The Action-Centered Subsystem

Nicholas Wilson, Sebastien Helie, Ron Sun

Cognitive Science, Rensselaer Polytechnic Institute

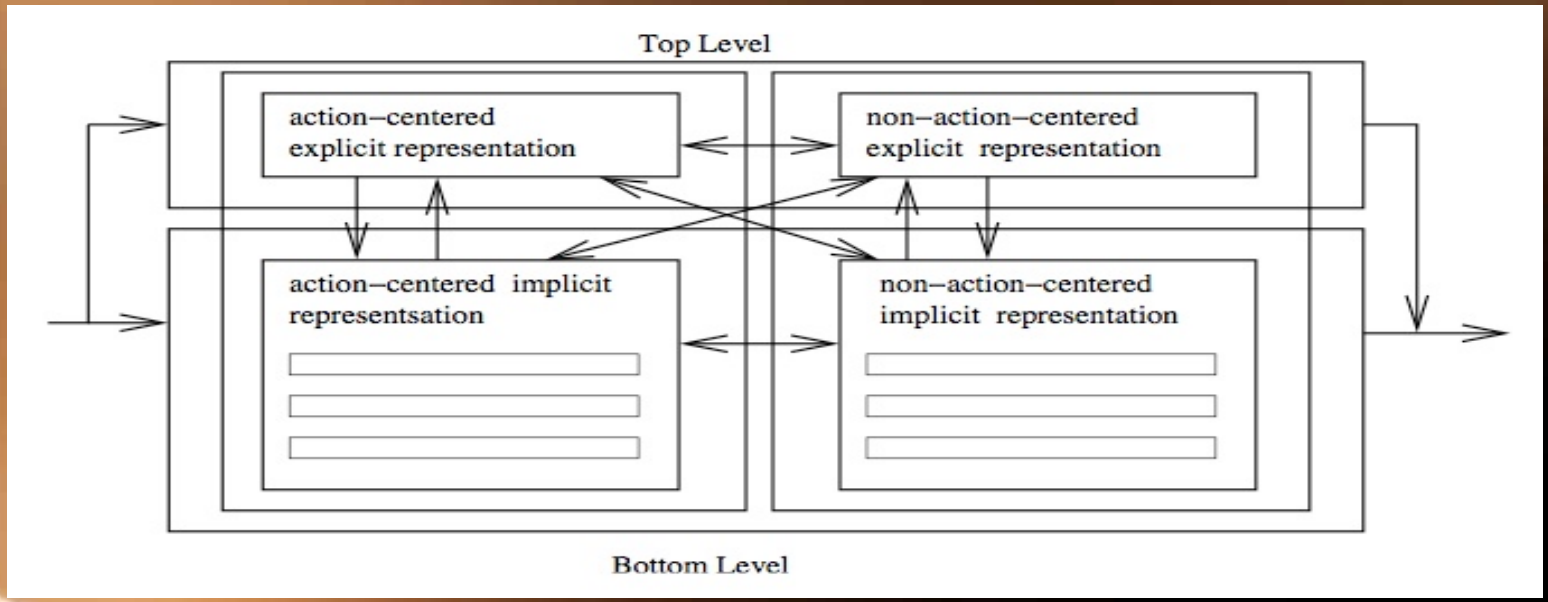
Outline

1. Representation
    1. Bottom-Level Representation
    2. Top-Level Representation
2. Learning
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. Level Integration
4. Working Memory & NACS
5. Simulation Examples
6. Summary

Representation

1. **Representation**
    1. Bottom-Level Representation
    2. Top-Level Representation
2. Learning
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. Level Integration
4. Working Memory & NACS
5. Simulation Examples
6. Summary

Representation



- Bottom Level: Implicit Representation
- Top Level: Explicit Representation

Representation

- Action Decision making process:
  1. Observe the current state.
  2. Compute in the bottom level the “value” of each of the possible actions in the current state. Stochastically choose one action.
  3. Find out all the possible actions at the top level, based on the current state information and the existing rules in place at the top level. Stochastically choose one action.
  4. Choose an appropriate action by stochastically selecting or combining the outcomes of the top level and the bottom level.
  5. Perform the selected action and observe the next state along with any feedback (i.e. reward).
  6. If feedback is received, update the bottom level in accordance with Q-learning (implemented with a backpropogation network).
  7. Update the top level using an appropriate learning algorithm (for constructing, refining, and deleting explicit rules).
  8. Go back to step 1.

Representation

1. Representation
    1. **Bottom-Level Representation**
    2. Top-Level Representation
2. Learning
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. Level Integration
4. Working Memory & NACS
5. Simulation Examples
6. Summary

Representation

- Bottom Level
  - Implicit knowledge:
    - Inaccessible
    - Reactive (fast decision making)
  - Inaccessible nature captured by distributed representations.
  - Backpropogation Neural Networks
    - Representational units are capable of accomplishing tasks but are generally not individually meaningful.
      - Renders distributed representation less accessible.

Representation

- Backpropogation Networks on the bottom level are known as Implicit Decision Networks (IDNs)
- The current state is represented as dimension-value pairs:
  - $(dim_1,val_1)(dim_2,val_2)...(dim_n,val_n)$
    - Each pair corresponds to one input node of the network
  - Three types of inputs:
    - Sensory Input
    - Working Memory Items
    - Goal Structure Items

Representation

- Actions are represented as nodes on the output layer.
  - Three types of actions:
    - Working Memory
    - Goal
    - External
- Each action consists of one or more action dimensions
  - Can be in the form:
    - $(dim_1,val_1)(dim_2,val_2)...(dim_n,val_n)$

Representation

- The level of activation for a node within an Implicit Decision Network is calculated using a sigmoid activation function:

$$o = \frac{1}{1 + e^{-\sum_{i=1}^{n} w_i x_i}}$$

Where x_i is the value of input i, w_i is the weight of input i, and n is the number of input nodes

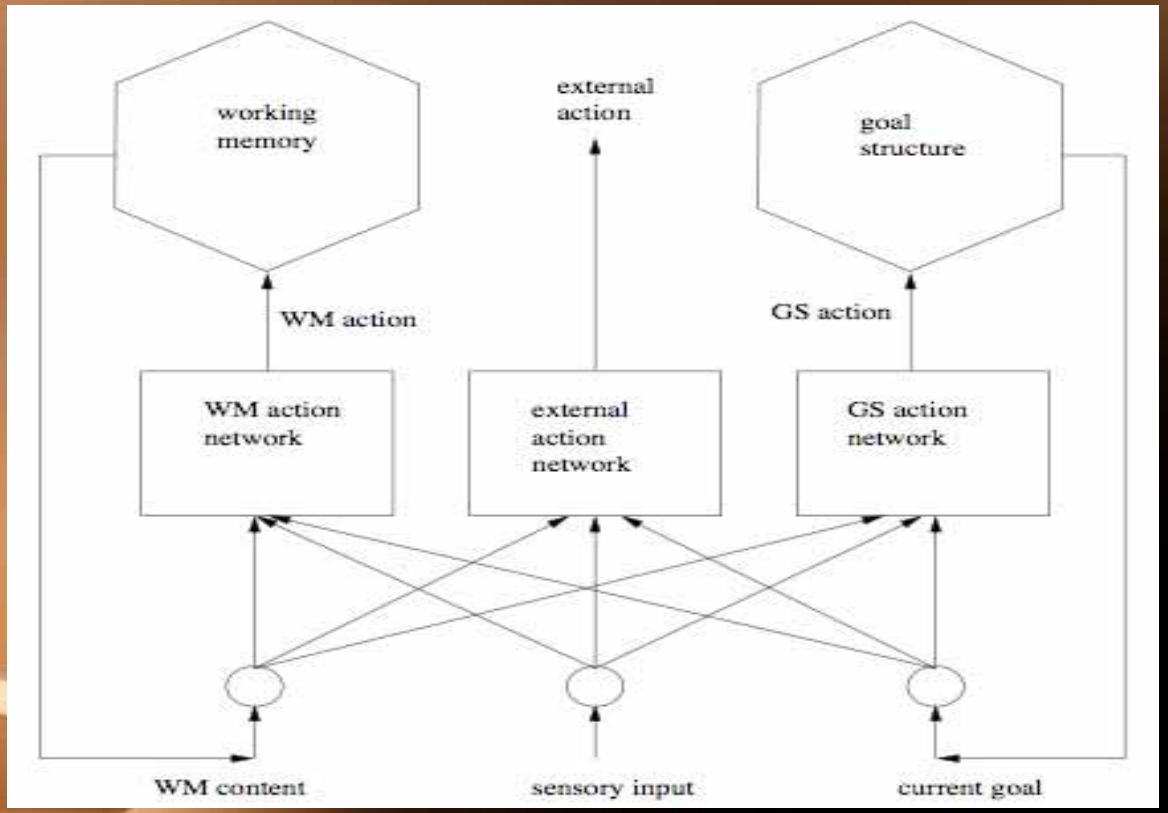
Representation

- An action is chosen based on a Boltzmann distribution of the activations of the output nodes.
- The probability of selecting a particular action *i* in the bottom-level is:

$$p(i \mid x) = \frac{e^{A_i/\tau}}{\sum_j e^{A_j/\tau}}$$

Where A_i is the activation of action *i* and τ is the noise (temperature) parameter

Representation



Representation

Questions?

