. Because these two components had lower response times than other components, [1] a lower overall response time resulted for the simulated consistent group. The difference was statistically significant ($F(98) = 291.191$, $p < 0.0001$).

CLARION also matched the transfer performance difference between the two groups in experiment 1, as shown in Figure 5. During the transfer phase of experiment 1, the performance of the simulated consistent group was worsened, compared with its performance at the end of training ($F(98) = 537.010$, $p < 0.0001$); the transfer performance of the simulated consistent group was in fact worse than that of the simulated varied group ($F(98) = 41.672$, $p < 0.0001$). This is because the simulated consistent group relied more on the IDN and the NACS during training and therefore, the BLAs of its counting rules were lower. As a result, it took more time to apply the counting rules during transfer, which it had to apply, due to the fact that it had to deal with a different set of instances during transfer. (Note that this explanation was not offered by the ACT-R simulation; see Johnson 1998) The performance of the simulated varied group hardly changed, compared with its performance at the end of training ($F(98) = 2.534$, $p = 0.1147$), because it relied mostly on the counting rules at the top level during training, which was equally applicable to both training and transfer. As a result, its counting rules had higher BLAs, and therefore it performed better than the simulated consistent group during transfer. The difference was statistically significant ($F(98) = 41.672$, $p < 0.0001$).

[1] It is either inherently so, as in the case of the IDN, or due to frequent use (higher BLAs), as in the case of the NACS.

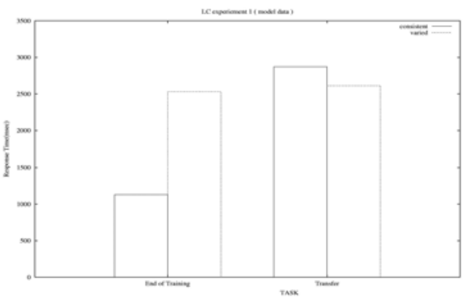


Figure 5: Simulation 1 of experiment 1 of the letter counting task.

As indicated by Figure 6, which is to be compared to Figure 3, this simulation also captured accurately the human data of experiment 2. During transfer in experiment 2, due to the change in the task setting (counting down as opposed to counting up), the practiced rule for counting up was no longer useful. Therefore, both simulated groups had to use a new counting rule (for counting down), which had only the initial BLA in both cases. Similarly, both simulated groups had to use a new instance retrieval rule (for reverse retrieval), which also had only the initial BLA in both cases. Both simulated groups performed worse than at the end of training for the above reason ($F(98) = 56.305$, $p < 0.0001$ and $F(98) = 1383.894$, $p < 0.0001$ for the simulated consistent and the simulated varied group respectively). (Note that this explanation was not offered by the ACT-R simulation, see Johnson 1998.)

Moreover, this simulation captured the fact that the varied group performed worse than the consistent group during transfer (Figure 6). This difference was explained by the fact that the simulated consistent group had more BLAs associated with chunks than the simulated varied group, because the simulated consistent group had more practice with these chunks. These chunks were used in "reverse retrieval" during the transfer phase of experiment 2, because of the reverse relationship between the training and the transfer instances used in this experiment. Therefore, the simulated consistent group performed better than the simulated varied group in this phase ($F(98) = 41.121$, $p < 0.0001$).

The learning curves of the training phase in this simulation are shown in Figure 7. They should be compared to Figure 4.

Simulation 2

Model Setup. In this alternative simulation, only the ACS was used. Therefore, all the rules and other mechanisms related to the NACS were removed.

Simulation Results. The simulation captured the response time difference of the training phase of experiment 1. In the simulation of human performance of the training phase (Figure 2), the simulated consistent

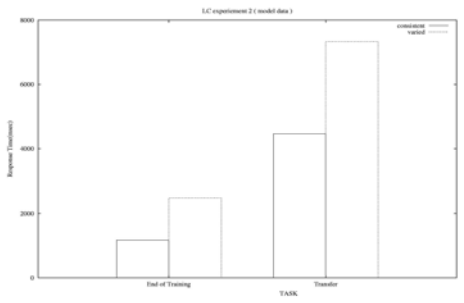


Figure 6: Simulation 1 of experiment 2 of the letter counting task.

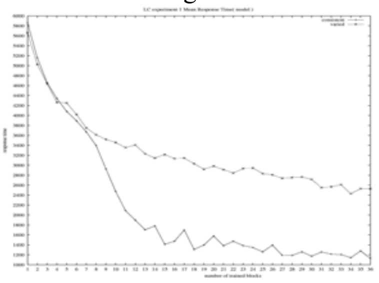


Figure 7: The learning curves of simulation 1 of the letter counting task.

group had a lower response time (Figure 8), the same as in the human data. The difference between the two simulated groups was statistically significant ($F(98) = 61.056$, $p < 0.0001$).

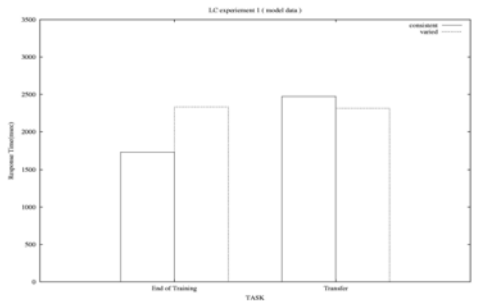


Figure 8: Simulation 2 of experiment 1 of the letter counting task.

