

CLARION: The Interaction of Implicit and Explicit Processes

-

-

Ron Sun

Dichotomies:

— the distinction between procedural knowledge and declarative knowledge (e.g., Anderson 1982, Fitts and Posner, 1967; Keil, 1989; Sun, 1995).

— the distinction between the conceptual and the subconceptual (Smolensky 1988)

— the distinction of analytic and intuitive thinking (Dreyfus and Dreyfus 1989)

— implicit memory vs. explicit memory

— implicit learning vs. explicit learning

— unconscious vs. conscious perception

→ duality of the mind

The Key: The interaction of the Implicit and the Explicit:

— implicit learning research: the lack of explicit knowledge

— implicit memory research: the lack of explicit recall

— but what about the interaction between implicit and explicit processes?

complex, multi-faceted, and continuous

→ needs due research attention

Evidence of Interaction: numerous

— verbalization: Stanley et al. (1989),
Sun et al. (2001)

— explicit search: Berry and Broadbent
(1988)

— degree of awareness: Willingham et
al. (1989), Nissen and Bullemer (1987)

— dual task (aware vs. unaware): Cur-
ran and Keele (1993)

etc. etc.

Bottom-Up Learning:

— The distinction between top-down learning vs. bottom-up learning

—→ bottom-up learning: a key neglected aspect

Indications of bottom-up learning

— *Implicit learning*

Berry and Broadbent (1988), Willingham et al. (1992), Reber (1989), Stanley et al. (1989)

— *Implicit memory*

Schacter (1987)

— *Instrumental conditioning*

— *Developmental psychology*

Karmiloff-Smith (1986), Keil (1987), Mandler (1993)

Usefulness of Explicit Knowledge:

- Expedite learning
- Facilitate transfer
- Facilitate communication among cognitive agents

→ the **synergy** effect

See Sun et al. (1998, 2001) (in terms of synergy in both machine learning and human learning).

also Mathews et al. (1989)

Many Issues:

- How can we best capture implicit processes computationally? How can we best capture explicit processes computationally?
- How do the two types of knowledge develop along side each other and influence each other's development?
- Is bottom-up learning (or parallel learning) possible, besides top-down learning? How can they (bottom-up learning, top-down learning, and parallel learning) be realized computationally?
- How do the two types of acquired knowledge interact during skilled performance? What is the impact of that interaction on performance? How do we capture such impact computationally?

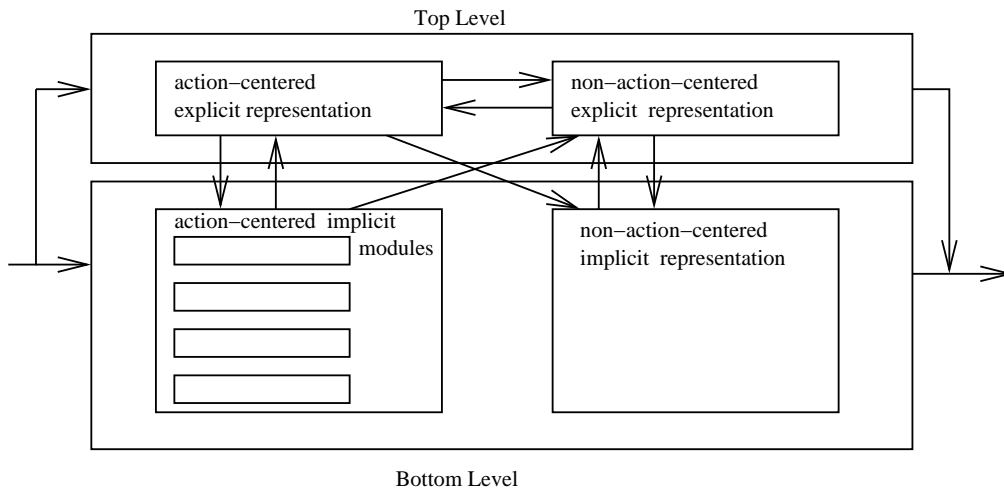
More interesting kinds of cognitive models:

- Autonomous learning
 - Learning both implicit and explicit knowledge simultaneously
 - Bottom-up learning
 - Using both implicit and explicit knowledge in performance
- integrated hybrid architectures for cognitive agents

The CLARION Model: Basic Postulates

- Representational difference: The two levels employ two different types of representations and thus have different degrees of accessibility.
- Learning difference: Different learning methods are used for the two levels.
- Bottom-up and top-down learning: When there is no sufficient a priori knowledge available, the learning is bottom-up.
- Action-centered vs. non-action-centered representation: At each level, there are both action-centered and non-action-centered representations present.
- Separability: The relative contributions of the two levels (in learning or performance) can be experimentally manipulated.