Representation

1. Representation
    1. Bottom-Level Representation
    2. **Top-Level Representation**
2. Learning
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. Level Integration
4. Working Memory & NACS
5. Simulation Examples
6. Summary

Representation

- Top Level
  - Explicit Rules:
    - Thought process is more accessible
    - Actions are “rationally” deduced
    - Slower than Implicit Knowledge, but also more precise
    - “Condition ---> Action” rule pairings
      - Condition Chunks
      - Action Chunks
  - Rules in the top level come in three forms (more on this later):
    - Extracted and Refined Rules (RER rules)
    - Independent Rules (IRL rules)
    - Fixed Rules (FR rules)

Representation

- Chunks are collections of dimension/value pairings that represent either conditions or actions in the top level.
  - Chunk-id: $(dim_{i1},val_{i1})(dim_{i2},val_{i2})...(dim_{in},val_{in})$
    - e.g., *table-1: (size, large)(color, white)(number-of-legs, 4)*
- Each chunk is represented by a node in the top level
- Each dimension/value pairing is represented by a node in the bottom level
- A top-level rule contains one or more conditions and one action (possibly with multiple dimensions)

Representation

- Action recommendation is based on these factors:
  - Base-Level Activation (BLA)
    - Recency-based value (for determining RTs)
  - Utility
    - Measures the usefulness of a rule based on the cost and benefit of the rule (for selecting rules)
  - In addition to other numerical measures

Representation

- Base-Level Activation (BLA) measures the odds of needing a rule based on the history of its use.
  - Can be used to filter out impertinent rules before further processing
- BLA can be calculated as a recency-based value:

$$B_j^r = iB_j^r + c \times \sum_{l=1}^{n} t_l^{-d}$$

Where t_l is the *lth* use of rule *j* and iB_j is the initial value.
By default c = 2, d = 0.5

Representation

- Rule selection in the top level uses a Boltzmann distribution of the utility of the rule.
- Utility is calculated using the following equation:

$$U_j^r = benefit_j - \nu \times \cos t_j$$

υ Is a scaling factor

Representation*

- Benefit:

$$benefit_j = \frac{c_7 + PM(j)}{c_8 + PM(j) + NM(j)}$$

Where PM(j) = number of positive rule matches and NM(j) = number of negative rule matches. By default $c_7 = 1$, $c_8 = 2$

- Cost:

$$cost_j = \frac{execution - time - of - rule - j}{average - execution - time - of - rules}$$

Values need to be estimated --change text above to remove spaces

Representation

Questions?

Learning

1. Representation
    1. Bottom-Level Representation
    2. Top-Level Representation
2. **Learning**
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. Level Integration
4. Working Memory & NACS
5. Simulation Examples
6. Summary

Learning

- Bottom-Level Learning
  - Uses Backpropagation to perform error correction within the bottom level
  - Three learning methods:
    - Standard Backpropagation
    - Q-Learning
    - Simplified Q-Learning
- Top-Level Rule Learning
  - Three methods:
    - Bottom-up rule extraction and refinement (RER)
    - Independent Rule Learning (IRL)
    - Fixed Rule Learning (FR)

Learning

1. Representation
    1. Bottom-Level Representation
    2. Top-Level Representation
2. Learning
    1. **Bottom-Level Learning**
    2. Top-Level Rule Learning
3. Level Integration
4. Working Memory & NACS
5. Simulation Examples
6. Summary

Learning*

- Standard Backpropagation (for three-layer network)
  - Calculate error in the output layer using:

$$err_i = target(x,a_i) - Q(x,a_i)$$

Where *target*(x,a_i) is the target output for node *i* and $Q(x,a_i)$ is the actual output of node *i*

  - Output weights are updated as follows:

$$\Delta w_{ji} = \alpha x_{ji} \delta_j$$

$$\delta_j = err_j o_j (1 - o_j)$$

Where x_{ji} is the *ith* input of output unit *j* and o_j is the output of unit j

Learning*

- Standard Backpropagation (cont.)
  - Weights are updated in the hidden layer by:

$$\Delta w_{ji} = \alpha \delta_j x_{ji}$$

$$\delta_j = o_j (1 - o_j) \sum_k \delta_k w_{kj}$$

Where x_{ji} is the *ith* input to hidden unit *j*, α is the learning rate, o_j is the output of hidden unit *j*, and *k* denotes all of the units downstream (in the output layer) of hidden unit *j*

Learning

- Q-Learning
  - Reinforcement learning algorithm
  - Updating based on the temporal difference in evaluating the current state and the action chosen
  - Uses backpropagation, except error is calculated in the output layer using:

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

Where $r+\gamma e(y)$ estimates the (discounted) total reinforcement to be received from the current point on.

Learning

- Q-Learning (cont.)
  - Q(x,a) is defined by:

$$Q(x,a) = \max_{a_i : i=1,2,3,\ldots} \left(\sum_{i=0}^{\infty} \gamma^i r_i \right)$$

Where γ is a discount factor, a_i is an action that can be performed at step i, and r_i is the reinforcement received at step i

  - e(y) is calculated using:

$$e(y) = \max_b (Q(y,b))$$

Where y is the new state resulting from action a in state x

Learning

- Simplified Q-Learning
  - Basic form of reinforcement learning
  - Temporal credit assignment is not involved
  - Most useful when immediate feedback is available
  - Error is calculated in the output layer using:

$$err_i = \begin{cases} r - Q(x, a_i) & \text{if } a_i = a \\ 0 & \text{otherwise} \end{cases}$$

Learning

- Context for Reinforcement Learning
  - Sensory Input ---> MS ---> MCS ---> *Reinforcement signal* (to be used in the ACS)
  - Sensory Input ---> Action Output (forming implicit reactive routines within the ACS, by reinforcement learning)
  - In addition to other loops

Learning

Questions?

Learning

1. Representation
    1. Bottom-Level Representation
    2. Top-Level Representation
2. Learning
    1. Bottom-Level Learning
    2. **Top-Level Rule Learning**
3. Level Integration
4. Working Memory & NACS
5. Simulation Examples
6. Summary

Learning

- Top Level Learning
  - Bottom-up rule extraction and refinement (RER)
    - “Condition --> Action” pairs are extracted from the bottom level and can be generalized or specialized as necessary
  - Independent rule learning (IRL)
    - Rules of various forms are independently generated either by random or in a domain-specific order and then refined or deleted as needed
  - Fixed Rules
    - Rules are obtained by pre-endowment, prior knowledge (through prior experiences), or by external sources

Learning

- Rule extraction and refinement (RER)
  - Basic idea of the algorithm:
    - If an action decided by the bottom level is successful then a rule is constructed and added to the top level
    - In subsequent interactions with the world, the rule is refined by considering the outcome of applying the rule:
      - If the outcome is successful, the condition of the rule may be generalized to make it more universal
      - If the outcome is not successful, then the condition of the rule should be made more specific

Learning

- Rule extraction
  - Check the current criterion for rule extraction
    - If the result is successful according to the current rule extraction criterion, and there is no rule matching the current state and action, then perform extraction of a new rule
      - “Condition ---> Action”
      - Add the extracted rule to the action rule store at the top level

Learning

- Rule extraction (cont.)
  - A rule is extracted based on a positivity criterion, for example:

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

  - In cases where feedback is immediately available, the positivity criterion can be simplified to $r > threshold_{RER}$
  - This determines whether or not action a is reasonably good (Sun and Peterson 1997,1998)

