Top-Down versus Bottom-Up Learning in Cognitive Skill Acquisition

Ron Sun
Department of Cognitive Science
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
EMAIL: rsun@rpi.edu

Xi Zhang
Department of CECS
University of Missouri, Columbia, MO 65211

November 12, 2003

ACKNOWLEDGMENT: This work is supported in part by Army Research Institute contract DASW01-00-K-0012. Thanks to Robert Mathews for various discussions. Thanks also to Vasant Honavar and the referees for their helpful comments.

Abstract

This paper explores the interaction between implicit and explicit processes during skill learning, in terms of top-down learning (that is, learning that goes from explicit to implicit knowledge) versus bottom-up learning (that is, learning that goes from implicit to explicit knowledge). Instead of studying each type of knowledge (implicit or explicit) in isolation, we stress the interaction between the two types, especially in terms of one type giving rise to the other, and its effects on learning. The work presents an integrated model of skill learning that takes into account both implicit and explicit processes and both top-down and bottom-up learning. We examine and simulate human data in the Tower of Hanoi task. The paper shows how the quantitative data in this task may be captured using either top-down or bottom-up approaches, although top-down learning is a more apt explanation of the human data currently available. These results illustrate the two different directions of learning (top-down versus bottom-up), and thereby provide a new perspective on skill learning.

1 Introduction

This paper studies the interaction between implicit and explicit processes in skill learning. Specifically, it explores two directions of skill learning: top-down and bottom-up learning. As defined in Sun et al (2001), top-down learning goes from explicit to implicit knowledge, while bottom-up learning goes from implicit to explicit knowledge. Instead of studying each type of knowledge (implicit or explicit) in isolation, In the present work, we want to explore the interaction between the two types of learning, especially in terms of one type giving rise to the other (bottom-up or top-down learning).

In this work, we test possibilities of bottom-up versus top-down learning by using Tower of Hanoi, which is arguably a benchmark problem in high-level cognitive skill acquisition and has been used in many previous studies of skill acquisition, cognitive modeling, and cognitive architectures (see, e.g., Simon 1975, Anzai and Simon 1979, Zhang and Norman 1994, VanLehn 1995, Anderson 1993, Anderson and Lebiere 1998, Fum and Missier 2001, Altmann and Trafton 2002). The simulation of this task expands the scope of the CLARION model, from the original low-level skill domains (e.g., Sun et al 2001) to high-level cognitive skill tasks. Therefore, the choice of this task is justified.

To explore bottom-up and top-down learning, the paper presents an integrated model of skill learning that takes into account both implicit and explicit processes and on top of that, both top-down and bottom-up learning, although the model was initially designed as a purely bottom-up learning model. We examine and simulate human data in the Tower of Hanoi task. The work shows how the quantitative data in this task may be captured using either top-down or bottom-up approaches, although we show that top-down learning is a more apt explanation of some of the human data currently available in this task.

To summarize our results briefly, we achieved rather successful simulations in both cases—top-down and bottom-up learning, but to different degrees. Both our bottom-up and top-down models successfully captured the data concerning numbers of moves. Moreover, a top-down model was also successful in capturing some response time (RT) data. [1] Furthermore, simulation using only the bottom level (implicit learning) was not successful, suggesting that implicit learning alone was inadequate for this task.

Overall, the results of our simulation suggest that both directions are possible in human cognitive skill acquisition, and the actual direction may be either bottom-up or top-down (or a mix of the two),

[1] However, this may be due to the fact that such data were collected under the condition that facilitated top-down learning. See the discussion section for more discussions of this point.

depending on task settings, instructions, and other variables. These results provide a new perspective on skill learning.

In the remainder of this paper, we first introduce the ideas of top-down versus bottom-up learning through a theoretical model. Then, in section 3, we explicate some details concerning the computational implementation of the model. In section 4, we describe a sequence of simulations that contrast top-down versus bottom-up learning in a variety of ways. Discussions in section 5 complete this paper.

2 Top-Down versus Bottom-Up: The CLARION Model

The role of implicit learning in skill acquisition and the distinction between implicit and explicit learning have been widely recognized in recent years (see, e.g., Reber 1989, Stanley et al 1989, Willingham et al 1989, Anderson 1993, Regehr and Brooks 1993, Seger 1994, Proctor and Dutta 1995, Cleeremans et al 1998, Stadler and Frensch 1998, Aizenstein et al 2000). However, although implicit learning has been actively investigated, complex, multifaceted interaction between the implicit and the explicit (and the importance of this interaction) have not been universally recognized. To a large extent, such interaction has been downplayed or ignored, with only a few notable exceptions (e.g., Mathews et al 1989, Sun et al 1998, 2001). Research has been focused on showing the *lack* of explicit learning in various learning settings (see especially Lewicki et al 1987) and on the controversies stemming from such claims.

Despite the lack of studies of interaction, it has been gaining recognition that it is difficult, if not impossible, to find a situation in which only one type of learning is engaged (Reber 1989, Seger 1994, Sun et al 2001, but see Lewicki et al 1987). Our review of existing data has indicated that, while one can manipulate conditions to emphasize one or the other type, in most situations, both types of learning are involved, with varying amounts of contributions from each type (see, e.g., Sun et al 2001; see also Stanley et al 1989, Willingham et al 1989). Reber (1989) pointed out that nearly all complex skills in the real world (as opposed to controlled laboratory settings) involved a mixture of explicit and implicit processes interacting in some ways; the relations between the two might be complex. Even the performance in commonly used implicit learning tasks may involve a combination of explicit and implicit learning processes (Sun et al 2001).

Empirical demonstrations of interaction can be found in Stanley et al (1989), Willingham et al (1989), Bowers et al (1990), Wisniewski and Medin (1994), and Sun et al (2001). See also Karmiloff-

Smith (1986) and Mandler (1992). These demonstrations involved a variety of means, including experimental manipulations such as verbalization, explicit instructions, and dual tasks.

Likewise, in the development of cognitive architectures (e.g., Newell 1990, Rosenbloom et al 1993, Anderson 1983, 1993, Meyer and Kieras 1997, Anderson and Lebiere 1998), the distinction between procedural and declarative knowledge has been proposed for a long time, and adopted by many in the field (see Anderson 1983, 1993 in particular). The distinction maps roughly onto the distinction between explicit and implicit knowledge, because procedural knowledge is generally inaccessible while declarative knowledge is generally accessible and thus explicit. However, in work on cognitive architectures, focus has been almost exclusively on top-down models, the bottom-up direction has been largely ignored, paralleling and reflecting the related neglect of the interaction of explicit and implicit processes in the skill learning literature.

However, there are a few pieces of work that did demonstrate the parallel development of the two types of knowledge or the extraction of explicit knowledge from implicit knowledge (e.g, Rabinowitz and Goldberg 1995, Owen and Sweller 1985, Willingham et al 1989, Stanley et al 1989, Aizenstein et al 2000; see also Karmiloff-Smith 1986, Mandler 1992), contrary to usual top-down approaches in developing cognitive architectures.

Given the evidence of the complex interaction between implicit and explicit processes, the question is how we account for the interaction computationally. To better understand this interaction, we need to look into a number of issues related to the interaction:

- How can we capture implicit and explicit processes in one unified computational model?
- In such a model, how do the two types of knowledge develop along side each other via top-down and bottom-up learning?
- In such a model, how do the two types of knowledge interact during skilled performance?

To tackle these issues, the model CLARION was developed (Sun 1997, 1999, Sun et al 1998, 2001). CLARION is an integrative model with a dual representational structure. It consists of two levels: the top level encodes explicit knowledge and the bottom level encodes implicit knowledge. See Figure 1. There are also auxiliary components: working memory and goal stack (not shown in the figure).

First, the inaccessible nature of implicit knowledge is captured by subsymbolic distributed representations provided by a backpropagation network (Rumelhart et al 1986). This is because represen-

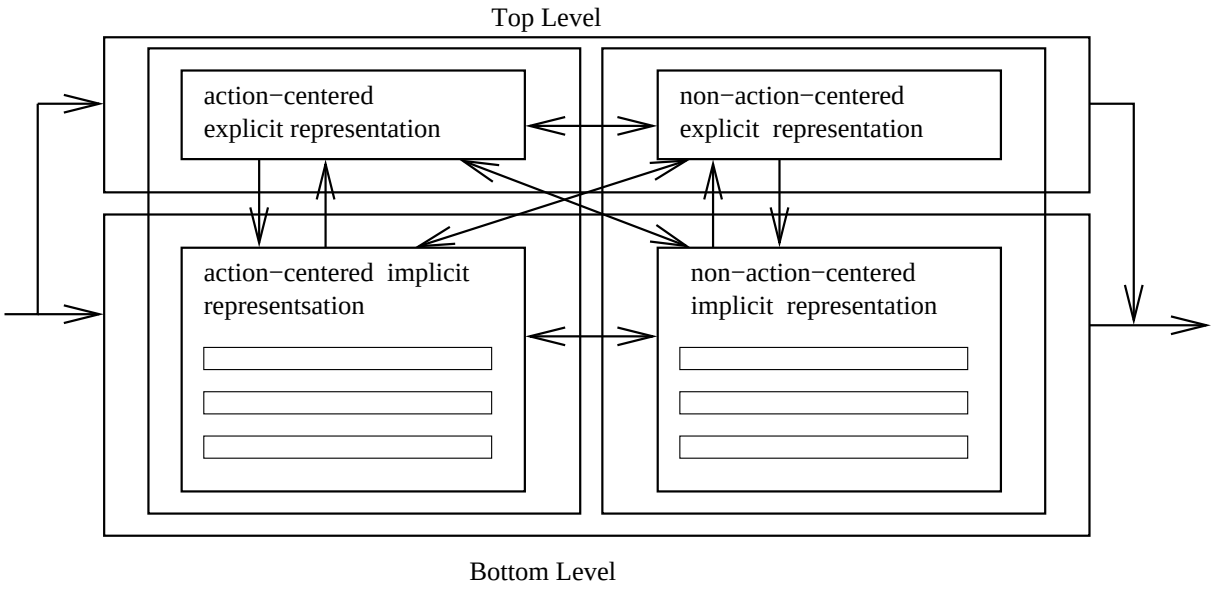

Figure 1: The CLARION architecture.

tational units in a distributed representation are capable of accomplishing tasks but are subsymbolic and generally not individually meaningful (see Rumelhart et al 1986, Sun 1994). That is, they are relatively inaccessible. This characteristic of distributed representation accords with the relative inaccessibility of implicit knowledge. (See Sun 2002 for detailed discussions of this point. Given the length limitation, we will not get into a philosophical argumentation in support of this representational account of implicitness here.)

In contrast, explicit knowledge may be captured in computational modeling by a symbolic or localist representation (Clark and Karmiloff-Smith 1993), in which each unit is easily interpretable and has a clear conceptual meaning. Thus, they are relatively more accessible than distributed representation. This characteristic captures the property of explicit knowledge being more accessible and manipulable (Smolensky 1988, Sun 1994).

At each level of the model, there may be multiple modules, both *action-centered* modules and *non-action-centered* modules (Schacter 1990, Moscovitch and Umilta 1991). We will focus only on action-centered modules in this work.

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 the learning settings where correct input/output mappings are available, straight backpropagation (a supervised learning algorithm) can be used for each network (Rumelhart et al 1986). Such supervised learning procedures require the

a priori determination of a uniquely correct output for each input. In the learning settings where there is no input/output mapping externally provided, reinforcement learning can be used, especially Q learning (Watkins 1989) implemented using backpropagation networks (see section 3).

The action-centered explicit knowledge at the top level can also be learned in a variety of ways in accordance with the localist representations used. Because of the representational characteristics, one-shot learning based on hypothesis testing (Bruner et al 1956, Busemeyer and Myung 1992, Nosofsky et al 1994, Sun et al 1998, 2001) is needed.