The CLARION Model: Basic Architecture



Pseudo-code

1. Observe the current state x .
2. Compute in the bottom level the Q-value of each of the possible actions (a_i 's) associated with the current state x : $Q(x, a_1)$, $Q(x, a_2)$,, $Q(x, a_n)$.
3. Find out all the possible actions (b_1, b_2, \dots, b_m) at the top level, based on the the information x and other available information (which goes up from the bottom level) and the rules in place at the top level.
4. Choose an appropriate action a , considering the values of a_i 's and b_j 's (which are sent down from the top 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 the *Q-Learning-Backpropagation* algorithm, based on the feedback information.
7. Update the top level using the *Rule-Extraction-Refinement* algorithm.
8. Go back to Step 1.

Bottom level: Q-values and Q-Learning.

– Compute $Q(x, a)$ for state x and action a

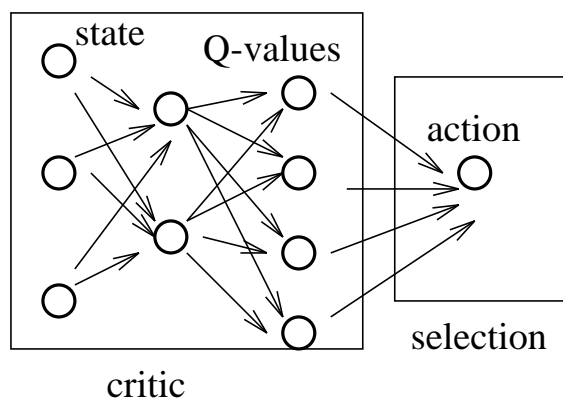
– Stochastic decision making:

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

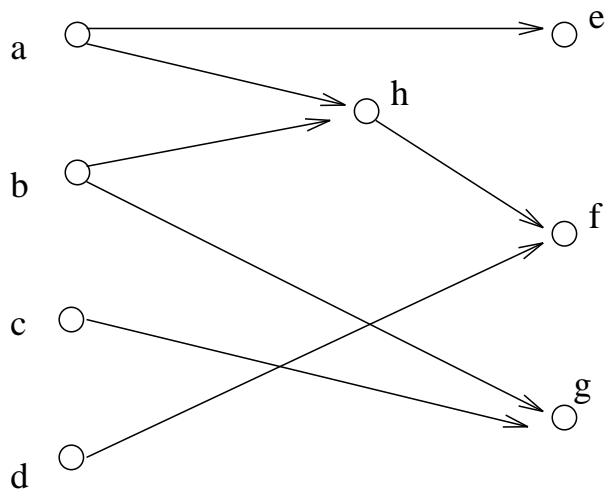
where α controls the degree of randomness

– Sequentiality: need to “plan” — the temporal credit assignment problem.

– Update $\Delta Q(x, a) = \alpha(r + \gamma e(y) - Q(x, a))$ where $e(y) = \max_a Q(y, a)$,



The Top Level: Connectionist Rule Network



- direct translation
- logical operations
- variable binding

The Rule-Extraction-Refinement (RER) Algorithm.

— The basic idea:

If an action decided by the bottom level is successful then extract a rule. Then, in subsequent interactions with the world, refine the extracted rule: if the outcome is successful, generalize (“expand”) the condition of the rule; if the outcome is not successful, then specialize (“shrink”) the condition of the rule.

Rule Learning Details

— based on an *information gain* measure

$PM_a(C)$ (i.e., Positive Match)

$NM_a(C)$ (i.e., Negative Match)

positivity/negativity: determined by

$\max_b Q(y, b) - Q(x, a) + r > threshold,$

$$IG(A, B) = \log_2 \frac{PM_a(A) + 1}{PM_a(A) + NM_a(A) + 2} - \log_2 \frac{PM_a(B) + 1}{PM_a(B) + NM_a(B) + 2}$$

- *Construction*: if $r + \gamma e(y) - Q(x, a) > \text{threshold}$, where a is the action performed in state x and y is the resulting new state [**that is, if the current step is successful**], and if there is no rule that covers this step in the top level, set up a rule $C \rightarrow a$, where C specifies the values of all the input dimensions exactly as in x .
- *Expansion*: if $IG(C, \text{all}) > \text{threshold1}$ and $\max_{C'} IG(C', C) \geq 0$, where C is the current condition of an applicable rule, *all* refers to the 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 an expanded condition is potentially better**], then set $C'' = \text{argmax}_{C'} IG(C', C)$ as the new (expanded) condition of the rule.
- *Shrinking*: if $IG(C, \text{all}) < \text{threshold2}$ and $\max_{C'} IG(C', C) > 0$, where C is the current condition of an applicable rule, *all* refers to the 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 shrunk condition is better**], then set $C'' = \text{argmax}_{C'} IG(C', C)$ as the new (shrunk) condition of the rule.
- *Deletion*: included in *Shrinking*.