Learning

- Refinement
  - Extracted rules (and Independently learned rules) have rule statistics that guide rule refinement
    - Positive Match *PM:=PM+1* when the positivity criterion is met
    - Negative Match *NM:=NM+1* when the positivity criterion is not met
    - At the end of each episode, PM and NM are discounted by multiplying them by .9

Learning

- Refinement (cont.)
  - Based on PM and NM, an information gain measure (IG) may be calculated:

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

Where *A* and *B* are two different rule conditions that lead to the same action *a*, c_1 and c_2 are constants (1 and 2 respectively by default)

  - Measure essentially compares the percentages of positive matches under different conditions
    - If *A* can improve the percentage to a certain degree over *B*, then *A* is considered better than *B*

Learning

- Generalization
  - Check the current criterion for Generalization
    - If the result is successful according to the current generalization criterion then generalize the rules matching the current state and action
      - Remove these rules from the rule store
      - Add the generalized versions of these rules to the rule store at the top level

Learning

- Generalization (cont.)
  - A rule can be generalized using the information gain measure:
    - If $IG(C,all)>threshold_1$ and $max_{C'}IG(C',C)\geq 0$, then set $C''=argmax_{C'}(IG(C',C))$ as the new (generalized) condition of the rule
      - Where C is the current condition of the rule, all is the match-all rule, and C' is a modified condition such that $C'=C$ plus one value
    - Reset all the rule statistics

Learning

- Specialization
  - Check the current criterion for Specialization
    - If the result is unsuccessful according to the current specialization criterion then revise all the rules matching the current state and action
      - Remove the rules from the rule store
      - Add the revised (specialized) rules into the rule store at the top level

Learning

- Specialization (cont.)
  - A rule can be specialized using the information gain measure:
    - If $IG(C,all) < threshold_2$ and $max_{C'} IG(C',C) > 0$, then set $C'' = argmax_{C'}(IG(C',C))$ as the new (specialized) condition of the rule
      - Where C is the current state condition of the rule, all is the match-all rule, C' is a modified condition such that $C'=C$ minus one value
        - If *any dimension in C has no value left after specialization then the rule is deleted*
    - Reset all the rule statistics

Learning

- Example
  - SRT Task (Curran & Keele, 1993)
    - Repeating sequence of X marks each in 1 of 4 possible positions
    - Subjects learn to predict new positions on the basis of preceding positions
      - Learn the sequential relations embedded in the sequence
        - Leads to faster responding

Learning

- Example (cont.)
  - Learning (by iterative weight updating) in the bottom level promotes implicit knowledge formation
  - Resulting weights specify a function relating previous positions (input) to current position (output)
  - Acquired sequential knowledge at the bottom level can lead to the extraction of explicit knowledge at the top level

Learning

- Example (cont.)
  - Initial extraction step creates a rule that corresponds to the current input and output (as determined by the bottom level)
  - Generalization adds more possible values to the condition of the rule so that the rule may have more chances of matching new input
  - Specialization adds constraints to the rule (by removing possible matching values) to make the rule less likely to match new input
  - Applicability of these steps determined by the IG measure

Learning

- Example (cont.)
  - Suppose sequence is:

    1 2 3 2 3 4
  - Initially extracted rule may be:

    1 2 3 2 3 --> 4
  - Generalization may lead to a simplified rule:

    * 2 3 2 3 --> 4 (where * stands for “don’t care”)

    * 2 * 2 3 --> 4

    and so on
  - Continued generalizations and specializations are likely to happen, as determined by the IG measure (which is in turn determined by the performance of the rule)

Learning (RER)

Questions?

Learning

- Independent Rule Learning (IRL)
  - A variation of rule extraction and refinement
  - The bottom level is not used for initial rule extraction
  - Rules are generated either randomly or through a domain-specific order.
    - Then these rules are tested through experience using the IG measure
      - If the rule IG measure is below a threshold, then the rule is deleted.

Learning

- Independent Rule Learning (cont.)
  - Positivity criterion can be based on information from the bottom level (similar to RER), for example:

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

  - Positivity criterion can also be based on information from external sources (such as immediate reinforcement)

Learning

- Independent Rule Learning (cont.)
  - One possible information gain measure for IRL rule testing is:
    - If $IG(C,random) < threshold_3$ then delete the rule
    - This is equivalent to:

$$\text{If } IG(C) = \log_2\left(\frac{PM_a(C) + c_5}{PM(C) + NM(C) + c_6}\right) < threshold_4 \text{ then delete rule C}$$

Learning

- Fixed Rules (FR)
  - Externally given or acquired by pre-endowment
  - Enables top-down learning (assimilation)
    - Fixed Rules in the top level act in a supervisory capacity to guide proceduralization in the bottom level.
  - Can represent more than just propositional structures
    - More complex interaction and action sequences between conditions and actions
      - Schemas (Arbib 1980, Dretcher 1989)
      - Abstract behaviors (Mataric 2001)

Learning

Questions?

Level Integration

1. Representation
    1. Bottom-Level Representation
    2. Top-Level Representation
2. Learning
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. **Level Integration**
4. Working Memory & NACS
5. Simulation Examples
6. Summary

Level Integration

- Several Methods:
  - Stochastic Selection
  - Combination
    - Bottom-up Rectification
    - Top-down Guidance
- Assumes one bottom-level network and one top-level rule group
  - Details on coordinating multiple bottom-level networks and top-level rule groups can be found in the technical specifications

Level Integration

- Stochastic Selection
  - At each step, the probability of using any given rule set is:
    - P_{RER} = Probability of using RER rule set
    - P_{IRL} = Probability of using IRL rule set
    - P_{FR} = Probability of using FR rule set
  - The probability of using the bottom level is:
    - $P_{BL} = 1 - P_{RER} - P_{IRL} - P_{FR}$

Level Integration

- Deterministic selection of either the top level or the bottom level is a special case of stochastic selection
  - Selection probabilities in this case may be chosen by the Meta-Cognitive Subsystem (discussed later)
- Selection probabilities may be:
  - Fixed (pre-set/constant)
  - Variable

Level Integration

- Variable selection probabilities are calculated using "probability matching" as follows:

$$P_{BL} = \frac{\beta_{BL} \times sr_{BL}}{\Phi}$$

$$P_{RER} = \frac{\beta_{RER} \times sr_{RER}}{\Phi}$$

$$P_{IRL} = \frac{\beta_{IRL} \times sr_{IRL}}{\Phi}$$

$$P_{FR} = \frac{\beta_{FR} \times sr_{FR}}{\Phi}$$

Where *sr* stands for success rate β is a weighting parameter and $\Phi = \beta_{BL} \times sr_{BL} + \beta_{RER} \times sr_{RER} + \beta_{IRL} \times sr_{IRL} + \beta_{FR} \times sr_{FR}$

Level Integration

- Combination
  - Combining activation from the top level with Q-values in the bottom level
  - Then selecting an action based on the combined values using a Boltzmann distribution
    - Positive conclusions reached in the top level can add to action recommendations in the bottom level
    - Negative conclusions reached in the top level can veto actions in the bottom level

Level Integration

- Bottom-up Rectification:
  - Bottom-level outcome is sent to the top level
  - The top level utilizes and rectifies outcome in the bottom level with the knowledge in the top level
    - One possibility: weighted sum
  - Likely to happen in reasoning situations (Nisbett and Wilson 1977)

Level Integration

- Top-down Guidance:
  - Top-level outcome is sent down to the bottom level
  - The bottom level utilizes the outcome of the top level, along with its own knowledge, in making action-decisions
    - One possibility: weighted sum
  - Most likely happens in skilled learning and skilled performance

Level Integration

Questions?

Overview

- Action Decision making process:
  1. Observe the current state.
  2. Compute in the bottom level the “value” of each of the possible actions in the current state. Stochastically choose one action.
  3. Find out all the possible actions at the top level, based on the current state information and the existing rules in place at the top level. Stochastically choose one action.
  4. Choose an appropriate action by stochastically selecting or combining the outcomes of the top level and the bottom level.
  5. Perform the selected action and observe the next state along with any feedback (i.e. reward).
  6. If feedback is received, update the bottom level in accordance with Q-learning (implemented with a backpropogation network).
  7. Update the top level using an appropriate learning algorithm (for constructing, refining, and deleting explicit rules).
  8. Go back to step 1.

