An Integrated Theory of the Mind

John R. Anderson  Daniel Bothell$  
Michael D. Byrne  Christian Lebiere

Psychological Review, 2004

Computational Intelligence Seminar A
presented by Lars Büsing
Basic architecture

ACT-R (Adaptive Control of Thought - Rational)

- models for cognitive architecture
- features a highly modular structure
- unifies of insights from diverse branches of psychology (procedural vs. declarative knowledge, arithmetic, language, vision etc.)
- (ideally) allows predictions for psychological experiments that involve different human competences
- reflects embodiment of human cognition, ie. takes into account important facts about perception and action
- provides robust behavior in unknown&unexpected situations
Basics unit of knowledge

- **declarative knowledge**: chunk
  - $n$-tuple of words/symbols, $n \leq 10$ number of slots
  - facts: “2+3=5”
  - awareness: “there is a car to my left”
  - goals: “have lunch”

- **procedural knowledge**: production
  - if-then rule, eg. “if there is a curve then steer”
    - if-part: search pattern for chunks
    - then: actions, commands
Basics building blocks

- **modules:**
  - implement specialized competences
  - examples: memory, vision, actuators
  - no fixed number

- **production system:** reads information from periph. modules and executes productions (=actions), central decision unit

- **buffers:** interface between production system and peripheral modules, can hold one chunk

- content of the buffers = situation context

- only buffers hold conscious information

![Diagram of the production system and buffers](image-url)
Scheduling and time in ACT-R

- modules work in (pseudo) continuous time (ie small time steps)
- production system looks for productions whose *if*-part is true
- when *if*-part of a production is matched, action is carried out 50 ms later
- pattern matching, sending commands, reading buffers doesn’t take any time, carrying out commands in modules may take time

![Diagram of ACT-R system](image)
Essential modules

- production system
- procedural memory
- declarative memory
- goal module
- (visual module)
- (motor module)
Production system & procedural memory

- Search in procedural memory: does any if-part of a production match the current chunks in the buffers?
  - No
  - Yes
    - Is there a single matching production?
      - No
      - Yes
        - Wait 50 ms
        - Conflict resolution
      - Execute then-part of matching production
Production system & procedural memory

- procedural memory: set of available productions \( \{o_1, \ldots, o_N\} \)
- production \( o_i = (if(o_i), then(o_i)) \)
- \( if(o_i) \): set of patterns to which chunks can be matched operating on the chunks currently contained in the buffers
- \( then(o_i) \): set of actions/commands to be executed eg “update goal buffer”, “initiate movement”
Production system & procedural memory

Chunk matching:

- **pattern** \((b, c)\)
- \(c\) is a chunk (remember chunk=tuple of slots), whose slots can also be filled with special *not*-symbols (eg “!small”) and variable symbols (“?size”)
- \(b\) is a buffer whose content \(C_b\) should be tested for matching with chunk \(c\)
- **matching rules**: \(C_b\) matches \(c\) if \(\forall\) slots
  - \(\text{slot}(C_b) = \text{slot}(c)\)
  - if \(\text{slot}(c)\) is a *not*-symbol: \(\text{slot}(C_b) \in \text{slot}(c)\)
    - eg “dog !small friendly” matches “dog large friendly”
- **matching of variables**

Production System

<table>
<thead>
<tr>
<th>Buffer 1</th>
<th>Buffer 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;dog small relaxed&quot;</td>
<td>&quot;cat small friendly&quot;</td>
</tr>
</tbody>
</table>

if: \{(Buffer 1, "dog ?size !angry"), (Buffer 2,"cat ?size friendly")\}

then: .....
Production system & procedural memory

search in procedural memory: 
do an if-part of a production 
match the current chunks in the buffers?

is there a single matching production?

execute then-part of matching production

wait 50 ms

no

no

conflict resolution

yes
Conflict resolution:

- assume: if-parts of $o_1, \ldots, o_K$ are matched
- calculate utility $U_i$ of production $o_i$:

$$U_i = P_i \cdot G - C_i$$

- $P_i$: estimation of prob($G$ achieved|$o_i$ carried out) with successes/#trials
  also Bayesian estimation with prior possible
- $G$: value of current goal (stored in goal buffer)
- $C_i$: costs of production $o_i$;
  quantified in terms of time elapsed from execution to outcome of goal $G$
- probability of executing $o_i$ given by softmax over utilities
Declarative memory

- stores chunks eg. from visual buffers but also ones that were generated by productions

- retrieval: production passes a *pattern* to the declarative memory module which collects all matching chunks \( \{c_1, \ldots, c_K\} \)

- \( \forall c_i \in \{c_1, \ldots, c_K\} \) calculate activation \( A_i \) (formalization later...)