The implicit knowledge already acquired in the bottom level can be utilized in learning explicit knowledge at the top level through *bottom-up* learning (Sun et al 1998, 2001). Likewise, the explicit knowledge already acquired in the top level can be utilized in learning implicit knowledge at the bottom level through *top-down* learning.

3 Some Model Details

Here are some details of CLARION (from Sun 2002).

3.1 Overall Action Decision Making

The overall algorithm of CLARION's action decision making is as follows:

1. Observe the current state x.
2. Compute in the bottom level the "value" of each of the possible actions (a_i's) associated with the input state x: $Q(x, a_1)$, $Q(x, a_2)$, ..., $Q(x, a_n)$. Stochastically choose one action according to Q values.
3. Find out all the possible actions (b_1, b_2, ..., b_m) at the top level, based on the the current state information x (which goes up from the bottom level) and the existing rules in place at the top level. Randomly choose one action.
4. Choose an appropriate action a, by stochastically selecting the outcome of either the top level or the bottom level.
5. Perform the action a, and observe the next state y and (possibly) the reinforcement r.
6. Update the bottom level in accordance with an appropriate algorithm (to be detailed later), based on the feedback information.

7. Update the top level using an appropriate algorithm (for constructing, refining, and deleting rules, to be detailed later).
8. Go back to Step 1.

3.2 Action-Centered Bottom Level

3.2.1 Representation

The input to the bottom level consists of three groups: (1) sensory input, (2) working memory items, (3) the top item of the goal stack. The sensory input is divided into a number of input dimensions, each of which has a number of possible values. The goal input is also divided into a number of dimensions (one of which is the goal dimension, the values of which are possible goals and at most one of them can be activated at one time). The working memory is divided into dimensions as well. Thus, input state x is represented as a set of dimension-value pairs: $(dim_1, val_1)(dim_2, val_2)....(dim_n, val_n)$.

The output of the bottom level is the action choice. It consists of three groups of actions: working memory set/reset actions, goal stack push/pop actions, and external actions. These three groups are computed by separate networks. See Figure 2 for a sketch of the overall structure of the bottom level.

In each network, actions are selected based on Q values. A Q value is an evaluation of the "quality" of an action in a given state: $Q(x, a)$ indicates how desirable action a is in state x. At each step, given the input state x, we compute the Q values of all the actions (i.e., $Q(x, a)$ for all a's). We then use the Q values to decide probabilistically on an action to be performed, through a Boltzmann distribution of Q values:

$$p(a|x) = \frac{e^{Q(x,a)/\alpha}}{\sum_i e^{Q(x,a_i)/\alpha}} \tag{1}$$

Here α controls the degree of randomness (temperature) of the decision-making process. (This method is also known as Luce's choice axiom; Watkins 1989.)

3.2.2 Learning

The *Q-learning* algorithm (Watkins 1989) is a reinforcement learning algorithm. In the algorithm, $Q(x, a)$ estimates the maximum (discounted) cumulative reinforcement that can be received from the current state x on:

$$\max(\sum_{i=0}^{\infty} \gamma^i r_i) \tag{2}$$

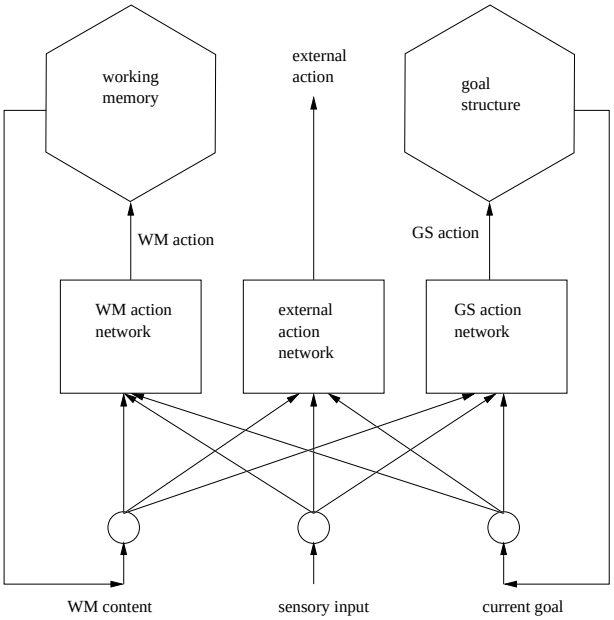

Figure 2: The three components of the bottom level.

where γ is a discount factor that favors reinforcement received sooner relative to that received later, and r_i is the reinforcement received at step i (which may be none).

Q values are gradually tuned, on-line, through successive updating, to enable sequential behavior to emerge. The updating of $Q(x, a)$ is based on:

$$\Delta Q(x, a) = \alpha(r + \gamma e(y) - Q(x, a)) \tag{3}$$

where γ is a discount factor, y is the new state resulting from action a in state x, and $e(y) = \max_b Q(y, b)$. Note that x and y include sensory inputs (internal and external), working memory items (if any activated), and the current goal (if exists). Thus, the updating is based on the *temporal difference* in evaluating the current state and the action chosen. In the above formula, $Q(x, a)$ estimates, before action a is performed, the maximum (discounted) cumulative reinforcement to be received if action a is performed; $r + \gamma e(y)$ estimates the maximum (discounted) cumulative reinforcement to be received, after action a is performed; so their difference (the temporal difference in evaluating an action) enables the learning of Q values that approximate the maximum (discounted) cumulative reinforcement prescribed earlier. Using Q-learning allows reactive sequential behaviors (that rely only on moment-to-moment input) to emerge.

Q-learning is implemented in backpropagation networks. Applying Q-learning, the training of the

backpropagation network is based on minimizing the following error at each step:

$$err_i = \begin{cases} r + \gamma e(y) - Q(x, a_i) & \text{if } a_i = a \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where i is the index for an output node representing the action a_i, and a is the action performed. Based on the above error measures, the backpropagation algorithm is applied to adjust internal weights (which are randomly initialized before training).

Clearly, calculation of Q values is done in a connectionist fashion through parallel spreading activation. Such parallel spreading of activation is assumed to be implicit as, e.g., in Hunt and Lansman (1986), Cleeremans and McClelland (1991), and Dienes (1992).

3.3 Action-Centered Top Level

At the top level, explicit knowledge is captured by a symbolic or localist representation (Clark and Karmiloff-Smith 1993), in which each unit is interpretable and has a clear conceptual meaning. This representation is different from that of the bottom level, because in the bottom level, backpropagation learning develops *internal* representation (in the hidden layer) that is *distributed* (whereby nodes are not individually meaningful).

3.3.1 Representation

The condition of a rule, similar to the input to the bottom level, consists of three groups: sensory input, working memory items, and the current goal. The output of a rule, similar to the output from the bottom level, is an action choice. It may be one of the three types: working memory actions, goal actions, and external actions.

Specifically, input x is made up of a number of dimensions (e.g., $x_1, x_2, ..., x_n$). Each dimension can have a number of possible values (e.g., $v_1, v_2, ..., v_m$). Rules are in the following form: *current-state* $\longrightarrow$ *action*. The left-hand side of a rule is a conjunction (i.e., AND) of individual elements. Each element refers to a dimension x_i of the input state x, specifying a value range, for example, in the form of $x_i \in (v_{i1}, v_{i2}, ..., v_{in})$. The right-hand side of a rule is an action recommendation a.

We translate the structure of a set of rules into that of a network (Sun 1992, 1994). Each value of each input dimension is represented by an individual node at the bottom level. Those nodes relevant to the condition of a rule are connected to the node at the top level representing that condition. When

given a set of rules, a rule network can be wired up at the top level, in which conditions and conclusions of rules are represented by respective nodes, and links are established that connect corresponding pairs of nodes (see Sun 1992 for further details).

For action decision making, at each step, one rule is randomly selected out of all the rules that match the current input state x. The selected rule gets to decide the action recommendation from the top level.

3.3.2 Bottom-Up Rule Learning

The *Rule-Extraction-Refinement* algorithm (RER) learns rules using information in the bottom level, to capture the bottom-up learning process (Stanley et al 1989, Karmiloff-Smith 1986). The basic idea of this algorithm is as follows: If an action decided by the bottom level is successful (i.e., if it satisfies a certain criterion) then the agent extracts a rule (with its action corresponding to that selected by the bottom level and with its condition specifying the current input state), and adds the rule to the top-level rule network. Then, in subsequent interactions with the world, the agent refines the constructed rule by considering the outcome of applying the rule: If the outcome is successful, the agent may try to generalize the condition of the rule to make it more universal; if the outcome is not successful, then the condition of the rule may be made more specific and exclusive of the current state.

The details of the operations used in the above algorithm (including rule extraction, generalization, and specialization) and the criteria measuring whether a result is successful or not (used in deciding whether or not to apply some of these operators) are described in Appendix. [2]

3.3.3 Fixed Rules

Some of the rules at the top level may be pre-coded and fixed. This type of rule (FR) may represent genetic pre-endowment of an agent presumably acquired through evolutionary processes, or prior knowledge acquired through prior experience in the world (Anderson 1983). For example, FRs may be given externally via instructions or textbooks.

Externally given FRs enable top-down learning. With these rules in place, the bottom level learns under the guidance of the FRs. That is, initially, the agent relies mostly on the FRs at the top level

[2]Notice that due to (random) variations in initial conditions and encountered stimuli sequences, both RER and Q-learning may learn different contents during different runs. Thus, individual differences may be captured.

for its action decision making. But gradually, when more and more knowledge is acquired by the bottom level through observing actions directed by the FRs, the agent becomes more and more reliant on the bottom level (given that the cross-level stochastic selection mechanism is adaptable). Hence, top-down learning takes place. (Note that this computational process of top-down learning is much simpler than that of, e.g., Maclin and Shavlik 1994.)

3.4 Goal Stack and Working Memory

The goal stack (GS for short) is simple (see, e.g., Anderson 1983). Items may be removed from the top of the goal stack, or may be added to the top of the stack. Only one goal may be active at a time—the top item on GS. A currently active goal becomes inactive when another item is added to the top of the goal stack, but may be reactivated when the items above it are removed. [3] Goal actions can be used either for pushing a goal onto the goal stack, or for popping a goal off the goal stack.

Working memory (WM for short) is for storing information temporarily for the purpose of facilitating subsequent decision making (Baddeley 1986). Working memory actions are used either for storing an item in the working memory, or for removing an item from the working memory.

4 Experiments

In this section, we conducted four simulations of the TOH task, using a 2×2 design: top-down versus bottom-up $\times$ WM/GS versus no WM/GS, which is based on our intended focus on top-down versus bottom-up learning, as well as based on the importance of WM/GS in traditional skill learning models.

4.1 Tower of Hanoi

Let us first review the TOH task, and some existing human data in this task. Tower of Hanoi is used here as an example of high-level cognitive skill learning, because it has been used extensively in cognitive skill acquisition research and is typical of the type of task addressed in such research. [4]

[3] Goal stack provides a simplified way of representing various drives, desires, needs, and motivations, and their interactions in an agent. Altmann and Trafton (2002) suggest that it is an approximation of a more complex cognitive process. Our use of GS here should also be viewed as an approximation of such a more complex process. Whether we use GS or a more complex process here is unimportant to the points to be made in this paper.

[4] We also dealt with and simulated other, more complex cognitive skill learning tasks. See Sun et al (2001, 2003) and Sun (2002).

Condition/No. of disks	2	3	4	5
No verbalization	0.0	2.1	4.3	21.2
Verbalization	0.0	0.0	0.9	1.3

Figure 3: The data of Gagne and Smith (1962), in terms of mean number of excess moves.

In the Tower of Hanoi task of Gagne and Smith (1962), there were three pegs. At the start, a stack of disks was stored on one of the pegs. The goal was to move these disks to another (target) peg. Only one disk can be moved at a time from one peg to another. These disks were of different sizes, and larger disks could not be placed on top of smaller disks. Initially, the stack of disks was arranged according to size so that the smallest disk was at the top and the largest was at the bottom.