Working Memory

1. Representation
    1. Bottom-Level Representation
    2. Top-Level Representation
2. Learning
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. Level Integration
4. **Working Memory & NACS**
5. Simulation Examples
6. Summary

Working Memory

- For storing memory on a temporary basis
  - Has the express purpose of facilitating subsequent action decision-making and/or reasoning
- Working memory involves:
  - Action-directed (“deliberate”) encoding of information
    - As opposed to automatic encoding
  - Gradual fading of information
  - Action-directed re-encoding (“refreshing”) of information
  - Limited storage capacity

Working Memory

- May be naturally divided into multiple sensory related sections:
  - Visuospatial information
  - Auditory/verbal information
  - Other types of information
- Each section consists of a certain number of slots
  - Each can hold the content of a chunk
- Capacity and chunk size are limited
  - as determined by domain characteristics

Working Memory

- Working memory actions may:
  - Add an item into working memory
    - *Set i:* the content of a chunk is set to working memory slot *i*
    - *Set {i}:* the content of multiple chunks are set to multiple slots in working memory
  - Remove an item from working memory
    - *Reset i:* the content of the *ith* working memory slot is removed
    - *Reset-all-WM*: removes all items from working memory
    - *Do-nothing*
- Working Memory actions can be performed by the top level alone or by a combination of the two levels

Working Memory

- Base-level activation
  - Determines how long past information should be kept around (when no reset action)
  - Recency-based:

$$B_i^w = iB_i^w + c \times \sum_{l=1}^{n} t_l^{-d}$$

where *i* indicates an item in working memory, *l* indicates the *lth* setting of that item, t_l is the time since the *lth* setting, and *iB* is the initial value of *B* (*c* and *d* are constants)

Working Memory

- If base-level activation of working memory item *i* is above a threshold:

$$B_i^w > threshold_{WM}$$

- Then the working memory item is used as input to the bottom level and the top level

Working Memory & NACS

- Working memory is also used to transmit information between the action-centered and non-action-centered subsystems (discussed later)
- Copying from the non-action-centered subsystem is minimally necessary for storing information into working memory
  - Due to the following considerations:
    - Must be able to retrieve and hold working memory items from long-term memory
    - Must be able to extract information in the current state, which is also recorded in the non-action-centered subsystem

Working Memory

Questions?

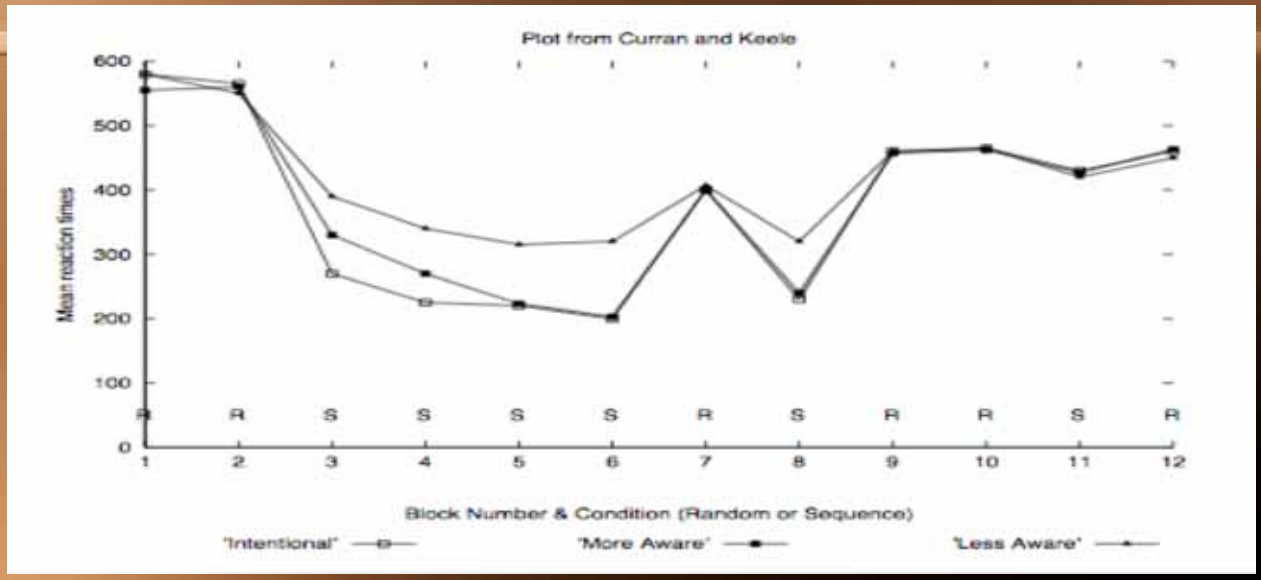
Simulation Examples

1. Representation
    1. Bottom-Level Representation
    2. Top-Level Representation
2. Learning
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. Level Integration
4. Working Memory & NACS
5. **Simulation Examples**
6. Summary

Simulation Examples

- Serial reaction time task (Curran & Keele 1993)
  - A sequence of X marks
  - Two phases:
    - Single task learning
    - Dual task transfer
  - Three groups of subjects:
    - Less aware
    - More aware
    - Intentional

Simulation Examples

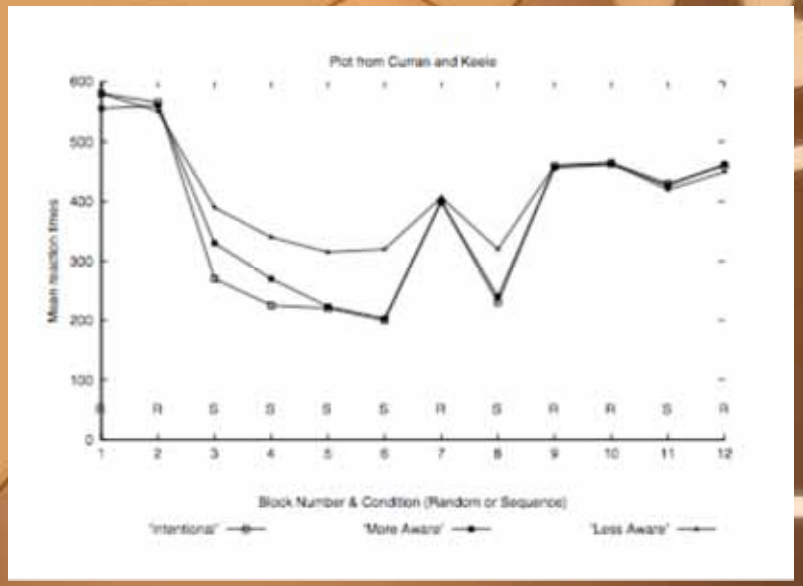


- 3 (intentional vs. more aware vs. less aware) X 2 (sequential vs. random) ANOVA:
  - significant difference across groups during the STL phase
  - No difference across groups during the DTL phase

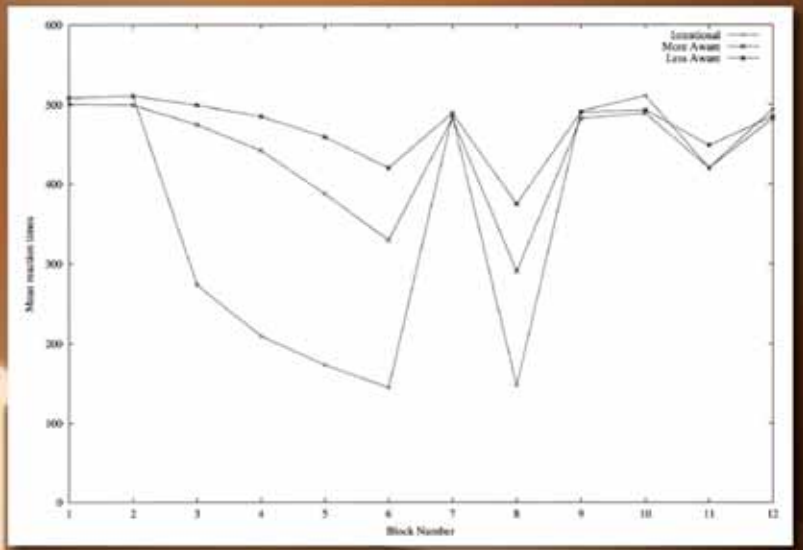
Simulation Examples

- Model Setup
  - Simplified Q-backpropagation learning in the bottom level
  - 7x6 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 data pattern