CLARION also matched the transfer performance difference of experiment 1. As in the human data (shown in Figure 2), during the transfer phase of experiment 1, the performance of the simulated consistent group was worsened compared with its performance at the end of training ($F(98) = 103.781$, $p < 0.0001$), and in fact it was worse than that of the simulated varied group ($F(98) = 35.452$, $p < 0.0001$).

However, in experiment 2, although the transfer performance of both simulated groups was worse compared with their respective performance at the end

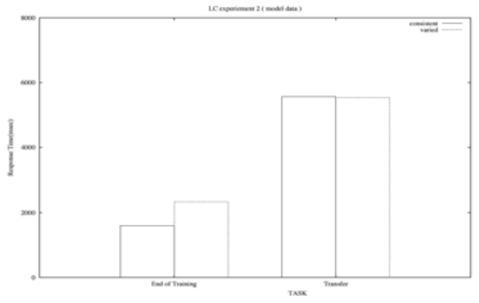


Figure 9: Simulation 2 of experiment 2.

of training ($F(98) = 2781.077$, $p < 0:0001$ and $F(98) = 5522.671$, $p < 0.0001$ for the simulated consistent and the simulated varied group respectively), the simulation failed to explain the fact that the varied group performed worse than the consistent group in transfer, due to the lack of the instance retrieval mechanism as in the previous simulation.

Discussions

CLARION provides some interesting interpretations of the human data. For example, it attributes the performance difference at the end of training between the consistent and the varied group to the difference between relying on implicit knowledge and relying on explicit rules. Beside incorporating some ACT-R interpretations of this task, CLARION goes beyond existing ACT-R (and other) simulations in providing interpretations that other models do not provide. The CLARION simulations are far more accurate than the ACT-R simulations (see Johnson 1998).

This good match between the simulation and the human data was obtained under the same set of parameters for all the groups and all the conditions involved. The only difference was that of stimuli, which did not require any change in model parameters for simulating different groups. Considering the fact that there were a total of three different conditions (training and transfer in experiment 1, and transfer in experiment 2), with two groups in each, it was not a trivial matter to obtain a good match using only one set of parameters. The match shows, to some extent, the cognitive validity of CLARION.

Comparing the two simulations, we see that, although the ACS alone could capture the data in this task to a certain extent, the use of both the ACS and the NACS led to better capturing of data. Thus, to some extent, the simulation of the letter counting task indicates the need of having both the ACS and the NACS.

Concluding Remarks

This work shows that it is possible and useful to separate the two dichotomies: implicit vs. explicit knowledge and action-centered vs. non-action-centered knowledge. We illustrate this separation through a cognitive architecture CLARION, which succeeded previously in simulating a variety of cognitive data (see Sun et al 2001, Sun 2002). This separation leads to new possibilities of interpreting data and new ways of understanding cognitive skill acquisition.

Acknowledgements

This work is supported in part by Army Research Institute contract DASW01-00-K-0012.

References

J. R. Anderson, (1993). *Rules of the Mind.* Lawrence Erlbaum Associates, Hillsdale, NJ.

J. Anderson and C. Lebiere, (1998). *The Atomic Components of Thought*, Lawrence Erlbaum Associates, Mahwah, NJ.

A. Destrebecqz and A. Cleeremans, (2001). Can sequence learning be implicit?, New evidence with the process dissociation procedure. *Psychonomic Bulletin and Review*, 8, 2, 343-350.

T. Johnson, (1998). Acquisition and transfer of declarative and procedural knowledge. *European Conference on Cognitive Modeling,* pp.15-22. Nottingham University Press, Nottingham, UK.

P. Lewicki, M. Czyzewska, and H. Hoffman, (1987). Unconscious acquisition of complex procedural knowledge. *Journal of Experimental Psychology: Learning, Memory and Cognition*. 13 (4), 523-530.

R. Proctor and A. Dutta, (1995). *Skill Acquisition and Human Performance*. Sage Publications, Thousand Oks, CA.

M. Rabinowitz and N. Goldberg, (1995). Evaluating the structure-process hypothesis. In: F. Weinert and W. Schneider, (eds.) *Memory Performance and Competencies*. Lawrence Erlbaum, Hillsdale, NJ.

A. Reber, (1989). Implicit learning and tacit knowledge. *Journal of Experimental Psychology: General*. 118 (3), 219-235.

D. Schacter, (1987). Implicit memory: History and current status. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13, 501-518.

C. Seger, (1994). Implicit learning. *Psychological Bulletin*. 115 (2), 163-196.

R. Sun, (1995). Robust reasoning: integrating rule-based and similarity-based reasoning. *Artificial Intelligence*. 75, 2. 241-296.

R. Sun, (2002). *Duality of the Mind*. Lawrence Erlbaum Associates, Mahwah,NJ.

R. Sun, E. Merrill, and T. Peterson, (2001). From implicit skills to explicit knowledge: a bottom-up model of skill learning. *Cognitive Science*, 2001.