In Gagne and Smith (1962), subjects were given 3-disk, 4-disk, and 5-disk versions of the task in succession, each version running until a final stable solution was found, and their mean numbers of moves (and excess moves) were recorded. Some subjects were instructed to verbalize: They were asked to explain why each move was made. The performance of the two groups of subjects (verbalization versus no verbalization) was compared. In this task, we intend to capture the verbalization effect on performance.

Figure 3 shows the performance of the two groups in terms of mean number of excess moves (in excess of the minimum required number of moves in each version). Comparing the verbalization group and the no verbalization group, the advantage of verbalization was apparent. ANOVA indicated that there was a significant difference between verbalization and no verbalization ($p < 0.01$).

Note that there have been corroborating findings in the literature. For instance, Stanley et al (1989) and Berry (1983) found that concurrent verbalization could help to improve subjects' performance in a dynamic control task under some circumstances. Reber and Allen (1978) similarly found that verbalization could help artificial grammar learning. Ahlum-Heath and DiVesta (1986) also found that verbalization led to better performance in learning Tower of Hanoi. In general, verbalization tends to force subjects to become more explicit and rely more on explicit processes (as argued in Sun et al 2001 and Sun 2002).

There have also been data concerning response time of each move made by human subjects in this task. For example, the RT data from Anderson (1993) are shown in Figure 4 (which incidentally included only a portions of the moves in each case). The data were obtained under the special instructions to subjects that encouraged a goal recursion approach (Anderson 1993). Among other

things, the instructions stated that "it is important to get the largest disks to their goal pegs first, since the ability to move a disk becomes more restricted as it becomes larger"; one should "create a situation in which you can move the larger disks, and concentrate on the smaller disks only after this has been done." There are four sets of data: the RT data for the performance on the 2-, 3-, 4-, and 5-disk versions of TOH (Anderson 1993). Each data point indicated the delay involved in moving a particular disk. Examining all of these cases, it is clear that there was a regular pattern of peaks, at disk 1, 3, 5, and so on, which arguably reflected planning carried out at those points, during which goal recursion (establishing a sequence of subgoals to be accomplished) happened. Such planning and goal recursion might be (at least partially) induced by the above instructions (Anderson 1993).

Another set of RT data is available in Anderson and Lebiere (1998), which had the complete data for all the moves in the 4-disk and 5-disk cases (Ruiz 1986). See Figure 5.

4.2 Bottom-Up Simulation 1

It is conceivable that some subjects learn implicitly first and then acquire explicit knowledge on that basis. Therefore, in this simulation, we used a configuration of (1) Q-learning at the bottom level and (2) RER learning at the top level . This is the quintessential configuration for bottom-up learning, in which implicit knowledge is first acquired in the bottom level, and then through practice, explicit knowledge is *extracted* from the bottom level and established at the top level (cf. Gunzelmann and Anderson 2003).

The Model Setup. At the bottom level, we used Q-learning implemented with backpropagation. A reward of 1 was given, when the goal state was reached. When the largest disk not on the target peg was moved to the target peg, a reward of 0.8 was given. If an illegal move was selected (such as attempting to move a large disk to the top of a smaller disk), a penalty of -0.1 was given. No WM and GS were used in this case.

At the top level, RER was used. The rule extraction threshold was 0.1. The rule generalization threshold was 3.6. The rule specialization threshold was 1.0. (See Appendix regarding details of these thresholds.) To capture the verbalization condition, we lowered these thresholds to 0.1, 2.0, and 0.5, respectively, to encourage more activities at the top level. The justification is that, as has been shown in previous work, subjects became more explicit under the verbalization condition (see, e.g., Stanley et al 1989 and Sun et al 2001). [5]

[5]When the model goes from the case of n disks to the case of $(n+1)$ disks, we transferred rules learned for the

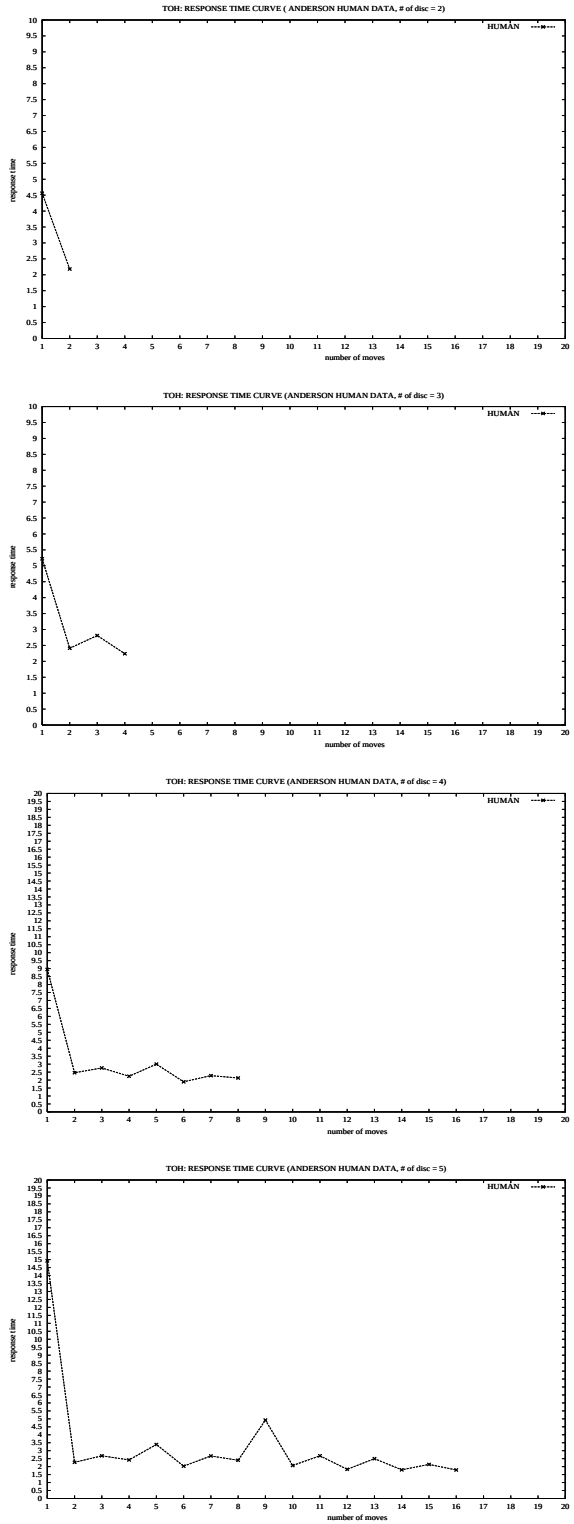

Figure 4: The response time data of Tower of Hanoi from Anderson (1993). Four cases are included.

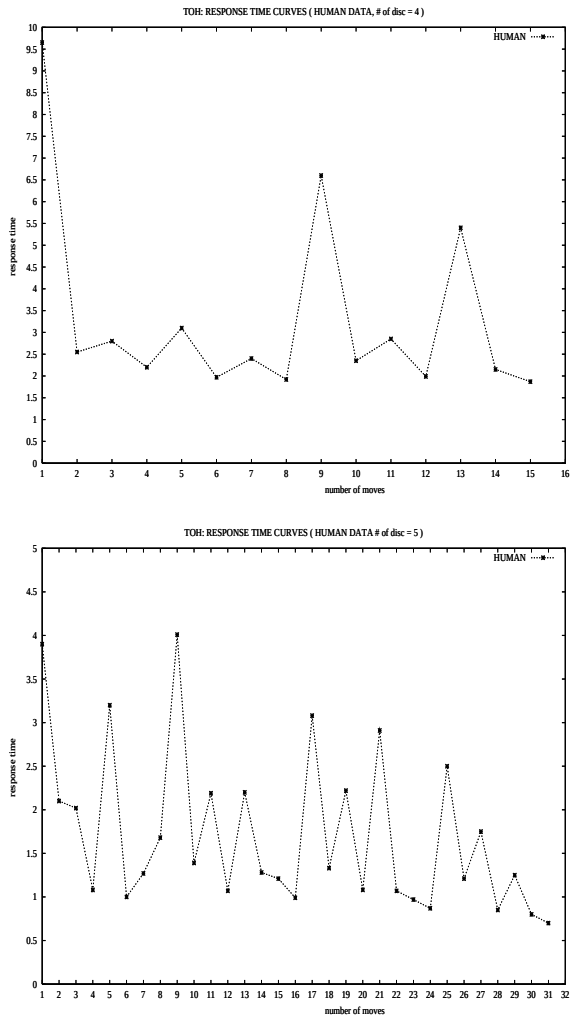

Figure 5: The response time data of Tower of Hanoi from Anderson and Lebiere (1998). Two cases are included.

Condition/No. of disks	2	3	4	5
No verbalization	0.0	0.1	1.9	8.9
Verbalization	0.0	0.0	0.2	1.5

Figure 6: The simulation of Gagne and Smith (1962), in terms of mean number of excess moves.

The parameters for determining stochastic selection of outcomes between the two levels were set at $\beta_{BL} = 5, \beta_{RER} = 1$ (see Appendix for details). For capturing the heavier reliance on the top level by the verbalization group (in correspondence with the known effect of verbalization—subjects became more explicit; Stanley et al 1989, Sun et al 2001), we changed the parameters to $\beta_{BL} = 1, \beta_{RER} = 2$ in the simulation of the verbalization group.

Corresponding to the human experiments, the stopping criterion during training was that a simulated subject reached the "almost-optimal" moves for 3 consecutive episodes, and was thus "stable" (whereby "almost optimal" was defined as within 160% of the optimal number of moves), or reached the maximal limit of training episodes (which was $5 * 3^{\text{number of disks}}$). [6] In calculating the dependent measure of this task (the mean number of excess moves), first, the mean number of total moves was calculated as the group mean of the average number of moves of the last three episodes. Then, the number of excess moves (beyond the minimum number of necessary moves) was calculated on that basis.

The Match. 20 model "subjects" were simulated in each group. The result of our simulation is shown in Figure 6. Analogous to the analysis of the human data, ANOVA (number of disks $\times$ verbalization versus no verbalization) indicated that in the model data, there was likewise a significant main effect between verbalization and no verbalization ($p < 0.01$), confirming the verbalization effect in the human data as we mentioned earlier.

In addition, we compared this bottom-up simulation with a *bottom-only* simulation, in which the top level was in effect shut down. This bottom-only simulations consistently failed to learn, whatever

n-disk version to the case of $(n+1)$ disks, because this task was highly recursive. The transfer to the $(n+1)$ case was accomplished by three consecutive stages applied recursively: (1) move the n sub-tower (all disks but the largest) from the original peg to the spare peg, (2) move the largest disk $(n+1)$ from the original peg to the target peg, and (3) move the n sub-tower from the spare peg to the target peg. The rules learned in the n-disk case can be used to carry out different stages of the $(n+1)$-disk case.

[6] Note that when "almost optimal" performance was achieved, a set of RER rules were extracted that corresponded to the sequence of steps.

parameter values we tried, even when given 10 times as many training trials. This result strongly suggested the importance of top-level explicit knowledge and bottom-up learning— Without them, the task was hard to learn. On the other hand, without the bottom level, the top level could not learn efficiently either (because it had to rely on random search). These two facts were consistent with our "synergy hypothesis" earlier (see Sun and Peterson 1998, Sun et al 2001): The reason why there are these two distinct levels is because of the synergy that may be generated from the interaction of the two levels. The interaction of the two levels helps to improve learning, and/or facilitate performance and transfer (Sun et al 2001).

However, throughout our experiments, the bottom-up simulation (as well as the bottom-only and the top-only simulation) failed to capture the RT data reported earlier. They failed to capture those peaks in the RT data, which we believed reflected planning and goal recursion (establishing a sequence of subgoals) in the human performance.