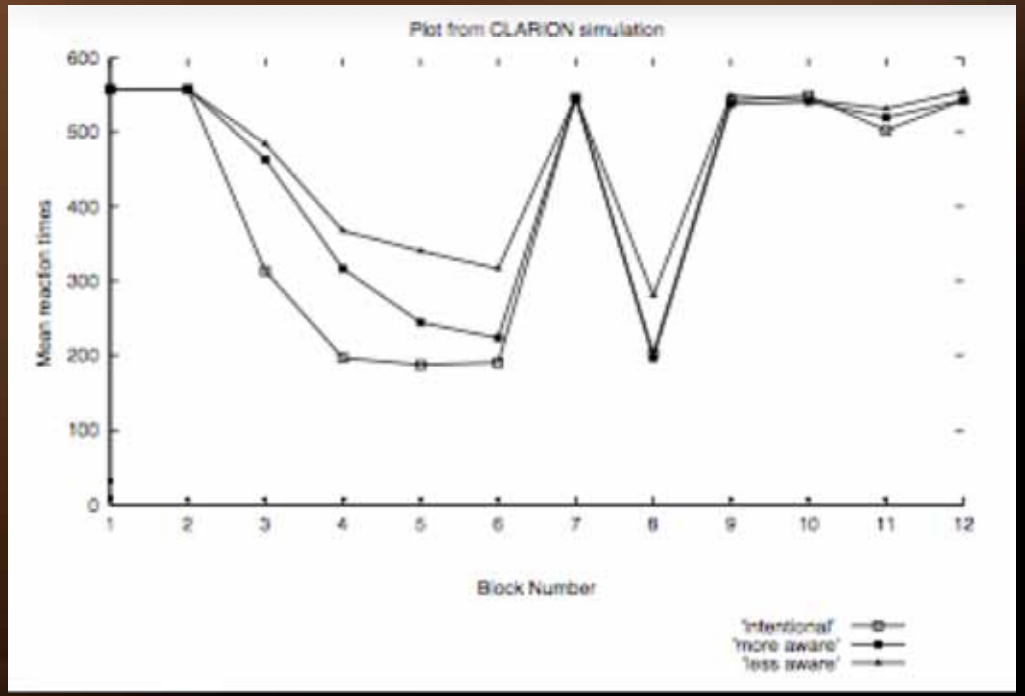
Simulation Examples



Plot from Curran & Keele



Plot from ACT-R Simulation



Plot from CLARION Simulation

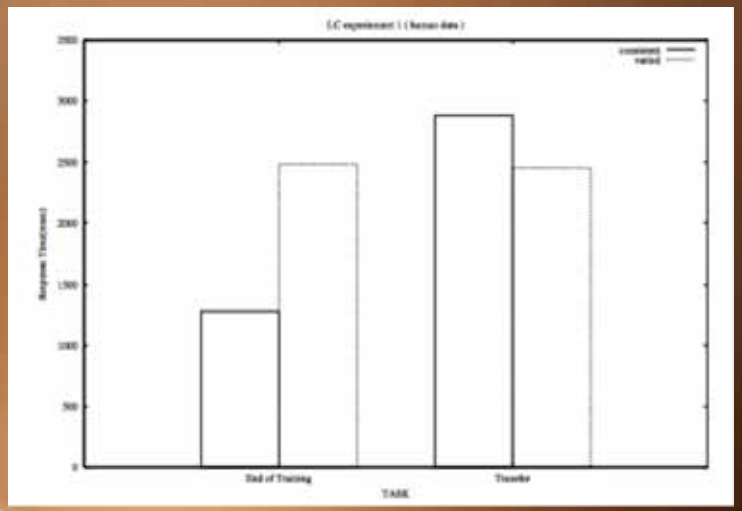
Simulation Examples

- Letter counting task (Rabinowitz & Goldberg 1995)
  - Experiment 1
    - *letter1 + number = letter2*
    - The consistent group:
      - 36 blocks of training (the same 12 addition problems in each)
    - The varied group:
      - 6 blocks of training (the same 72 addition problems in each)
    - Transfer
      - 12 new addition problems (repeated 3 times)

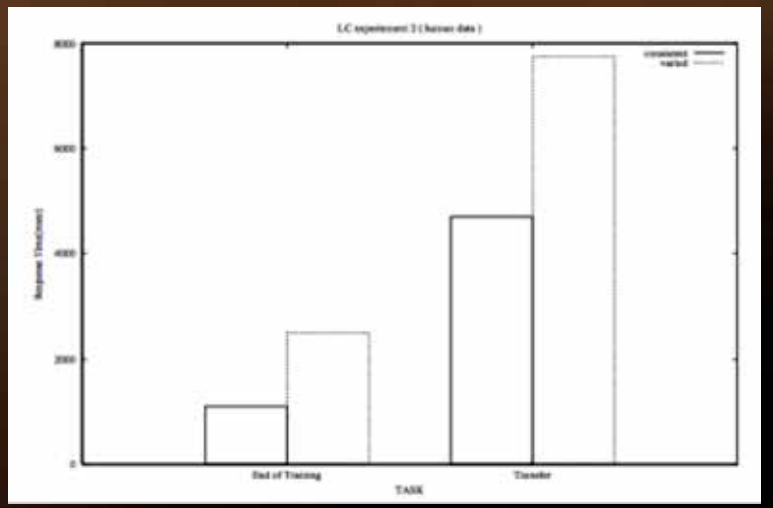
Simulation Examples

- Letter counting task (cont.)
  - Experiment 2
    - Same training
    - Transfer:
      - 12 subtraction problems (repeated 3 times)
      - *letter1 - number = letter2 (reverse of addition problems)*

Simulation Examples



- Experiment 1



- Experiment 2

Simulation Examples

- Model Setup
  - Simplified Q-backpropagation learning in the bottom level IDNs
  - 35 input units, 26 output units, and 30 hidden units
  - Fixed Rules
    - If goal=addition-counting, start-letter=x, number=y, then starting with x, repeat y times counting up
    - If goal=subtraction-counting, start-letter=x, number=y, then starting with x, repeat y times counting down

Simulation Examples

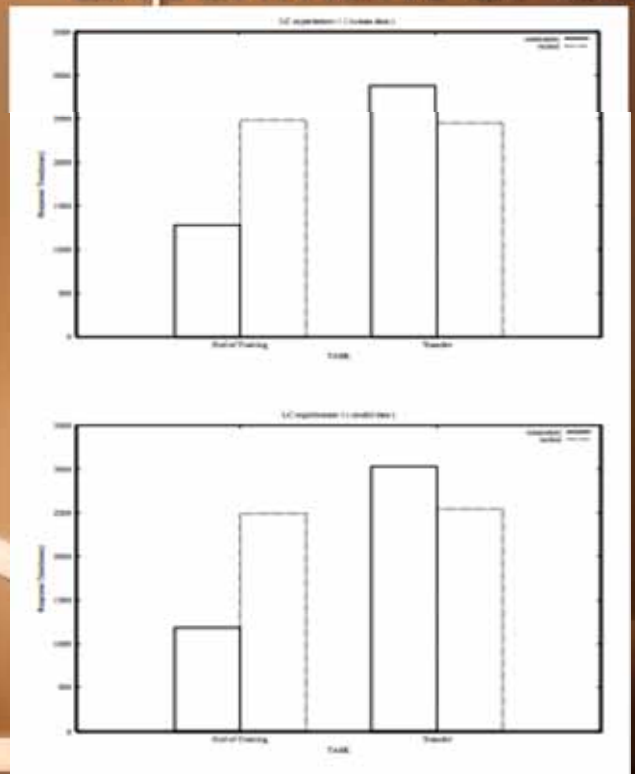
- Model Setup (cont.)
  - Rule utility (for rule selection) and rule base-level activation (for response time)
  - Response Time:

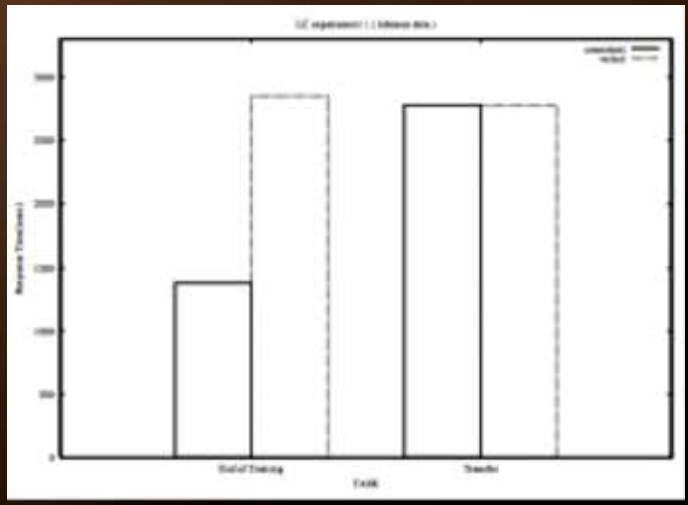
$$DT_{TL} = y \times t_{counting} + c \times \frac{1}{bla(rule)}$$

$$DT_{BL} = constant$$

Simulation Examples

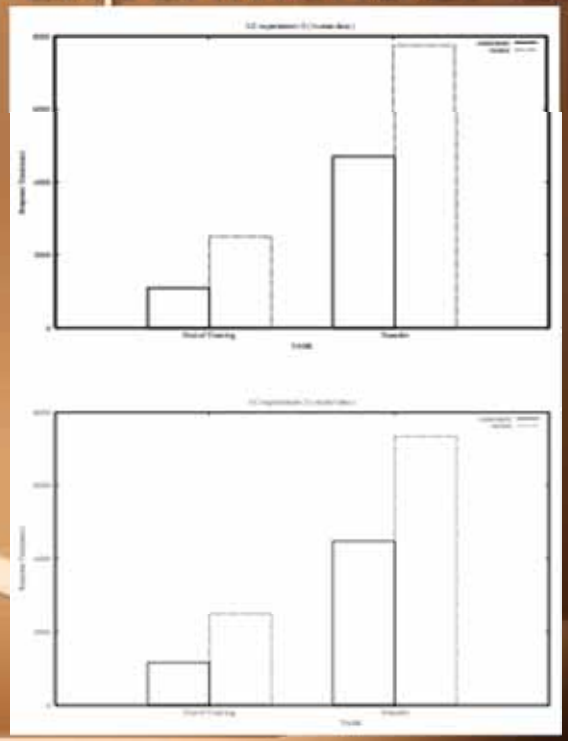
- Experiment 1: CLARION vs. ACT-R (add labels

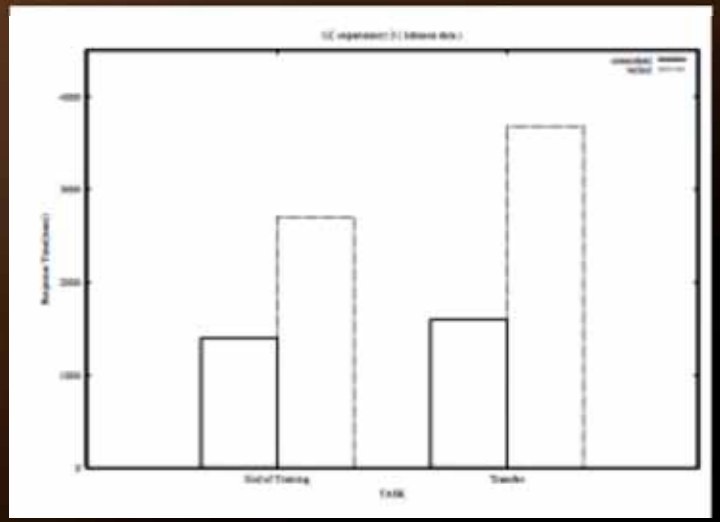




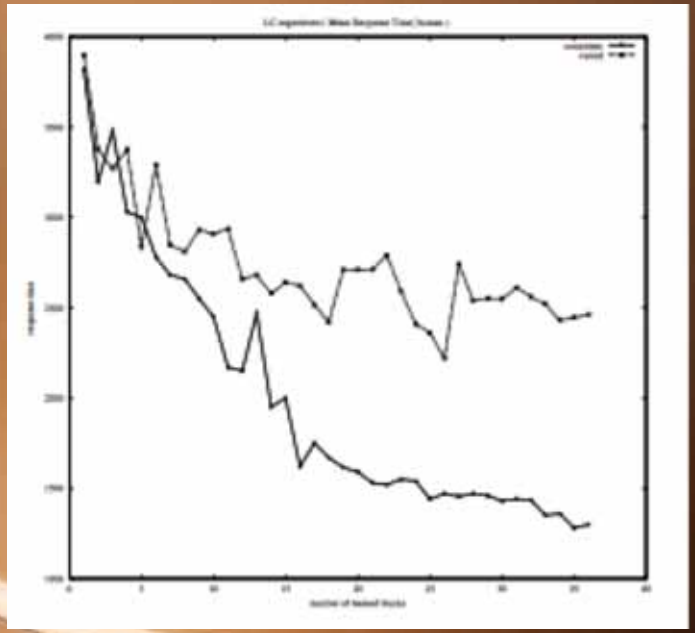
Simulation Examples

- Experiment 2: CLARION vs. ACT-R (cont.)





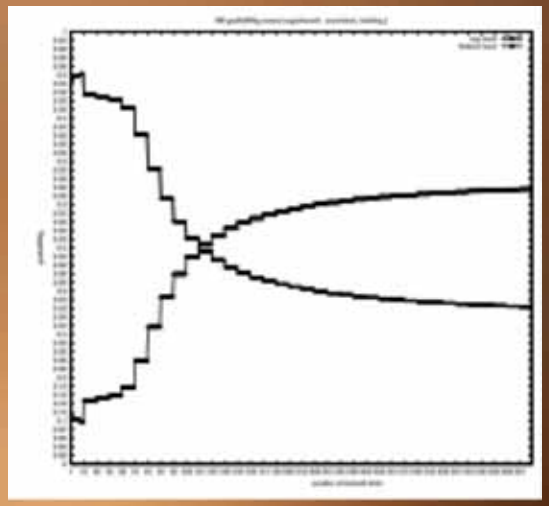
Simulation Examples



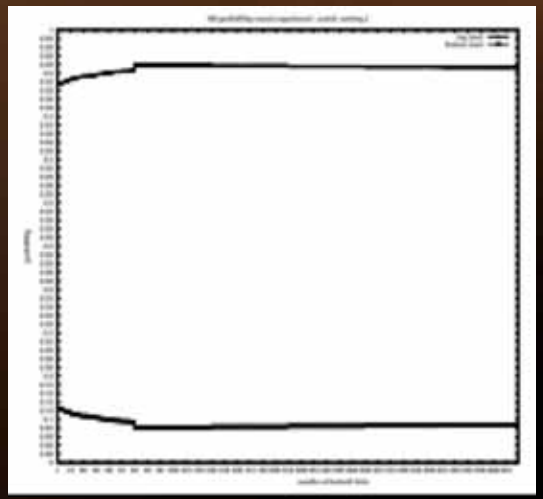
- Learning curve of Rabinowitz and Goldberg (1995)

- Learning curve during the simulation

Simulation Examples



- Combination probability of the consistent group during training in the simulation



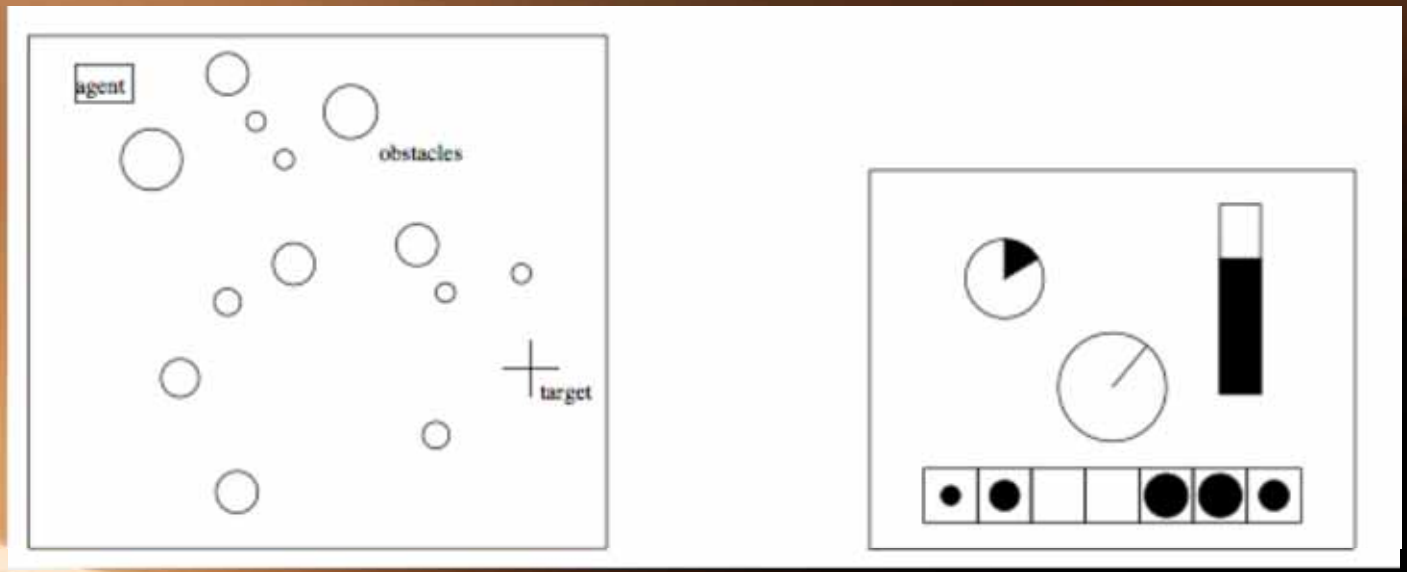
- Combination probability curve of the varied group during training in the simulation

Simulation Examples

Questions?