The IRL (Independent Rule Learning) Algorithm: without using the bottom level for the initial extraction

1. Either completely randomly or in a particular domain-specific order, rules of various forms are independently generated at the top level
2. Then, these rules are tested through experience using an IG measure
3. Optionally, specialization and generalization may be performed
4. If the IG measure of a rule is below a threshold, then the rule is deleted

One possible positivity measure: $\gamma \max_b Q(y, b) + r - Q(x, a) > threshold_{IRL}$

One possible deletion method: “if $IG(C, random) < threshold_3$, we delete the rule C ”

Specialization methods: “if $IG(C, random) < threshold_5$ and $\max_{C'} IG(C', C) > 0$, we shrink C to $argmax_{C'} IG(C', C)$, where $C' = C$ minus one value”

Generalization methods: “if $IG(C, random) > threshold_4$ and $\max_{C'} IG(C', C) > 0$, we expand C to $argmax_{C'} IG(C', C)$, where $C' = C$ plus one value”

Extracting plans

— data structures.

CSS : n pairs of $(s, p(s))$,

CSS' : m states not covered by CSS .

— the algorithm

Set the current state set $CSS = \{(s_0, 1)\}$ and
 $CSS' = \{\}$

Repeat until the termination conditions are satisfied (e.g., $step > D$)

- For each action u , compute the probabilities of transitioning to each of all the possible next states (for all $s' \in S$) from each of the current states ($s \in CSS$):

$$p(s', s, u) = p(s) * p_{s,s'}(u)$$

- For each action u , compute its estimated utility with respect to each state in CSS :

$$Ut(s, u) = \sum_{s'} p(s', s, u) * \max_v Q(s', v)$$

That is, we calculate the *probabilities* of reaching the goal after performing action u from the current state s .

- For each action u , compute the estimated utility with respect to *all* the states in CSS' :

$$Ut(CSS', u) = \sum_{s \in CSS'} \sum_{s'} p(s) * p_{s,s'}(u) \max_v Q(s', v)$$

- For each state s in CSS, choose the action u_s with the highest utility $Ut(s, u)$:

$$u_s = \operatorname{argmax}_u Ut(s, u)$$

- Choose the best default action u with regard to all the states in CSS' :

$$u = \operatorname{argmax}_{u'} Ut(CSS', u')$$

- Update CSS to contain n states that have the highest n probabilities, i.e., with the highest $p(s')$'s:

$$p(s') = \sum_{s \in CSS} p(s', s, u_s)$$

where u_s is the action chosen for state s .

- Update CSS' to contain m states that have the highest m probabilities calculated as follows, among those states that are not in the new (updated) CSS:

$$p(s') = \sum_{s \in CSS \cup CSS'} p(s', s, u_s)$$

Properties:

— nonconditional plans are special cases of conditional plans.

If we set $n = 0$ and $m > 0$, we then in effect have a nonconditional plan extraction algorithm and the result from the algorithm is a nonconditional plan.

— if we set $m = 0$, then we have a purely conditional plan (with no default action attached).

— branching factor:

start with a small number, and gradually expand by adding to the number of branches, until $p(G) > \delta$ or when a time limit is reached

Properties:

Theorem 1 *Weak completeness I: if there exists a limited-depth conditional solution (i.e., a conditional plan) to a planning problem, then the algorithm will find a conditional solution (provided that we set the goal threshold above 0).*

Theorem 2 *Weak completeness II: if there exists a limited-depth nonconditional solution to a planning problem, then the algorithm will find a conditional plan and a nonconditional plan (if we set the goal threshold above 0).*

Theorem 3 *Strong completeness I: if there exists a limited-depth conditional solution to a planning problem that has a success probability $p > \delta$, then the algorithm will find a (conditional) solution with $p' > \delta$ (if we set the goal threshold to be δ).*

Theorem 4 *Strong completeness II: if there exists a nonconditional solution to a planning problem that has a success probability $p > \delta$, then the algorithm will find a (conditional and/or nonconditional) plan with $p' > \delta$ (if we set the goal threshold to be δ).*

Theorem 5 *Soundness: if the algorithm returns a solution (a conditional or nonconditional plan) with a probability p , then the true success probability of the solution is greater than or equal to p .*

Combining two levels.