4.3 Bottom-Up Simulation 2

We want to compare two possibilities of simulating the Tower of Hanoi task: one with WM and GS and the other without, both based on bottom-up learning (i.e., RER). The reason to involve WM/GS was because the use of WM/GS was prevalent in theorizing on cognitive skill acquisition, and in particular in prior modeling of TOH. Therefore, we extended the previous simulation and conducted the following bottom-up simulation adding both WM and GS. The same as external actions, all actions on WM and GS were learned, through bottom-level backpropagation networks and through top-level RER rules.

The Model Setup. To implement this simulation, we set up the following: (1) For deciding on each type of action (external, GS, or WM actions), there is a corresponding network and a corresponding set of RER rules, respectively. (2) The input to each network is the same, including sensory input, the top GS item, and WM items. (3) The output of the networks include external actions, GS actions and WM actions, respectively. (4) At each step, if the actions are decided by the top level, we use the existent RER rule sets to get three actions—external, GS or WM actions; if the actions are decided by the bottom level, we select an action from the output of each network (for external, WM, and GS actions respectively). (5) The chosen actions are coordinated in execution, and the top level and the bottom level are updated then.

Computationally speaking, GS is not necessary. But we included GS in this model, because there were indications that GS was used in human performance (Anderson 1983, 1993). For our simulation,

each GS item included the following information (concerning both a subtower and a focal disk):

DSIZE: Size of SUBTOWER

FROM: Current peg of SUBTOWER

TO: Target peg of SUBTOWER

DSIZE1: Size of FOCAL-DISK

FROM1: Current peg of FOCAL-DISK

TO1: Target peg of FOCAL-DISK

A subtower is a set of disks at the top of a peg. The focal disk is the disk beneath a subtower. In each GS item, TO and TO1 describe the intended operations, while other items describe the current state. [7] Multiple goal items could be stored in GS, one on top of another.

A simple and plausible set of goal recursion rules is as follows (cf. Anderson 1993), where function LOC indicates the peg on which a disk is located, function SIZE indicates the size of a disk, and DSIZE, SUBTOWER, FOCAL-DISK etc. refer to the corresponding components of the top goal item on GS:

If DSIZE > 0, then push a new goal for moving a subtower of size DSIZE-1 to the spare peg and for moving the disk of size DSIZE to its target peg.

If DSIZE $= 0$, then make a move of FOCAL-DISK to its target peg.

If LOC(SUBTOWER)=TO and LOC(FOCAL-DISK) $\neq$ TO1, then move FOCAL-DISK to its target peg.

If LOC(SUBTOWER)=TO and LOC(FOCAL-DISK)=TO1, then pop the current goal.

Recall that a similar set of rules was hand-coded into the model in the ACT-R simulation of Anderson (1993). However, in this simulation, we did not use such hand-coded, a priori rules in the model. We want the model itself to learn something that has essentially the same effect (in both the bottom level and the top level).

The Match. The result of our simulation is shown in Figure 7. Analogous to the analysis of the human data, ANOVA (number of disks $\times$ verbalization versus no verbalization) indicated that in the

[7] Note that this set of information is redundant. For example, we have, by necessity, FROM=FROM1, DSIZE = DSIZE1 -1, and TO specifies the only peg that is different from FROM/FROM1 and TO1. In addition, we can ascertain FROM/FROM1 from observing the current configurations of the pegs. Thus, the only two pieces of information that are necessary are DSIZE1 and TO1.

Condition/No. of disks	2	3	4	5
No verbalization	0.0	1.6	3.2	10.5
Verbalization	0.0	0.4	0.9	2.5

Figure 7: The simulation of Gagne and Smith (1962), in terms of mean number of excess moves.

model data, there was a significant main effect between verbalization and no verbalization ($p < 0.01$), confirming again the verbalization effect we discussed.

However, as in the previous simulation, this simulation failed to capture the RT data reported earlier, in particular those peaks that might reflect planning and goal recursion.

This comparison (bottom-up learning with or without GS/WM) shows that GS/WM does not make much difference at all, in the context of bottom-up learning (RER at the top level). It is possible that in other circumstances, GS/WM may make a significant difference. The following two simulations address this issue.

4.4 Top-Down Simulation 1

This alternative simulation of the TOH task was aimed at demonstrating the use of "fixed rules", along the line of Anderson's (1993) model, in capturing human data. Our simulation also involved the bottom level, which might interfere with the top-level FRs. Therefore, compared with Anderson's, this constituted a more complex simulation, using a more complete cognitive model that involved both explicit and implicit knowledge (Reber 1989, Seger 1994, Sun 1994, 2002). Furthermore, this simulation explored top-down (i.e., explicit-to-implicit) learning in the TOH task (Mathews et al 1989, Regehr and Brooks 1993).

This alternative simulation of TOH was also aimed at demonstrating the role of goal stack and working memory in capturing human data, extending Anderson's (1993) model. (Thus, this simulation should also be compared to the next one using FRs but not WM and GS.)

The Model Setup. Fixed rules were used, which implemented, as a subset, Anderson's (1993) analysis of subjects' performance of this task. That is, we first implemented the FRs involving GS and WM mentioned earlier, which were similar to the rules used in Anderson (1993), in the top level of CLARION. However, this simulation was more complex than top-level only (rule-based only) simulations because it had to deal with the interference from the bottom level, as the bottom level was

Condition/No. of disks	2	3	4	5
No verbalization	0.00	1.50	4.90	12.55
Verbalization	0.00	0.25	0.90	2.65

Figure 8: The third simulation of Gagne and Smith (1962), in terms of mean number of excess moves.

running in parallel with the top-level rules but might recommend different actions and thus interfere with the top-level goal recursion process. The main change lay in the process of popping a sequence of goals from GS, when a move made by the bottom level was not consistent with the top goal in GS. In that case, we kept popping goals until reaching a goal on the GS that was consistent with the move. (Due to length and complexity, the extended set of fixed rules is presented in Appendix.)

In the bottom level, Q-learning was used. The schedule for reinforcement was as follows: A simulated "subject" was given 1.0, if the goal was achieved, 0.1 if the move by the bottom level was consistent with the current goal, 0.0 if the move by the bottom level was inconsistent with the current goal but the action was legal, -0.1 if the move was illegal. Due to the involvement of fixed rules, Q-learning at the bottom level was under the "guidance" of the top level in this simulation.

For the non-verbalization subjects, the parameters for stochastic selection of outcomes of the two levels were set at $\beta_{FR} = 1, \beta_{BL} = 1$. For capturing the performance of the verbalization subjects, the parameters were adjusted to reflect their tendencies to rely more on the top level (that is, subjects became more explicit): $\beta_{FR} = 10, \beta_{BL} = 1$.

Note that RER rules were not used. However, RER rules could be easily added. We tried adding RER rule learning. There was no significant difference in results.

The Match. The result, comparing verbalization versus no verbalization, is shown in Figure 8. Analogous to the analysis of the human data, ANOVA (number of disks $\times$ verbalization versus no verbalization) indicated that, in the model data, there was a significant main effect between verbalization and no verbalization ($p < 0.01$), confirming again the verbalization effect we discussed.

We further tackled the capturing of the RT data from Anderson (1993), which incidentally included only a portion of the moves in each case. As mentioned earlier, the human data were obtained under the special instructions to subjects that encouraged goal recursion as embodied by the fixed rules used in the top level of CLARION. (The calculation of RTs is described in Appendix.) Figure 9 and Figure 10 show the data.

As shown by Figure 11, response times of the two simulated groups were reasonably close to the human data (where there was no distinction between verbalization and no verbalization). Although the match of both groups was excellent, the match between the simulated verbalization group and the human RT data was closer. The MSEs for the simulated verbalization group were 0.002 (2-disk), 0.529 (3-disk), 0.252 (4-disk), and 1.555 (5-disk). The overall MSE for the simulated verbalization group was 0.967. The MSEs for the simulated non-verbalization group were 0.222 (2-disk), 0.086 (3-disk), 0.579 (4-disk), and 3.271 (5-disk). The overall MSE for the simulated non-verbalization group was 1.925. Since MSEs do not provide a sense of the relative magnitude of deviations, we also calculated the relative MSEs (i.e., the χ^2 mean difference). [8] The relative MSEs for the simulated verbalization group were 0.001 (2-disk), 0.107 (3-disk), 0.098 (4-disk), and 0.299 (5-disk). The overall relative MSE for the simulated verbalization group was 0.200. The relative MSEs for the simulated non-verbalization group were 0.049 (2-disk), 0.024 (3-disk), 0.109 (4-disk), and 0.375 (5-disk). The overall relative MSE for the simulated non-verbalization group was 0.236. The match between the model and human data was excellent, judging from the numbers.

We also captured the RT data of Anderson and Lebiere (1998), which had the complete data of all the moves in two cases. The data were available for the 4-disk and 5-disk cases only. The comparisons were thus in these two cases. Figure 12 shows the match.

As shown by Figure 13, The MSEs for the simulated verbalization group were 0.358 (for the 4-disk case) and 0.261 (for the 5-disk case). The overall MSE for the simulated verbalization group was 0.292. The relative MSEs for the simulated verbalization group were 0.109 (for the 4-disk case) and 0.173 (for the 5-disk case). The overall relative MSE for the simulated verbalization group was 0.152. The match between the model and human data was excellent, judging from the numbers, roughly comparable to that of Anderson and Lebiere (1998). [9]

In both cases, these characteristic RT peaks in the human data (see Figures 9, 10, and 12) were captured in simulation by a sequence of goal pushes at each of these points. According to the FRs used (see Appendix), it happened that at these points, multiple goals needed to be pushed onto GS. In other words, at these points, planning needed to be carried out. Planning accounted for the RT

[8] The relative MSE is defined as follows: $\frac{1}{n}\sum_{i=1}^{n}\frac{(HumanMean_i - ModelMean_i)^2}{HumanMean_i}$.

[9] In comparison, the MSEs for the simulation of Anderson and Lebiere (1998) based on ACT-R were 0.061 and 0.176 for these two cases, respectively. Their overall MSE was 0.14. The relative MSEs for the simulation of Anderson and Lebiere (1998) based on ACT-R were 0.020 and 0.110 for these two cases, respectively. Their overall relative MSE was 0.081.

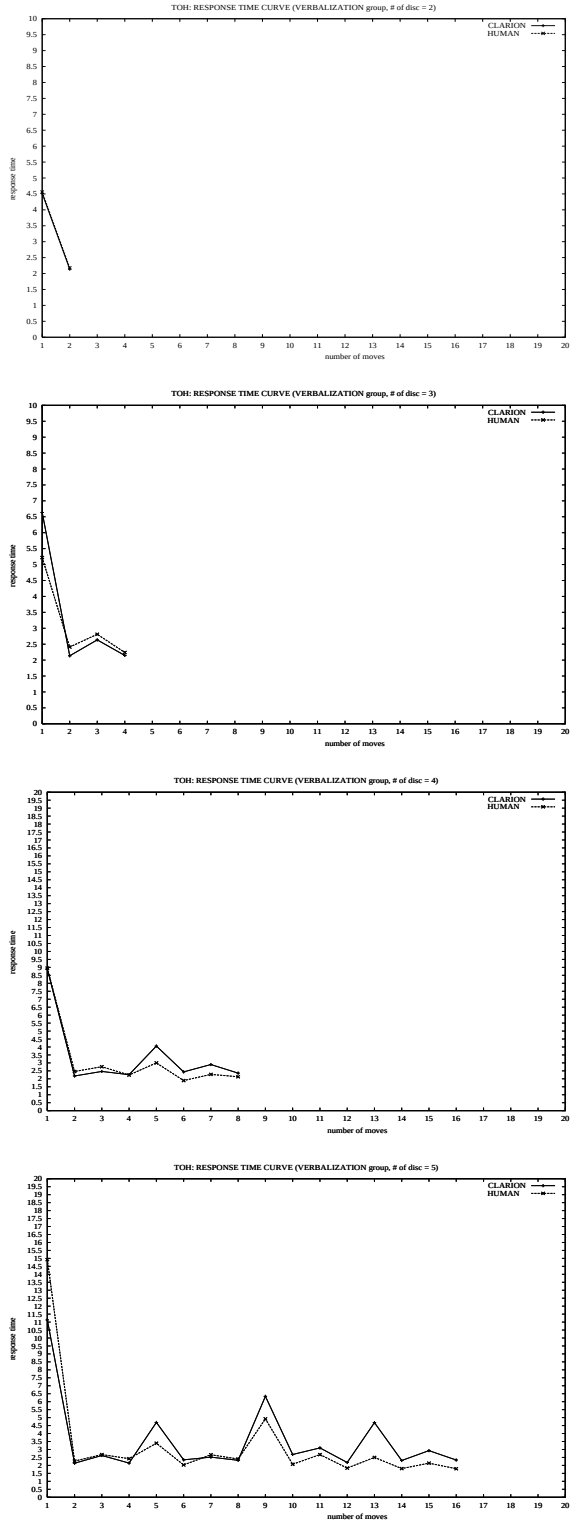

Figure 9: Simulation of the response time data of Tower of Hanoi from Anderson (1993). Four cases are included. The verbalization group is used.