Simulation Examples

- Minefield navigation task (Sun et al 2001)



Simulation Examples

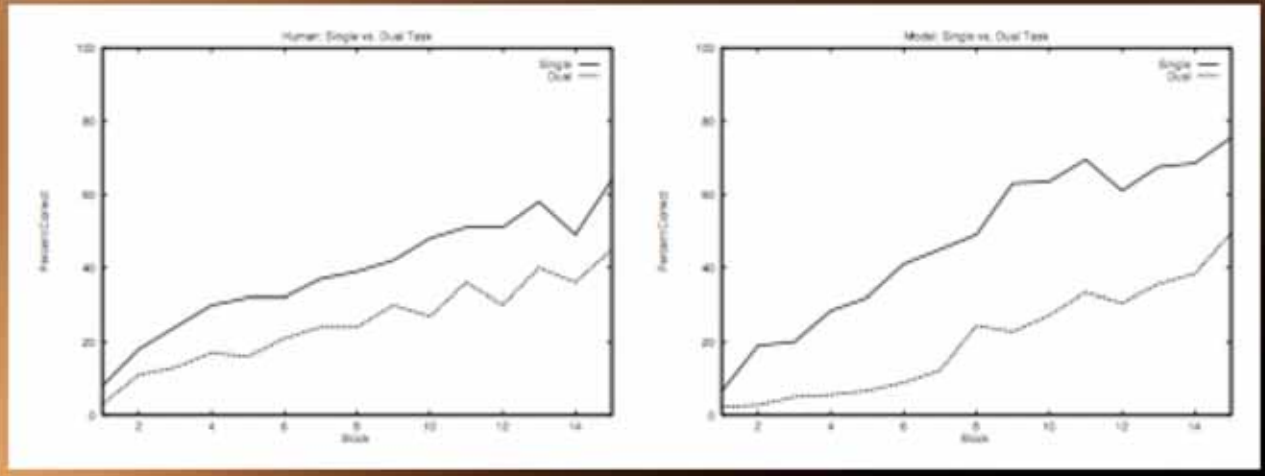
- Four training conditions:
  - Standard training condition
  - Verbalization condition
  - Dual-task condition
  - Transfer conditions

Simulation Examples

- Model Setup
  - 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
  - Model started with no more a priori knowledge about the task than a typical human subjects
  - 10 human subject were compared to 10 model subjects in each experiment

Simulation Examples

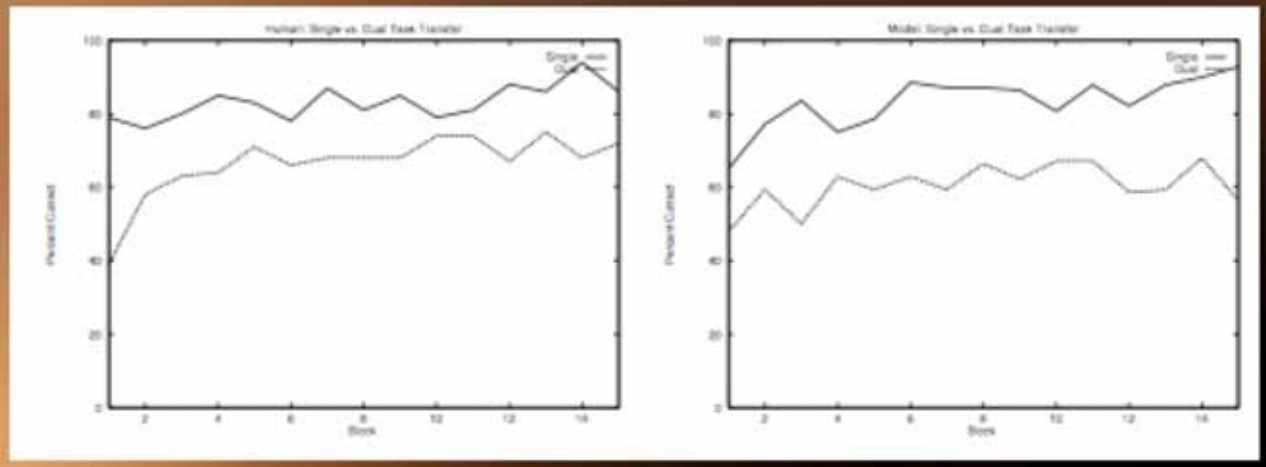
- The effect of the dual task condition on learning:



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

Simulation Examples

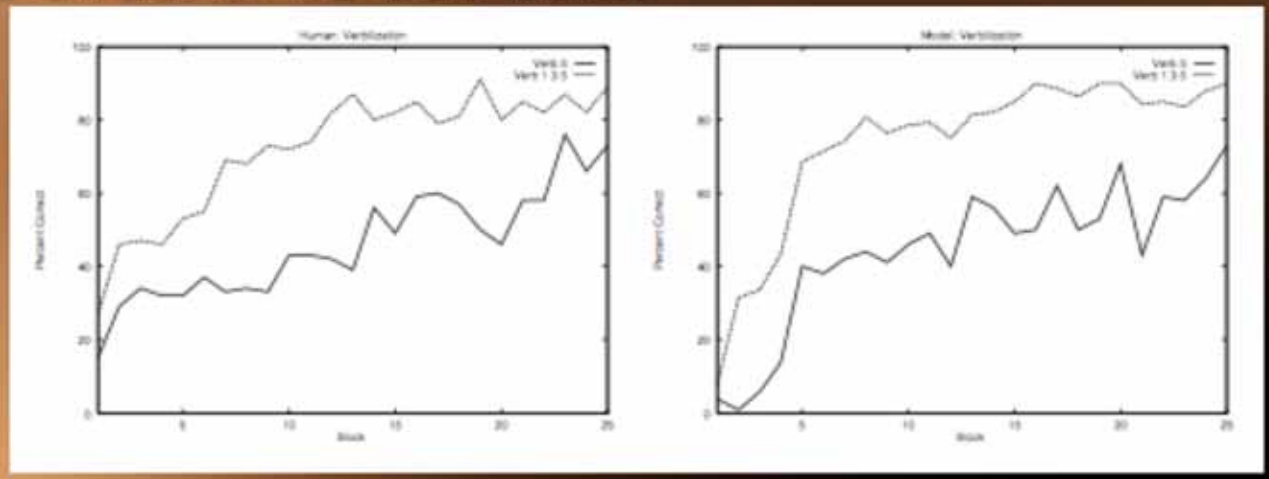
- The effect of the dual task condition on transfer



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

Simulation Examples

- The effect of verbalization



- 4 (days) X 2 (human vs. model) X 2 (verbalization vs. Standard) 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

Simulation Examples

Academic science task (Naveh & Sun 2006)

- Number of authors contributing a certain number of articles follows an inverse power law: a Zipf distribution (Lotka 1926)
- Simon (1957): a simple stochastic process for approximating Lotka's law
  - The probability that a paper will be published by an author who has published *i* articles is a/i^k
- Gilbert (1997) simulated Lotka's law
  - Assumed that authors were non-cognitive and interchangeable; it neglected a host of cognitive phenomena that characterized scientific inquiry (e.g., learning, creativity, evolution of field expertise, etc.)