— Through averaging of the two levels:

$$w_1 * v + w_2 * q$$

where w_1 and w_2 are automatically determined through probability matching.

— Through stochastic selection of one of the two levels:

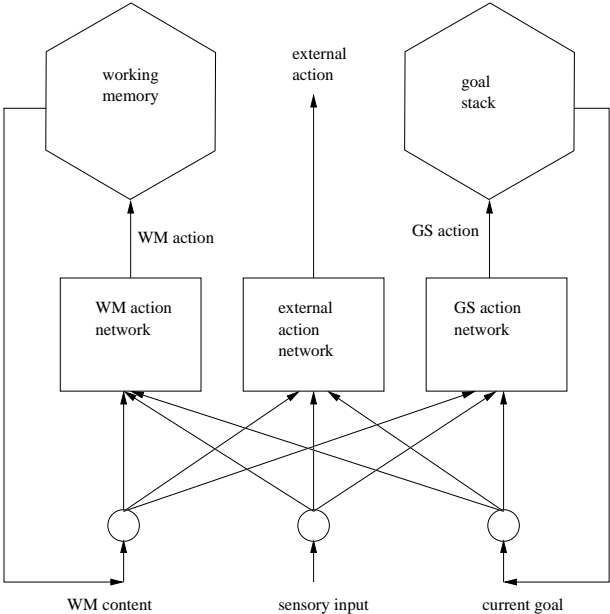
P_1 : top level

P_2 : bottom level

where P_1 and P_2 are automatically determined through probability matching.

Or use fixed weights/probabilities

Other Components:



| entity | evidential support | selection | response time |
|---------|--------------------|-----------|---------------|
| rule | rule support | utility | bla |
| chunk | strength | | bla |
| WM item | activation (bla) | | |
| EM item | | bla | |
| GS item | | | |

rule extraction density

rule extraction probability

Simulation Experiments:

— Process control tasks

Berry and Broadbent (1988),

Stanley et al. (1989),

Dienes and Fahey (1995),

— Serial reaction time tasks

Lewicki et al. (1987),

Curran and Keele (1993)

— Alphabetic arithmetic (letter counting) tasks

Rabinowitz and Goldberg (1995)

— Tower of Hanoi

Gagne and Smith (1962)

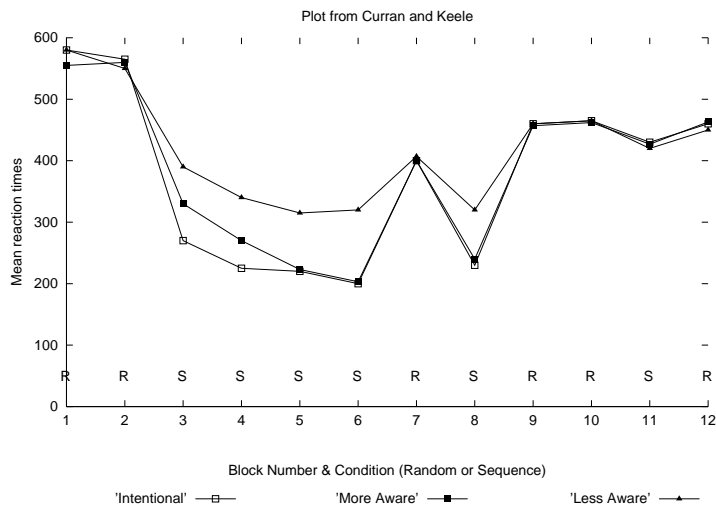
— Minefield navigation

Sun et al. (2001)

Serial Reaction Time Tasks: Curran and Keele (1993)

- a sequence of X marks
- three phases: dual task practice, single task learning, and dual task transfer
- three groups of subjects: “less aware”, “more aware” and “intentional”

Data: Curran and Keele (1993)