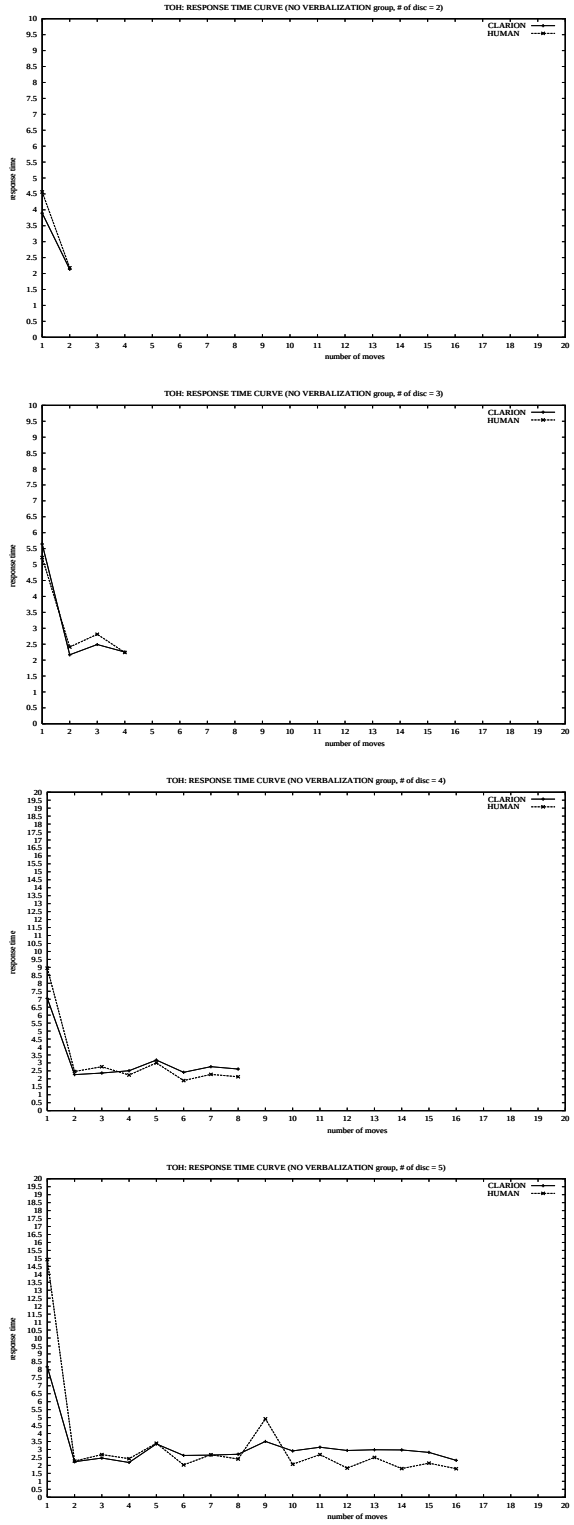

Figure 10: Simulation of the response time data of Tower of Hanoi from Anderson (1993). Four cases are included. The no verbalization group is used.

the verbalization group:

	MSE	relative MSE
2-disk	0.002	0.001
3-disk	0.529	0.107
4-disk	0.252	0.098
5-disk	1.555	0.299
overall	0.967	0.200

the non-verbalization group:

	MSE	relative MSE
2-disk	0.222	0.049
3-disk	0.086	0.024
4-disk	0.579	0.109
5-disk	3.271	0.375
overall	1.925	0.236

Figure 11: The MSEs and the relative MSEs of RT simulations of Tower of Hanoi.

peaks.

We also examined the learning of the bottom level, although in this case the rules at the top level took the bulk of the responsibility for deciding the moves. One way of examining the learning of the bottom level was testing its performance alone periodically to see whether there was any improvement. Improvement over time was clearly demonstrated with regard to number of moves in an episode, as shown by Figure 14. Another way of detecting the improvement of performance was observing the increase of the number of consistent moves made by the bottom level (i.e., the number of moves consistent with what was recommended by the fixed rules at the top level). Figure 15 showed such an increase. Yet another way of detecting the improvement of performance was observing the change in terms of number of goal pops caused by failure to achieve goals. This number should go down over time, which was confirmed by Figure 16. In addition, we may also observe the change of the probability of selecting the bottom level, which should increase over time, if the bottom level was learning and thus improving its performance (see Appendix). Figure 17 showed such an increase.

Together, these figures showed the gradual learning of the bottom level, guided by the fixed rules at the top level. The learning was clearly due to the guidance provided by the top level. Therefore, they demonstrated top-down learning, the opposite of bottom-up learning that we demonstrated earlier.

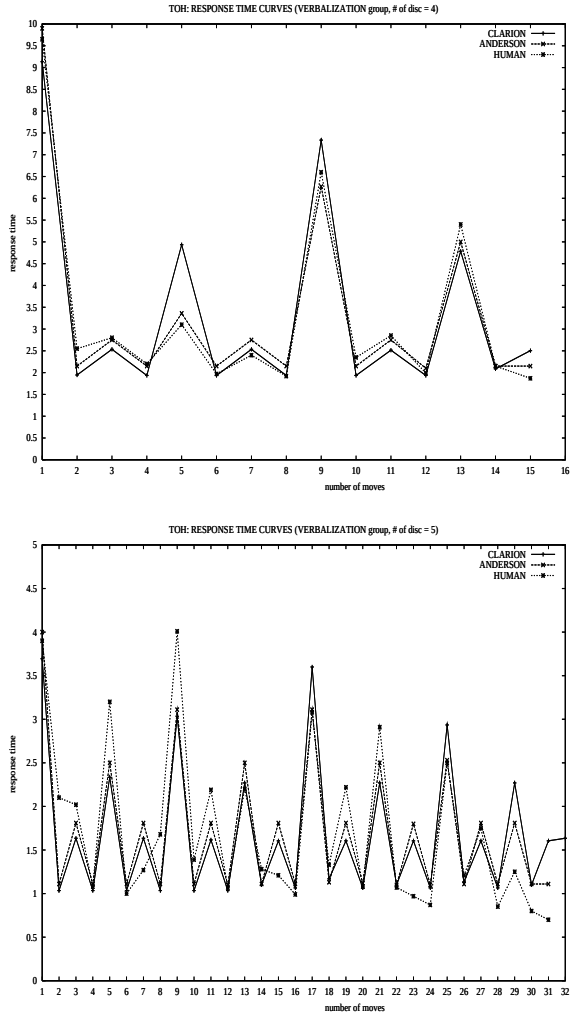

Figure 12: Simulation of the response time data of Tower of Hanoi from Anderson and Lebiere (1998). Two cases are included. The verbalization group is used.

the verbalization group:

	MSE	relative MSE
4-disk	0.358	0.109
5-disk	0.261	0.173
overall	0.292	0.152

Figure 13: The MSEs and the relative MSEs of another set of RT simulations of Tower of Hanoi.

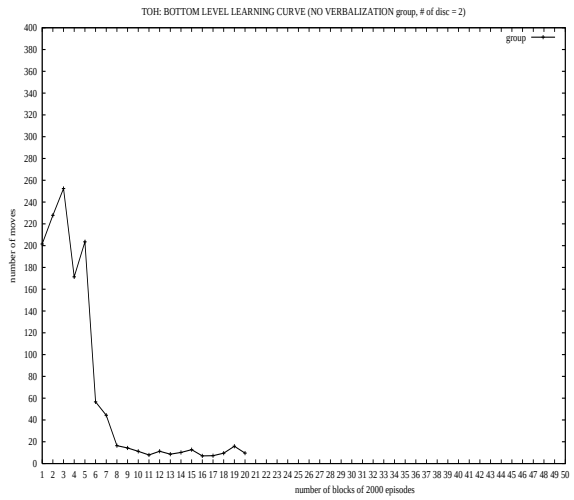

Figure 14: Performance improvement of the bottom level in terms of average number of moves in an episode.

The simulation shows that CLARION, despite its focus on bottom-up learning, can accommodate this (secondary) direction of learning as well.

Some may argue that top-down learning was rather slow. The answer is two-fold: Computationally speaking, backpropagation learning invariably requires a long training time, often longer than human subjects by orders of magnitude, when compared trial by trial. This is an outstanding issue in connectionist cognitive modeling. However, psychologically speaking, we need not compare models and humans on a trial-by-trial basis. It may be the case that several trials of a model can be equivalent to one trial of a human. Alternatively, rehearsal and memory consolidation may account for many of the model trials (McClelland et al 1995), and thus reduce the number of actual model trials.

This particular simulation shows that the CLARION framework can accommodate traditional accounts of human performance in this task (such as Simon 1975, Anzai and Simon 1979, Ruis 1986, Anderson 1993, Anderson and Lebiere 1998, Gunzelmann and Anderson 2003). Moreover, it extends such accounts by incorporating implicit processes (at the bottom level) as well as explicit processes (at the top level). The role of the bottom level in this task (and other high-level cognitive skill tasks) is identified to be that of "quick-and-dirty" reactions that may lead to bad performance initially due to interference with top-level rule-guided actions, but may also lead to faster and better performance later on given sufficient training.

The account of human RT data is important, because such an account has been viewed as the hallmark of a successful simulation. Here, we show that CLARION can capture the essential characteristics

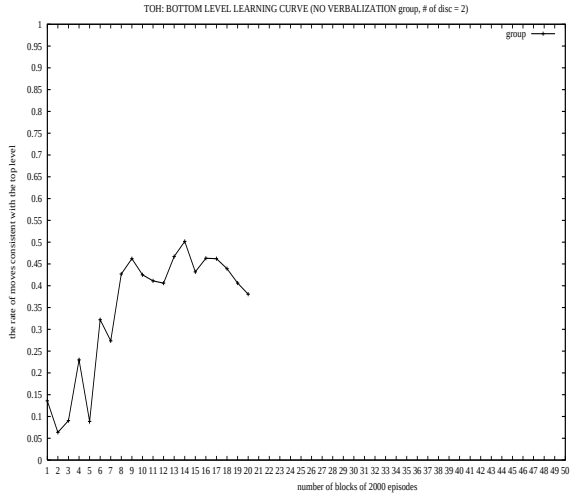

Figure 15: Performance improvement of the bottom level in terms of rate of consistent moves.

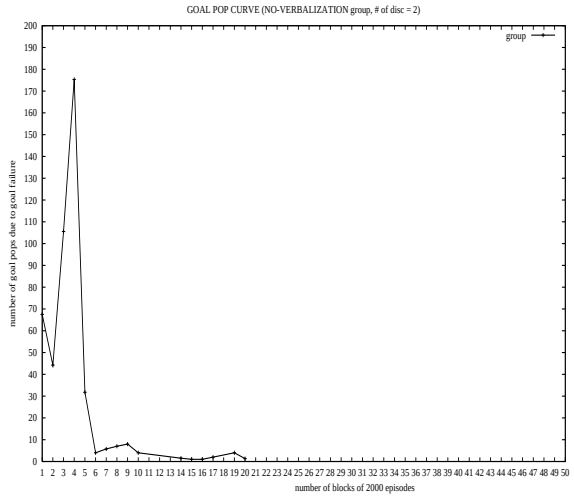

Figure 16: Performance improvement of the bottom level in terms of number of goal pops caused by failure.