Simulation Examples

Table 1 Number of authors contributing to *Chemical Abstracts*

Number of Publications	Actual	Simon's estimate	Gilbert's simulation	CLARION simulation
1	3991	4050	4066	3803
2	1059	1160	1175	1228
3	493	522	526	637
4	287	288	302	436
5	184	179	176	245
6	131	120	122	200
7	113	86	93	154
8	85	64	63	163
9	64	49	50	55
10	65	38	45	18
11 or more	419	335	273	145

Simulation Examples

Table 2 Number of authors contributing to *Econometrica*

Number of Publications	Actual	Simon's estimate	Gilbert's simulation	CLARION simulation
1	436	453	458	418
2	107	119	120	135
3	61	51	51	70
4	40	27	27	48
5	14	16	17	27
6	23	11	9	22
7	6	7	7	17
8	11	5	6	18
9	1	4	4	6
10	0	3	2	2
11 or more	22	25	18	16

Simulation Examples

- The CLARION simulation data for the two journals matched the real-world data well
- The CLARION simulation data for the two journals could be fit to the power curve $f(i) = a/i^k$

Journal	a	k	Pearson R	R-square	RMSE
CA	3806	1.63	0.999	0.998	37.62
E	418	1.64	0.999	0.999	4.15

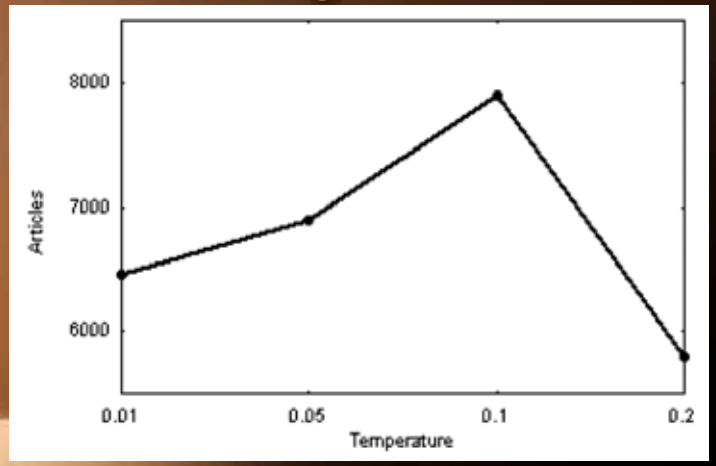
Simulation Examples

- The number of papers per author reflected cognitive processes of authors, as opposed to being based on auxiliary assumptions
- Emergent, not a result of direct attempts to match the human data
- More distance between mechanisms and outcomes, to come up with deeper explanations

Simulation Examples

Varying cognitive parameters:

- Most prolific under a moderately high temperature setting



- Serendipity in scientific discovery!

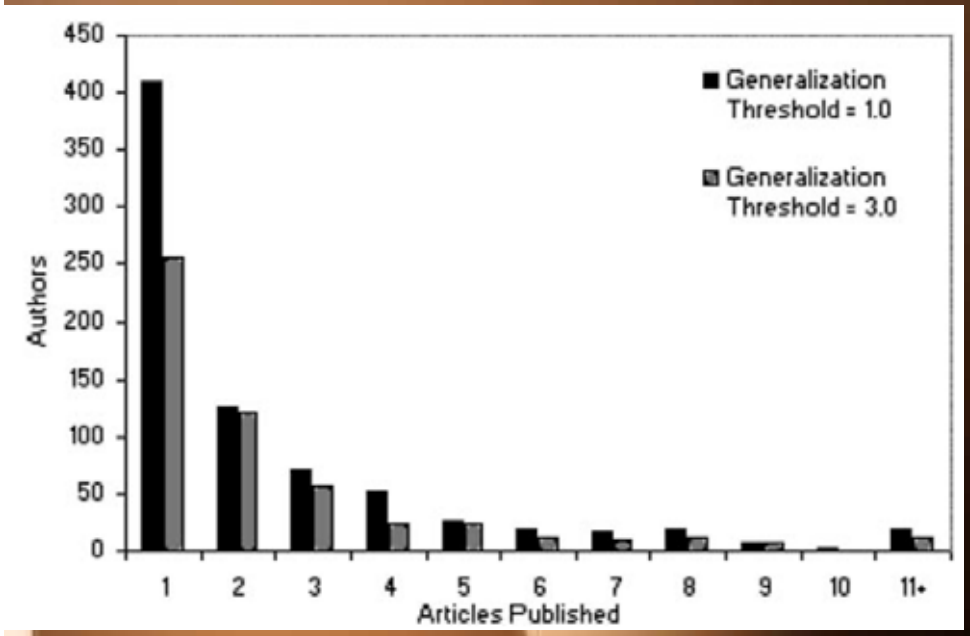
Simulation Examples

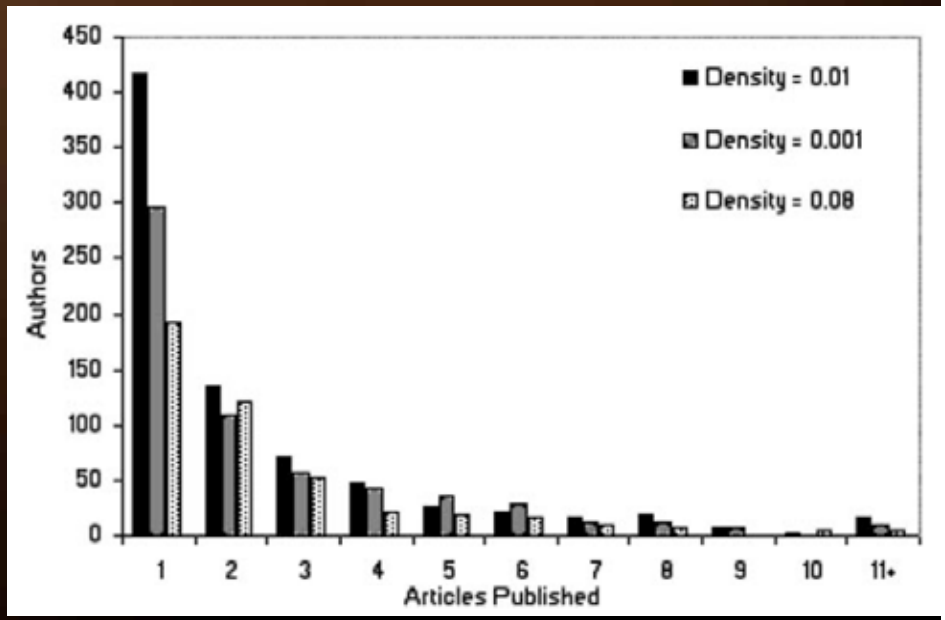
Varying cognitive parameters

- Generate different communities producing different numbers of papers, by varying cognitive parameters
- Power curves are obtained under different cognitive parameter settings

Simulation Examples

Varying cognitive parameters





Simulation Examples

Varying cognitive parameters

- Cognitive-social invariance
- General applicability and validity of the model
- Generate new theories and hypotheses
- Reduce the need for costly (or impossible) human experiments

Simulation Examples

Questions?

Summary

1. Representation
    1. Bottom-Level Representation
    2. Top-Level Representation
2. Learning
    1. Bottom-Level Learning
    2. Top-Level Rule Learning
3. Level Integration
4. Working Memory & NACS
5. Simulation Examples
6. **Summary**

Summary

- Bottom level: implicit representation
  - Captured by backpropagation networks
  - Types of bottom-level learning:
    - Standard backpropagation
    - Q-learning
    - Simplified Q-learning

Summary

- Top level: explicit representation
  - “Condition --> Action” pairs
  - Types of top-level rules:
    - Rule extraction and refinement (RER)
    - Independent Rule Learning (IRL)
    - Fixed Rules (FR)

Summary

- Level Integration
  - Stochastic Selection
    - Level is chosen probabilistically
  - Combination
    - Bottom-up rectification
      - Bottom-level outcome rectified at the top level
    - Top-down guidance
      - Top-level outcome assists action decision-making at the bottom level

Summary

- Working Memory
  - Involves:
    - Action-directed encoding of information
      - As opposed to automatic encoding
    - Gradual fading of information
    - Action-directed re-encoding (“refreshing”) of information
    - Limited storage capacity
  - Working memory is also used to transmit information between the action-centered and non-action-centered subsystems.

Summary

Thank You

Questions?