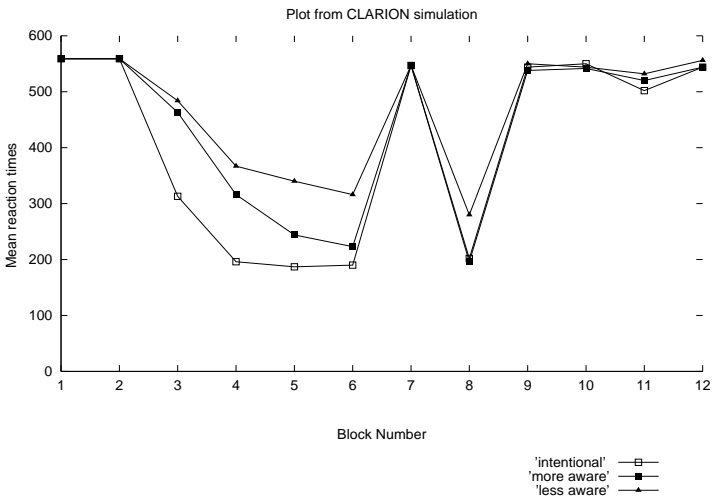
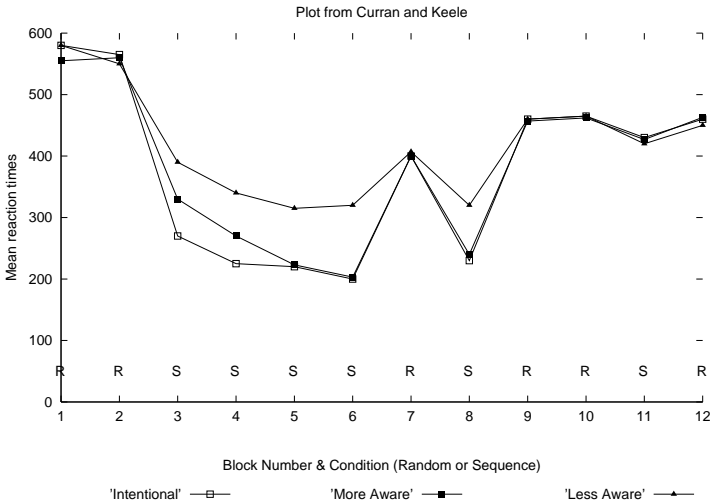
— ANOVA (intentional vs. more aware vs. less aware x sequential vs. random): significant difference across groups during the second phase

— ANOVA (intentional vs. more aware vs. less aware x sequential vs. random): no significant difference across groups during the third phase

Model Setup:

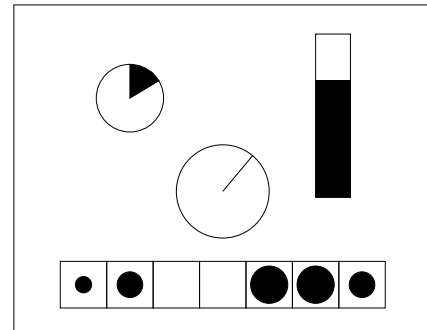
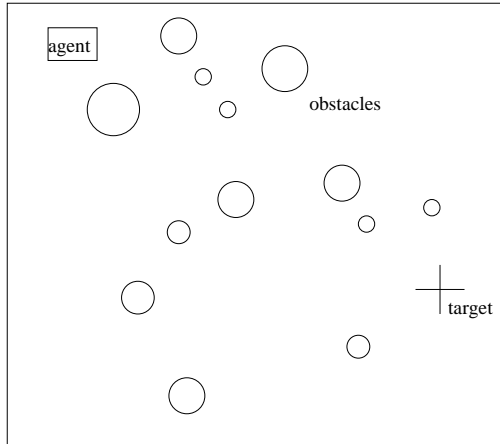
- simplified QBP
- 7 x 6 input units (both primary and secondary tasks) and 5 output units
- RER rule learning
- Three groups: “less aware” —use higher rule learning thresholds; “more aware” — use lower rule learning thresholds; “intentional” — code given knowledge in the top level
- linear transformation: $RT_i = a * e_i + b$
- ANOVA confirmed the match

Simulation:



The Minefield Navigation Task:

Sun et al. (2001)



Five training conditions:

- The standard training condition.
- The verbalization conditions.
- The over-verbalization condition.
- The dual-task condition.
- The transfer conditions.

Their effects

Model Setup with CLARION:

- The effect of the dual task: captured by reduced top-level activities (through raised rule learning thresholds)

- The effect of (regular) verbalization: stem from heightened rule learning activities (Stanley et al. 1989) and to a lesser extent, from rehearsing previous episodes.