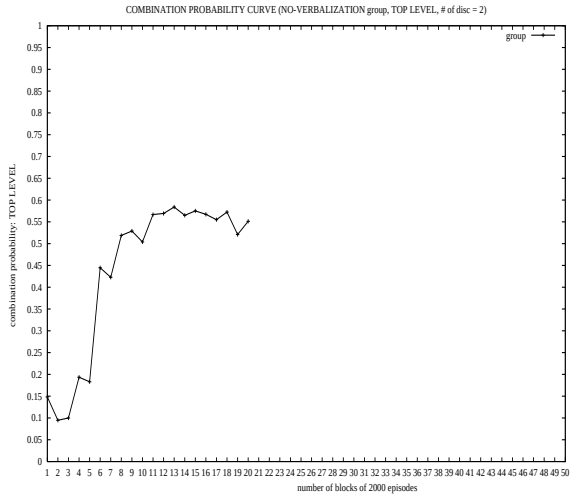

Figure 17: Performance improvement of the bottom level as demonstrated by the change of the combination parameter.

of the human RT data.

4.5 Top-Down Simulation 2

As a comparison to the previous simulation, this new simulation used a different set of fixed rules—a set of rules that decided which disk to move and in which direction without the use of GS/WM. This set of rules avoided using GS or WM in deciding a move, because it could make decisions without any additional information other than the current peg/disk configuration.

The Model Setup. First, we define the three pegs as follows: Peg 0 is the original peg, peg 1 is the spare peg in the middle, and peg 2 is the goal peg. We define the disks as 0, 1, 2,...., n (where n = 2, 3, 4, 5, ...), from the smallest to the largest. We also define two kinds of disk moves: (1) clockwise moves are ones going from peg m to peg $m-1$ mod 3. This kind of move includes: peg 2 to peg 1, peg 1 to peg 0, and peg 0 to peg 2. (2) counter-clockwise moves are ones going from peg m to peg m+1 mod 3. This kind of move includes: peg 0 to peg 1, peg 1 to peg 2, and peg 2 to peg 0.

With the above definitions, it is easy to show that it is generally true in an optimal sequence of disk moves that the largest disk (n) moves clockwise, the second largest disk (n-1) moves counter-clockwise, the third largest disk (n-2) moves clockwise, and so on. The clockwise moving disks include n, n-2, n-4, ..., n -2*i, ... (where i = 0, 1, ...) and the counter-clockwise moving disks include n-1, n-3, n-5, ..., n -2*i-1, ... (where i = 0, 1, ...). Therefore, we can come up with the following alternative FR

for dealing with the TOH task: If (1) a disk is at the top of any of the three pegs, (2) the disk is not the one moved most recently, (3) it is the smallest of all the disks that match conditions (1) and (2), then, move the disk clockwise or counter-clockwise according to its disk number. One can easily verify that this FR works properly with regard to any number of disks. This rule is rather appealing due to its simplicity. (However, it is evident that this rule, while easy to use, is hard for subjects to discover on their own, and therefore, we do not expect that too many subjects would use it (Gagne and Smith 1962).)

The bottom-level learning was the same as in the previous simulation. No GS or WM was used. No RER rule was used. The reinforcement schedule was the same as in the previous simulation. [10]

The Match. The simulation using this set of FRs achieved optimal performance immediately if only the top level was used. But if the bottom level was involved, it showed initially suboptimal performance. Over time, the performance of the bottom level improved, and eventually caught up with the performance of the top level. Due to the fact that exactly the same process as the previous simulation was in the working in this simulation, we will not repeat the details.

However, due to its lack of goal recursion processes (different from the immediately preceding simulation, but the same as the first 2 simulations), it could not capture the RT characteristics of human subjects as reported in Anderson (1993) and Anderson and Lebiere (1998). This was (at least partially) due to the experimental condition under which these human subjects were tested: They were explicitly encouraged to use goal recursion to solve this problem, instead of using alternative means such as that represented by the above FR. Therefore, naturally, their performance (including characteristics of their RTs) was best captured by the previous FRs implementing a goal recursion procedure. It is an open question whether alternative experimental setups can encourage subjects to adopt approaches similar to what is represented by this simulation, as well as by the first two simulations.

[10] When the bottom level is also involved, the top-level rule needs to deal with the interference by the bottom level. One simple way of doing this is for the top level to reverse a move made by the bottom level if it is inconsistent with the recommendation of the top level. This is feasible if the probability of choosing the bottom level is reasonably low.

5 Discussions

5.1 Comparing the Four Simulations

First, comparing using the bottom level only versus using the bottom level along with the top level, it is clear that the bottom level alone cannot capture human performance in this task adequately. Or put it another way, implicit learning as embodied by the bottom level was inadequate for learning this task as humans do. With the addition of explicit knowledge (in the top level), be it RER rules (extracted from the bottom level), or FRs (hand-coded a priori), the model performed adequately in capturing the human data (concerning numbers of moves).

There is the question of why RER rules can help the bottom level to learn better, since RER rules are extracted from the bottom level in the first place. The explanation has been discussed in detail in Sun and Peterson (1998): The conclusion, based on a systematic analysis, was that the explanation of the *synergy* between the two levels rests on the following factors: (1) the complementary representations of the two levels (discrete versus continuous); (2) the complementary learning processes (one-shot rule learning versus gradual Q value approximation); and (3) the bottom-up rule learning criterion used in CLARION. (Due to lengths, we will not repeat the analysis here. See Sun and Peterson (1998) for details.) In other words, although rules are extracted from the bottom level, the very process of extraction and the resulting representation (which is quite different from that of the bottom level) make rules different and useful.

A more technical explanation can also be provided, based on considerations of function approximation (Sun and Peterson 1998). A good action in a given state tends to be an action with an above-average Q value in the given state. But, with a neural network (a function approximator), each Q value tends to be "averaged" by neighboring Q values (for different actions and states), which is due to network generalization/approximation that tends to "smooth" the output Q value surface. Counteracting this tendency, extracted rules can restore those cut-off peaks in the Q value landscape, because of the use of individuated, crisp representation. These rules supplement the approximated Q values (produced by a function approximating neural network) and compensate for the (over)generalization therein. Synergy is thus generated as a result.

Next, comparing the bottom-up version involving GS/WM with the one without involving GS/WM (i.e., bottom-up simulation 1 versus 2), there is no significant difference. This, in a way, indicates that GS and WM are not necessary for capturing the human data of Gagne and Smith (1962). Although

for capturing those currently available RT data, the FRs using GS and WM were better than those not using them, the explanation may actually lie elsewhere: As stated before, the fact that goal recursion is prominent in the human data we examined may possibly be due to the experimental procedure used on human subjects that emphasized goal recursion, instead of due to the inherent importance of GS/WM (more discussions of this point later).

Comparing the top-down simulation 1 (with FRs) with the bottom-up ones (without FRs), we see that the FRs are good at capturing goal recursion, which is otherwise difficult to capture. We may ask the following question: Why can RER not learn this goal recursion procedure by itself? The explanation is as follows: Human subjects likely have a large amount of pre-existing knowledge about the world and many heuristic problem solving techniques learned over the life time. In solving TOH tasks, it is very likely that they apply these a priori knowledge to come up with an approximation of the goal recursion procedure. However, CLARION starts with no a priori knowledge that can be applied to come up with anything resembling goal recursion. As a result, it is slow to learn such a procedure (if ever). This explanation implies that if we can capture subjects' a priori knowledge better, a model may be constructed that can learn goal recursion. This is a topic for future research.

Comparing using FRs only versus using FRs plus the bottom level, the FRs only version (Anderson 1993) is simple, because there is a simple set of rules describing a simple procedure for solving the TOH problem. Because of the simplicity of the procedure, it is easy to program and run the model, and it is also easy to fit the model to the human RT data. However, such a model ignores an important aspect of human cognition, namely, implicit cognitive processes. As a result, the model seems too simple and too clean to be cognitively realistic. Incorporating the bottom level (which captures implicit processes) remedies to some extent this problem, and it thereby, in our view, makes the resulting model more cognitively realistic.

Comparing the two different versions of FRs (in the two top-down simulations respectively), along with the RER rules learned (in the first two simulations), we notice that it is not necessary that human subjects learn goal recursion (such as that studied by Anderson 1993). It is also possible that they learn alternative rules, such as those discovered by RER or those used for top-down simulation 2 (although much less likely). Even though the goal recursion procedure of Anderson (1993) is a generic and powerful one, it should be interesting to explore alternative knowledge that can be employed in solving this task (and other similar tasks). It is possible that if we provide subjects with different initial instructions, avoiding emphasizing goal recursion, they may be more likely to develop alternative knowledge and procedures. This should be explored further experimentally in the future.

5.2 Evaluation

Along with the simulation of other tasks (see Sun 1999, Sun et al 1998, 2001, 2003), we demonstrated that CLARION is capable of both bottom-up and top-down learning, although it was initially developed as a purely bottom-up learning model. The original reason for developing a bottom-up learning model was that in the existing literature, bottom-up learning has been very much ignored as pointed out by Sun and Peterson (1998) and Sun et al (2001, 2003), and therefore, there is a need to counter-balance this bias. Our bottom-up learning model, since then, has been successful in accounting for a wide variety of skill learning tasks in a bottom-up fashion, ranging from serial reaction time tasks (sequence learning tasks), to minefield navigation tasks (Sun et al 2001). But one lingering question has been: Can this same model account for different directions of learning, in particular top-down learning? The present work answers this question clearly in the affirmative: CLARION can not only account for bottom-up learning data, but also top-down learning ones.

There have been a variety of experimental demonstrations of both bottom-up and top-down learning. For example, Stanley et al (1989), Bowers et al (1990), Mandler (1992), Karmiloff-Smith (1986), and Sun et al (2001) showed evidence of bottom-up learning. Anderson (1983) and Anderson (1993) demonstrated top-down learning. Aizenstein et al (2000) showed some brain imaging data (increase and decrease of activities in certain brain regions) that might be interpreted based on bottom-up and top-down learning. Evidence supports the existence of both types of learning.

A shortcoming is that the data in the TOH task did not speak directly to the issue of top-down versus bottom-up learning. We do not have data that show clearly either top-down or bottom-up learning in this task. Such data, unfortunately, are not available. We have to use whatever is currently available. In this regard, we need more and better designed human experiments in the future that can provide deeper insights into the issue, in order to further validate our model in this respect.

In this work, we chose to use the task of Tower of Hanoi for testing possibilities of top-down versus bottom-up learning, because the task is a benchmark in high-level cognitive skill acquisition and has been used in many previous studies of skill acquisition, cognitive modeling, and cognitive architectures. Another reason for its selection is that simulating this type of task expands the range of coverage of CLARION: Instead of the low-level skill domains that we simulated initially, tackling this task (as well as a number of other tasks reported elsewhere) shows that CLARION can also capture high-level cognitive skill acquisition. Furthermore, although it may be ideal that we use a task whereby there are data that clearly show both top-down and bottom-up learning, this kind of task or data does not exist

at present. We have to use whatever is available that may be co-opted for our purpose. Therefore, based on the above considerations, the choice of Tower of Hanoi in this study is justified.

5.3 Comparisons with Other Models

Let us compare briefly CLARION with other computational cognitive models in general.

First, there have been various models of pure implicit learning. For example, Cleeremans and McClelland (1991) simulated a sequential reaction time task. They employed a recurrent backpropagation network that saw one position at a time but developed an internal context representation over time that helped to predict next positions. The model succeeded in matching human data in terms of degrees of dependency (conditional probabilities) on preceding segments in a sequence. However, their success was obtained through introducing additional mechanisms for several types of priming (e.g., short-term weight changes and accumulating activations).