- select \( c_i \in \{c_1, \ldots, c_K\} \) with largest activation \( A_i \) (or softmax)

- wait \( \propto \exp(-A_i) \) time steps and place \( c_i \) into the retrieval buffer

- chunk merging: if two chunks are identical in declarative memory then they are merged and strengthened
Declarative memory

possible model memory activation with a model of attention:

\[ A_i = B_i + \sum_{j \in \text{Obj}s} S_{ij} W_j \]

- base line activation \( B_i \)

\[ B_i = \ln \left( \sum_{j=1}^{n} t_j^{-d} \right) \]

\( t_{i,f} \) times of retrieval of chunk \( c_i \):
"memories which have been retrieved often tend to be more present"

- \( W_j \): (visual) attentional weight on object \( j \)
- \( S_{ij} \): association between object \( j \) and chunk \( i \), generality of memory vs. strength of association?
Declarative memory

probabilistic interpretation of activation:

\[ A_i = B_i + \sum_{j \in Objs} S_{ij} W_j \]

- context \( C = \{O_j | j \in Objs\} \), set of objects
- \( A_i = \log p(i|C) \), log posterior
- \( B_i = \log p(i) \), log prior
- assume naive Bayes: \( p(C|i) = \prod_{j \in Objs} p(O_j|i) \) and independence \( p(C) = \prod_{j \in Objs} p(O_j) \)

\[ S_{ij} W_j = \log(p(i|O_j)/p(i)) \]
Fan experiment

- hypothesis: $S_{ij} W_j \propto \log(p(i|O_j)/p(i))$ where

- subjects were given 10 facts about 3 persons and 3 places “a hippie was in the park”

- training to recognize the facts from false facts for 10 days

- time to distinguish true statement was recorded as function of fan and days of training

- $p(i|O_j)$ and $p(i)$ are given by fan-in of memory chunks
Fan experiment

- $A_i = B_i + \sum_j S_{ij} W_j$
- $B_i = \log(\text{days})$
- retrieval time = $\exp(-A_i)$
Goal module

- stores goal chunks
- has a stack structure
- goal chunks can be build recursively
Mapping: model → anatomy

- production system: basal ganglia & thalamus
  - striatum: matching
  - palladium: conflict resolution, "winner-lose-all" via double-inhibition
  - thalamus: action execution
- declarative memory: prefrontal cortex, sensitive wrt. memory storage and retrieval
- goal module: anterior cingulate cortex
Learning in ACT-R

- store new chunks in declarative memory
- build associations between objects and chunks in declarative memory and reevaluate activities of chunks
- learning of utilities of productions
- generate new productions by *production compilation*:
  - input: tuple of productions that are often executed one after another
  - output: single collapsed production
  - eg: memory retrieval + general production $\rightarrow$ specialized production
Example: model for past tense learning

- task: children learning past tense forms of regular and irregular verbs
- experimental results:
  - U-shaped learning curve for irregular verbs
  - acquisition of some irregular verbs then discovery of regular rule
  - children almost never get&use error signals
- previous ANN models do not perform well with realistic distributions of regular vs irregular verbs
Example: model for past tense learning

model:

- whenever a past tense form is perceived it is stored in declarative memory: eg "go - went"
- hypothesis: children have three built-in strategies (productions) for past tense
  - *then*: use present tense form,
    these mistakes cannot be tracked experimentally
  - *if*: if correct for is stored *then*: retrieve it
  - analogy: *then* look for some verb in declarative memory and generalize, slow, takes two cycles: retrieval of memory and apply analogy
- combination of retrieving a regular pair (eg "help - helped") and applying analogy rule is successful → production compilation to regular rule
- this model reproduces basic facts about past tense learning
Model results
Summary

- basic units of declarative and procedural knowledge: chunks and productions
- independent modules that interact via low bandwidth channels (buffers) with production system, the remaining information is encapsulated
- parallel processing of the modules & serial bottleneck in production system (decision unit)
- productions are executed sequentially depending on the context (=state of all buffers)
- learning is modeled on four different stages