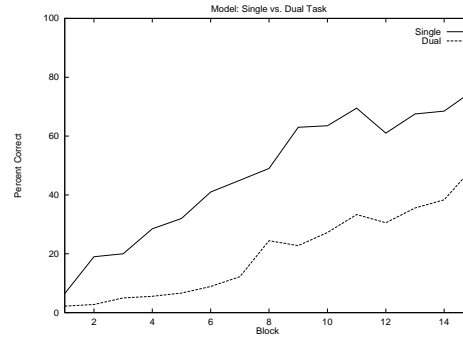
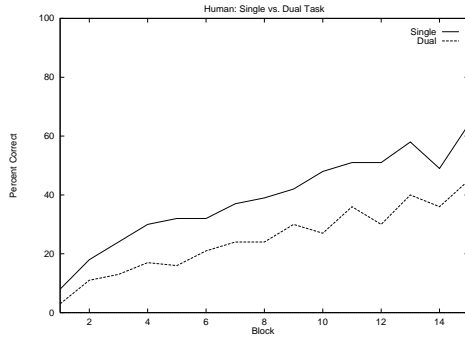
- The effect of over-verbalization: stem from overly heightened rule learning activities

- The model started out with no more a priori knowledge about the task than a typical human subject, so that bottom-up learning can be captured.

- 10 human subjects were compared to 10 model subjects (randomly selected) in each experiment.

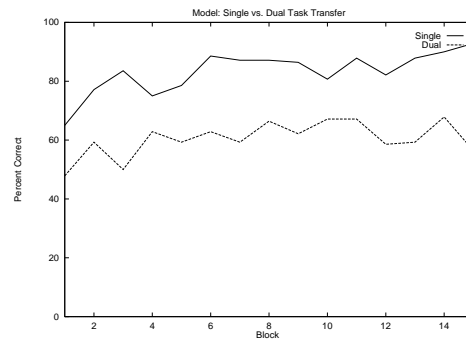
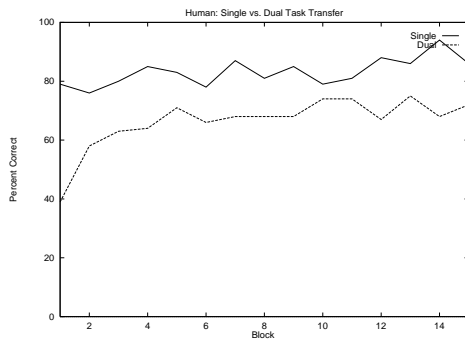
Simulation: The effect of the dual task condition on learning

2x2 ANOVA (human vs. model x single vs. dual task) indicated a significant main effect for single vs. dual task ($p < .01$), but no interaction between groups and task types,



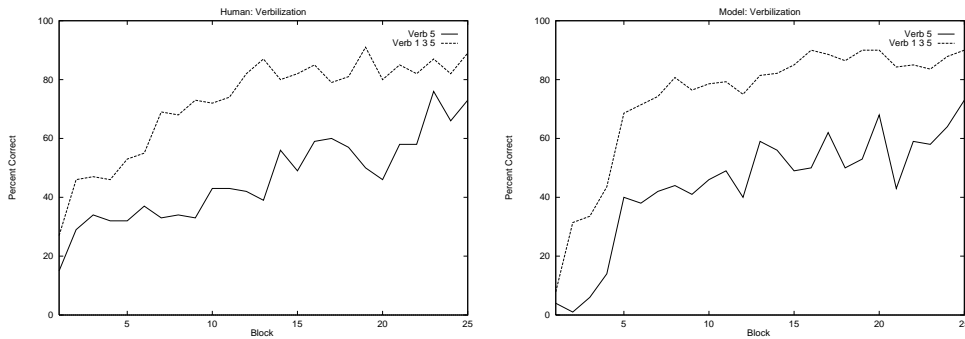
Simulation: The effect of the dual task condition on transfer

2x2 ANOVA (human vs. model x single vs. dual task) revealed a significant main effect of single vs dual task ($p < .05$), and no interaction between groups and task types



Simulation: The effect of verbalization

4 (days) x 2 (human vs. model) x 2 (verbalization vs. no verbalization) ANOVA indicated that both human and model subjects exhibited a significant increase in performance due to verbalization ($p < .01$), but that the difference associated with verbalization for the two groups was not significant.



Concluding Remarks:

— interaction of implicit and explicit processes

— manipulation through verbalization, explicit search, explicit instructions, and so on

— synergy between the two types of processes

— bottom-up learning is an essential way of human skill learning

— the model constitutes a plausible explanation of human learning and provides some interesting interpretations