Dienes (1992) compared connectionist networks (partially or fully recurrent), using the Delta learning rule (which was similar to backpropagation) or the Hebb rule (which focused on direct associations), and a number of memory-array (instance-based) models. The goal was capturing human learning data in artificial grammar learning tasks. [11] The models were successful in terms of accounting for the human data that was examined (which were not concerned with either top-down or bottom-up learning),

One general shortcoming of these above models is that mostly these models focused only on implicit learning, and they ignored (1) the role of explicit learning in these tasks, (2) the interaction between explicit and implicit processes in learning the tasks, and in particular (3) the possibility of bottom-up learning or top-down learning.

There are also many models of explicit learning. In SOAR (Rosenbloom et al 1993), learning is explicit: Learning consists of *chunking*, the creation of a new production that summarizes the process leading to achieving a subgoal.

There have been a few combined models of implicit and explicit learning. Among them are Cleeremans (1994), Erickson and Kruschke (1998), and Sun et al (1998, 2001). Cleeremans (1994) used a simple buffer network to capture the effect of explicit knowledge, along with a simple recurrent network for capturing implicit learning. The buffer network mimicked the explicit retrieval of explicitly

[11] Dienes attempted to match the models with the human data on a number of measures, including percent correct, rank ordering of string difficulty, percentage of strings on which an error was made on all the presentations, and percentage of strings on which an error was made only on some of presentations.

stored items, through using a backpropagation network that has a buffer as part of its input. The output of the buffer network was fed into the hidden units of the main network. The outcomes from the two networks were thus combined before the final output was produced. However, in other, more complex types of tasks (Sun et al 2001), the buffer network, as is, is inadequate for capturing explicit knowledge used in performing these tasks. Furthermore, in more complex tasks, implicit and explicit knowledge may have more complex interactions. While CLARION can accommodate more complex interactions, this model may have trouble doing so.

In Erickson and Kruschke's (1998) model, there are two modules: the rule module and the exemplar module. Both rules and exemplars (in their respective modules) are learned using gradient descent. The model, concerned with classification only, matched some human classification data well.

ACT-R (Anderson 1993, Anderson and Lebiere 1998) encodes each piece of knowledge (a chunk or a production) in both a symbolic (explicit) form and a subsymbolic (implicit) form. However, although it accounts for some top-down learning, the model does not account for bottom-up learning.

There have been a number of top-down models. Among them, Schneider and Oliver (1991) was concerned with automatization in skill learning. A deliberate (explicit) calculation was performed first but later, through repeated trials, an automatized (implicit) process took over. Both processes were implemented in neural networks. Their model implemented the process through which explicit knowledge was assimilated into implicit skills.

Hunt and Lansman (1986) hypothesized another top-down learning process for explaining automatization data. They incorporated two separate components in their model: They used a production system for capturing controlled processes, and a semantic network for capturing automatic processes. They hypothesized that, through practice, production rules were assimilated into the semantic network, thus resulting in automatic processes (through spreading activation in the semantic network).

On the other hand, there are few models in the other direction—that is, bottom-up learning. The only one that we are aware of is the present model CLARION. Consequently, the only model that combines top-down and bottom-up learning that we are aware of is CLARION.

5.4 Concluding Remarks

In summary, this work explores an important but often neglected issue in skill acquisition: the interaction of implicit and explicit learning, and the resulting two directions of learning—top-down versus

bottom-up (Sun 1997, 1999, Sun et al 2001). This issue evidently has been very much neglected in the literature, but is crucial in gaining a complete understanding of human skill acquisition processes. We would claim that human skill acquisition processes often involve both implicit and explicit processes and thus involve their interaction (Sun et al 1998, 2001, 2003); as a result, we need to understand top-down and bottom-up learning as two ways of interaction.

Our simulations are merely a starting point in exploring both possibilities in the Tower of Hanoi task, and do not conclusively prove them. However, our simulations showed various possible ways of capturing human data in this task, and led to some interesting insights: (1) One point is that both directions are viable ways for skill learning. (2) Furthermore, given the common experimental settings of this task, top-down learning may be a more apt way of capturing human performance in this task, (at least partially) due to the fact that this task is a high-level, highly structured, cognitive skill task—it involved much high-level, explicit thinking to begin with, and the task instructions further explicitly encouraged goal recursion. (3) However, even though top-down learning is predominant in this task, bottom-up learning is likely to be present also, as demonstrated by our simulations. (4) Comparing this work with Sun et al (2001), while this work shows the possibility of bottom-up learning in predominantly top-down learning situations, Sun et al (2001) showed the necessity of bottom-up learning in low-level skill domains. Thus the two papers are complementary: Together they demonstrate the importance and the general prevalence of bottom-up learning. (5) In terms of the model, the simulation of this task (along with other high-level tasks) expands the scope of CLARION—from low-level to high-level cognitive domains. (6) Thus, CLARION provides a general mechanistic (i.e., process-based) account of both implicit and explicit processes and in particular both top-down and bottom-up learning.

Finally, there are always more than one way of capturing human data. These finer distinctions need to be made at a theoretical level, which CLARION helps to bring out, and then they should be tested experimentally (through human experiments) in future work. Highlighting the issue is one of the major points of this work.

Appendix

Details of Bottom-Up Learning

The RER algorithm is as follows:

1. Update the rule statistics (to be explained later).

2. Check the current criterion for rule extraction, generalization, and specialization:

2.1. If the result is successful according to the current criterion, and there is no rule matching that state and that action, then perform *extraction* of a new rule: state $\longrightarrow$ action. Add the extracted rule to the rule network.

2.2. If the result is unsuccessful according to the current criterion, revise all the matching rules through *specialization*:

2.2.1. Remove the matching rules from the rule network.

2.2.2. Add the revised (specialized) versions of the rules into the rule network.

2.3. If the result is successful according to the current criterion, then generalize the matching rules through *generalization*:

2.3.1. Remove the matching rules from the rule network.

2.3.2. Add the generalized rules to the rule network.

Let us discuss the details of the operations used and the criteria measuring whether a result is successful or not.

At each step, we examine the following information: (x, y, r, a), where x is the state before action a is performed, y is the new state after an action a is performed, and r is the reinforcement received after action a. Based on that, we update (in Step 1 of the above algorithm) the positive and negative match counts for each rule condition and each of its minor variations (i.e., the rule condition plus/minus one possible value in one of the input dimensions), denoted as C, with regard to the action a performed: that is, $PM_a(C)$ (i.e., Positive Match, which equals the number of times that an input matches the condition C, action a is performed, and the result is positive) and $NM_a(C)$ (i.e., Negative Match, which equals the number of times that an input matches the condition C, action a is performed, and the result is negative).

Positivity or negativity (for updating PM and NM) may be determined by the following inequality:

$$\gamma \max_b Q(y, b) + r - Q(x, a) > threshold_{RER}$$

which indicates whether or not the action chosen according to a rule is reasonably good (Sun and Peterson 1998).

Each statistic is updated with the following formulas:

- $PM := PM + 1$ when the positivity criterion is met;
- $NM := NM + 1$ when the positivity criterion is not met.

- At the end of each episode, they are discounted: $PM := PM * 0.90$ and $NM := NM * 0.90$. The results are time-weighted statistics, which are useful in nonstationary situations.

Based on these statistics, we may calculate the information gain measure:

$$IG(A,B) = log_2 \frac{PM_a(A) + c_1}{PM_a(A) + NM_a(A) + c_2} - log_2 \frac{PM_a(B) + c_1}{PM_a(B) + NM_a(B) + c_2}$$

where A and B are two different rule conditions that lead to the same action a, and c_1 and c_2 are two constants representing the prior (the default values are $c_1 = 1, c_2 = 2$). The measure compares essentially the percentage of positive matches under different conditions A and B (with the Laplace estimator; Lavrac and Dzeroski 1994). If A can improve the percentage to a certain degree over B, then A is considered better than B. In the following algorithm, if a rule is better compared with the corresponding match-all rule (i.e, the rule with the same action but with the condition that matches all possible input states), then the rule is considered successful.

We decide on whether or not to extract a rule based on the positivity criterion, which measures whether the current step is successful or not, fully determined by the current step (x, y, r, a):

- *Extraction*: if the current step is positive (according to the current positivity criterion; e.g., $r + \gamma e(y) - Q(x,a) > threshold_{RER}$, where a is the action performed in state x and y is the resulting new state and if there is no rule that covers this step in the top level, set up a rule $C \longrightarrow a$, where C specifies the values of all the input dimensions exactly as in the current state x and a is the action performed at the current step.

On the other hand, *generalization* and *specialization* operators is based on the afore-mentioned information gain measure. Generalization amounts to adding an additional value to one input dimension in the condition of a rule, so that the rule will have more opportunities of matching inputs, and specialization amounts to removing one value from one input dimension in the condition of a rule, so that it will have less opportunities of matching inputs. Here are the detailed descriptions of these two operators:

- *Generalization*: if $IG(C, all) > threshold1$ and $\max_{C'} IG(C', C) \geq 0$, where C is the current condition of a matching rule, *all* refers to the corresponding match-all rule (with regard to the same action specified by the rule), and C' is a modified condition such that C' = C plus one value (i.e., C' has one more value in one of the input dimensions) [**that is, if the current**

rule is successful and a generalized condition is potentially better], then set $C'' = argmax_{C'} IG(C', C)$ as the new (generalized) condition of the rule. Reset all the rule statistics. Any rule covered by the generalized rule will be placed in its children list. [12]

- *Specialization*: if $IG(C, all) < threshold2$ and $\max_{C'} IG(C', C) > 0$, where C is the current condition of a matching rule, *all* refers to the corresponding match-all rule (with regard to the same action specified by the rule), and C' is a modified condition such that $C' = C$ minus one value (i.e., C' has one less value in one of the input dimensions) [**that is, if the current rule is unsuccessful, but a specialized condition is better**], then set $C'' = argmax_{C'} IG(C', C)$ as the new (specialized) condition of the rule. [13] Reset all the rule statistics. Restore those rules in the children list of the original rule that are not covered by the specialized rule and the other existing rules. If specializing the condition makes it impossible for a rule to match any input state, delete the rule.

Combining the Two Levels

With probability P_{RER}, at a step, if there is at least one RER rule indicating a proper action in the current state, we use the outcome from that rule set; otherwise, we use the outcome of the bottom level (which is always available). With probability P_{FR}, if there is at least one fixed rule indicating a proper action in the current state, we use the outcome from that rule set; otherwise, we use the outcome of the bottom level (which is always available). With the remaining probability $P_{BL} = 1 - P_{RER} - P_{FR}$, we use the outcome of the bottom level. There exists some psychological evidence for such intermittent use of rules; see, e.g., VanLehn (1991) and Anderson (1993).

When using the outcome from the RER rule set at the top level, we use the action suggested by an RER rule randomly selected among all the RER rules matching the current input state. The same goes for FRs. When using the outcome from the bottom level, we use the stochastic decision process based on the Boltzmann distribution for selecting an action:

$$p(a|x) = \frac{e^{Q(x,a)/\alpha}}{\sum_i e^{Q(x,a_i)/\alpha}} \tag{5}$$

[12] The children list of a rule is created to keep aside and make inactive those rules that are more specific (thus fully covered) by the current rule. It is useful because if later on the rule is deleted or specialized, some or all of those rules on its children list may be reactivated if they are no longer covered.

[13] Clearly, we should have $threshold2 \leq threshold1$ to avoid oscillation.

where x is the current state, a is an action, and α controls the degree of randomness (temperature) of the decision-making process.

The cross-level selection parameters can be determined through the following "probability matching", where sr stands for success rate and β is a weighting parameter:

$$P_{BL} = \frac{\beta_{BL} * sr_{BL}}{\beta_{BL} * sr_{BL} + \beta_{RER} * sr_{RER} + \beta_{FR} * sr_{FR}}$$

and similarly for P_{RER} and P_{FR}.

We set $sr_{BL} = \frac{c_3 + \Sigma PM}{c_4 + \Sigma PM + \Sigma NM}$, where the summations are over all the input/action pairs matching the bottom-level decisions; c_3 and c_4 represents the prior, where the default is $c_3 = 1$ and $c_4 = 2$. Other sr's can be determined similarly.

Further Details of Simulation Setups

Fixed Rules in Top-Down Simulation 1. The extended set of FRs was as follows. The condition of each FR included 3 components: (1) working memory items (pop-flag and push-flag), (2) external inputs (disks on pegs), and (3) goals. The actions of the FRs might be: pushing a goal onto the GS, popping a goal, or actually moving a disk from one peg to another. Function LOC indicated the peg on which a disk was located, function SIZE indicated the size of a disk, and DSIZE, SUBTOWER, FOCAL-DISK etc. referred to the corresponding components of the top goal on the GS.

> If the current state is the initial state and DSIZE > 0, then push a new goal for moving a subtower of size DSIZE-1 to the spare peg and for moving the disk of size DSIZE to its target peg.
>
> If the current state is the initial state and DSIZE $= 0$, then make a move of FOCAL-DISK to its target peg.
>
> If the current state is not the initial state, SIZE (RecentlyMovedDisk) $\geq$ DSIZE1, and POP-FLAG = true[14], then pop a goal.
>
> If the current state is not the initial state and SIZE (RecentlyMovedDisk) $<$ DSIZE1, then set POP-FLAG = false and set PUSH-FLAG = true
>
> If the current state is not the initial state, PUSH-FLAG = true[15], LOC(SUBTOWER) = TO, and LOC(FOCAL-DISK) $\neq$ TO1, then move FOCAL-DISK to its target peg.

[14]This means that it is in the stage of popping goals.

[15]This means that it is in the process of pushing goals

If the current state is not the initial state, PUSH-FLAG = true, LOC(SUBTOWER) = TO, and LOC(FOCAL-DISK) = TO1[16], then pop the current goal.

If the current state is not the initial state, PUSH-FLAG = true, LOC(SUBTOWER) $\neq$ TO, and DSIZE > 0, then push a new goal for moving a new subtower of DSIZE-1 to the peg other than the current peg and the target peg of SUBTOWER, and for moving the disk of size DSIZE to the target peg of SUBTOWER.

If the current state is not the initial state, PUSH-FLAG = true, LOC(SUBTOWER) $\neq$ TO, and DSIZE=0, then make a move of FOCAL-DISK to its target peg.

Note that, each time, before these FRs at the top level were invoked, POP-FLAG was set to *true*, and PUSH-FLAG to $false$.

At each step, a selection was made regarding whether the top level or the bottom level was to make an action decision. When the top level was chosen to make an action decision, these top-level rules could keep running until a physical movement of a disk was achieved. Then, at the next step, a new selection was made regarding whether the top level or the bottom level got to make an action decision next. On the other hand, if the bottom level was selected, the bottom level would suggest an actual movement of a disk. After the movement of a disk, a new step began, in which a new selection was made regarding whether the top level or the bottom level got to make an action decision next.

RT Equations and Parameters. In calculating response times, we used the following RT equations:

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

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

where RT was the total response time, PT was the perceptual time, AT was the motor action time, and DT was the decision time. The following parameter settings were adopted: $PT_{BL} = 500ms, DT_{BL} = 350ms, AT_{BL} = AT_{TL} = 1500ms, PT_{TL} = PT_{BL} + 135ms = 635ms, DT_{TL}$ = number-of-pushes x push-time + number-of-pops x pop-time (where push-time = 2500 ms and pop-time = 500 ms).

At first glance, it may seem that there are too many free parameters. However, notice the fact that many parameters are additive, and thus many of them can be collapsed into a single parameter. In fact, RTs are determined by a linear function of number-of-pops and number-of-pushes. There are only *three* free parameters.

[16]This means that the current goal has been achieved.

References

M. Ahlum-Heath and F. DiVesta, (1986). The effect of conscious controlled verbalization of a cognitive strategy on transfer in problem solving. *Memory and Cognition.* 14, 281-285.

H. Aizenstein, A. MacDonald, V. Stenger, R. Nebes, J. Larson, S. Ursu, and C. Carter, (2000). Complementary category learning systems identified using event-related functional MRI. *Journal of Cognitive Neuroscience.* 12 (6), 977-987.

E. Altmann and J. Trafton, (2002). Memory for goals: An activation-based model. *Cognitive Science*, 26, 39-83.

J. R. Anderson, (1983). *The Architecture of Cognition.* Harvard University Press, Cambridge, MA

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.

Y. Anzai and H. Simon, (1979). The theory of learning by doing. *Psychological Review*, 86, 124-140.

A. Baddeley, (1986). *Working Memory.* Oxford University Press, New York.

D. Berry, (1983). Metacognitive experience and transfer of logical reasoning. *Quarterly Journal of Experimental Psychology*, 35A, 39-49.

K. Bowers, G. Regehr, C. Balthazard, and Parker, (1990). Intuition in the context of discovery. *Cognitive Psychology.* 22. 72-110.

J. Bruner, J. Goodnow, and J. Austin, (1956). *A Study of Thinking.* Wiley, New York.

J. Busemeyer and I. Myung, (1992). An adaptive approach to human decision making: learning theory, decision theory, and human performance. *Journal of Experimental Psychology: General*, 121 (2), 177-194.

A. Clark and A. Karmiloff-Smith, (1993). The cognizer's innards: a psychological and philosophical perspective on the development of thought. *Mind and Language.* 8 (4), 487-519.

A. Cleeremans, (1994). Attention and awareness in sequence learning. *Proc. of Cogntiive Science Society Annual Conference*, 330-335.

A. Cleeremans and J. McClelland, (1991). Learning the structure of event sequences. *Journal of Experimental Psychology: General.* 120, 235-253.

Z. Dienes, (1992). Connectionist and memory-array models of artificial grammar learning. *Cognitive*

Science. 16. 41-79.

M. Erickson and J. Kruschke, (1998). Rules and exemplars in category learning. *Journal of Experimental Psychology: General*, 127, 107-120.

D. Fum and F. Missier, (2001). Adaptive selection of problem solving strategies. *Proceedings of Cognitive Science Society Annual Conference*, 313-319. Lawrence Erlbaum Associates, Mahwah, NJ.

R. Gagne and E. Smith, (1962). A study of the effects of verbalization on problem solving. *Journal of Experimental Psychology*, 63, 12-18.

G. Gunzelmann and J. R. Anderson, (2003). Problem solving: Increased planning with practice. *Cognitive Systems Research*, 4 (1), 57-76

E. Hunt and M. Lansman, (1986). Unified model of attention and problem solving. *Psychological Review*. 93 (4), 446-461.

A. Karmiloff-Smith, (1986). From meta-processes to conscious access: evidence from children's metalinguistic and repair data. *Cognition*. 23. 95-147.

N. Lavrac and S. Dzeroski, (1994). *Inductive Logic Programming*. Ellis Horword, New York.

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. Maclin and J. Shavlik, (1994). Incorporating advice into agents that learn from reinforcements. *Proc.of AAAI-94*. Morgan Kaufmann, San Meteo, CA.

J. Mandler, (1992). How to build a baby. *Psychology Review*. 99, 4. 587-604.

R. Mathews, R. Buss, W. Stanley, F. Blanchard-Fields, J. Cho, and B. Druhan, (1989). Role of implicit and explicit processes in learning from examples: a synergistic effect. *Journal of Experimental Psychology: Learning, Memory and Cognition*. 15, 1083-1100.

J. McClelland, B. McNaughton and R. O'Reilly, (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102 (3), 419-457.

D. Meyer and D. Kieras, (1997). A computational theory of executive cognitive processes and human multiple-task performance: part 1, basic mechanisms. *Psychological Review*. 104 (1), 3-65.

M. Moscovitch and C. Umilta, (1991). Conscious and unconscious aspects of memory. In: *Perspectives on Cognitive Neuroscience*. Oxford University Press, New York.

R. Nosofsky, T. Palmeri, and S. McKinley, (1994). Rule-plus-exception model of classification learning.

Psychological Review. 101 (1), 53-79.

E. Owen and J. Sweller, (1985). What do students learn while solving mathematics problems? *Journal of Experimental Psychology*, 77 (3), 272-284.

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.

A. Reber and R. Allen (1978). Analogy and abstraction strategies in synthetic grammar learning: a functionalist interpretation. *Cognition.* 6, 189-221.

G. Regehr and L. Brooks, (1993). Perceptual manifestations of an analytic structure: The priority of holistic individuation. *Journal of Experimental Psychology: General.*

P. Rosenbloom, J. Laird, and A. Newell, (1993). *The SOAR papers: Research on Integrated Intelligence.* MIT Press, Cambridge, MA.

D. Ruiz, (1986). Learning and problem solving: What is learned while solving the Tower of Hanoi? Doctoral thesis, Stanford University, Stanford, CA.

D. Rumelhart, J. McClelland and the PDP Research Group, (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, MIT Press, Cambridge, MA.

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

W. Schneider and W. Oliver (1991), An instructable connectionist/control architecture. In: K. VanLehn (ed.), *Architectures for Intelligence*, Erlbaum, Hillsdale, NJ.

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

H. Simon, (1975). The functional equivalence of problem solving skills. *Cognitive Psychology*, 7, 268-288.

P. Smolensky, (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11 (1), 1-74.

W. Stanley, R. Mathews, R. Buss, and S. Kotler-Cope, (1989). Insight without awareness: on the interaction of verbalization, instruction and practice in a simulated process control task. *Quarterly*

Journal of Experimental Psychology. 41A (3), 553-577.

R. Sun, (1992). On variable binding in connectionist networks, *Connection Science*, Vol.4, No.2, pp.93-124.

R. Sun, (1994). *Integrating Rules and Connectionism for Robust Commonsense Reasoning.* John Wiley and Sons, New York, NY.

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

R. Sun, (1997). Learning, action, and consciousness: a hybrid approach towards modeling consciousness. *Neural Networks*, special issue on consciousness. 10 (7), pp.1317-1331.

R. Sun, (1999). Accounting for the computational basis of consciousness: a connectionist approach. *Consciousness and Cognition*, Vol.8, 529-565.

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

R. Sun and T. Peterson, (1998). Autonomous learning of sequential tasks: experiments and analyses. *IEEE Transactions on Neural Networks*, Vol.9, No.6, pp.1217-1234.

R. Sun, E. Merrill, and T. Peterson, (1998). A bottom-up model of skill learning. *Proc.of 20th Cognitive Science Society Conference*, pp.1037-1042, 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*, 25 (2), 203-244.

R. Sun, P. Slusarz and C. Terry, (2003). The interaction between the implicit and explicit processes: A dual process approach. Submitted for publication.

K. VanLehn, (1991). Impasse-free learning. Rule acquisition events in the discovery of problem-solving strategies. *Cognitive Science.* 15, 1-47.

K. VanLehn, (1995). Cognitive skill acquisition. *Annual Review of Psychology*, Vol.47, J. Spence, J. Darly, and D. Foss (eds.) Annual Reviews Inc, Palo Alto, CA.

C. Watkins, (1989). *Learning with Delayed Rewards.* Ph.D Thesis, Cambridge University, Cambridge, UK.

D. Willingham, M. Nissen, and P. Bullemer, (1989). On the development of procedural knowledge. *Journal of Experimental Psychology: Learning, Memory, and Cognition.* 15, 1047-1060.

E. Wisniewski and D. Medin, (1994). On the interaction of data and theory in concept learning. *Cognitive Science.* Vol.18, 221-281.

J. Zhang and D. Norman, (1994). Representations in distributed cognitive tasks. *Cognitive Science*, 18, 87-122.