Cognitive Supermodels
or: We Can’t Play 20 Models with Nature and Win, Either
or: Dr. Strangebrain, or, How I Learned to Stop Worrying and Turn On All the Learning Mechanisms

Dario Salvucci
Drexel University
A Tale of Two Modeling Efforts

- We can very roughly characterize the past decade of cognitive modeling work in 2 ways

  1. Modeling “software” for a specific domain
     - sometimes a focused experimental domain
       - e.g., task switching, dual-choice/PRP
     - sometimes an applied domain
       - e.g., driving
       - e.g., air-traffic control
     - good in their accounts of the details of a task
     - not as good for generalizing to (even very similar) tasks
       - e.g., could driver model be used for walking? biking?
(2) Modeling “hardware” aimed at a specific, cross-domain aspect of cognition

- several recent examples
  - production compilation
  - utility learning
  - threaded cognition

- good for their generalization of component cognitive skills across domains

- good for continuing to flesh out the architecture

- but agnostic (at best) / useless (at worst) without actual domain-specific “software”
  - or at least declarative instructions that result in “software”
A Tale of Two Modeling Efforts

- What we don’t have very much of:
  Cross-domain software models
    - a single model that accounts for behavior across many diverse task domains

- There have been instances of this, such as...
  - list memory: representations & rules
    - e.g., used for dialing #s
  - analogy

- Can we build on recent successes toward a larger-scale effort for cross-domain models?
Cognitive Supermodels

- Idea: Consider a single cognitive model with...
  - a single (initial) set of declarative chunks
  - a single (initial) set of production rules
  - with a single (initial) set of parameter settings
  - on a fixed cognitive architecture

- ... and try to account for behavior across a range of diverse domains...
  - list memory, algebra, dual-choice/PRP, etc.
  - driving, air-traffic control, etc.

- This is what I’ll call a Cognitive Supermodel
Another way to think about this:

- A person comes into your experiment.
- This person has a lifetime of experiences that has shaped his/her chunks, rules, parameters.
- Can we represent this canonical person as a single canonical model — a cognitive supermodel?
- Ideally, this model should represent the distribution of all possible participants in the experiment (i.e., of the target group: age, skill sets, etc.)
  - but for now, a canonical supermodel is hard enough
Development Environment

- We need a development environment that allows for testing of one or more models across many domains
  1. real-time simulation for visualization, fast simulation for fast estimates
  2. single-domain simulation for testing, cross-domain simulation for model fitting
  3. implementation of complex task environments

- LISP ACT-R can do all this (though #3 is harder)
- But I wanted something a bit more integrated
Java ACT-R

- Java ACT-R is a completely re-implemented version of ACT-R
  - includes all the basic functionality
  - currently uses the tutorial sample models for testing

- This is not jACT-R (by Tony Harrison)
  - jACT-R is intended to be plug & play for different architectural modules
    - can specify model in XML or LISP(ish) (except for params)
  - this new system is intended to be more monolithic and streamlined, centered on batch model testing

- Demo...
Java ACT-R

- What Java ACT-R does have
  - all the core learning mechanisms
  - all the core perceptual/motor mechanisms
  - simple evals (arithmetic + user-defined)
  - p*-style slot variables
  - threading, EMMA

- What Java ACT-R does not have
  - no chunk types!
    - can set any slot name to a slot value
    - “isa” like any other slot (except for fan & partial matching)
    - chunk types don’t seem to be necessary
Java ACT-R

- Many workshops ago (1995), I presented a new implementation of ACT-R in C
  - back then, LISP speed was an issue; now, it’s not
  - the challenge then & now: maintaining the code
    - the LISP implementation = the theory

- This new system
  - the goal is not to maintain correspondence with the LISP version
  - in fact, the goals are
    - (1) to explore different variations on ACT-R
    - (2) to use this as the foundation for a monolithic cognitive supermodel that accounts for a set of domains
A First Cognitive Supermodel

- For our first attempt at a cognitive supermodel, we need...
  - basic declarative knowledge
    - number facts, etc.
  - basic procedural skills
    - e.g., clicking an interface icon, typing a key
    - instruction-following skills
      - listening to and encoding instructions
      - following them to generate actions
    - extending previous work by Taatgen, Fu, Anderson, etc.
Instruction Following

- Listening / encoding instructions
  - encode instructions aurally, word-by-word
  - memorize each instruction (phrase) by rehearsal
  - go until “start <task>” is heard

- Following instructions
  - recall instruction one at a time
  - perform action or initiate associated subgoal
    - some instructions are a single rule
      - e.g., type a letter
    - some instructions initiate complex subgoals
      - e.g., the simple instruction “drive”
  - compiled with production compilation
#1: Paired Associates

## Instructions

to respond:
wait-for visual-change
read word
recall number for word
if success type number
wait-for visual-change
read number
memorize state
repeat

current instruction
(invisible to model)
current word
(presented as audio)
#1: Paired Associates

**Instructions**

**to respond:**
- wait-for visual-change
- read word
- recall number for word
- if success type number
- wait-for visual-change
- read number
- memorize state
- repeat

specifies the task/goal being defined
#1: Paired Associates

- **Instructions**

  to respond:

  - **wait-for visual-change**
  - read word
  - recall number for word
  - if success type number
  - wait-for visual-change
  - read number
  - memorize state
  - repeat
#1: Paired Associates

## Instructions

To respond:
- Wait for visual change
- Read word
- Recall number for word
- If success type number
- Wait for visual change
- Read number
- Memorize state
- Repeat
#1: Paired Associates

## Instructions

to respond:

wait-for visual-change
read word
recall number for word
if success type number
wait-for visual-change
read number
memorize state
repeat

```plaintext
(p read*done
  =goal>
  isa read
  object word
  ...
  =visual>
  isa text
  value =value
  =imaginal>
  ==> =imaginal>
  word =value
  =goal>
  ...
)
```
#1: Paired Associates

## Instructions

to respond:

- wait-for visual-change
- read word
- recall number for word
- if success type number
- wait-for visual-change
- read number
- memorize state
- repeat

“word” in instruction defines the slot name for recall

memorized imaginal state contains both word and number; it is this state that is recalled
#1: Paired Associates

- **Instructions**

  to respond:
  - wait-for visual-change
  - read word
  - recall number for word
  - if success type number
  - wait-for visual-change
  - read number
  - memorize state
  - repeat
#1: Paired Associates

**Instructions**

to respond:
wait-for visual-change
read word
recall number for word
if success type number
wait-for visual-change
read number
**memorize state**
repeat

all task information accumulated in the imaginal buffer (“problem state”); can be memorized via repeated rehearsals
#1: Paired Associates

## Instructions

to respond:
- wait-for visual-change
- read word
- recall number for word
- if success type number
- wait-for visual-change
- read number
- memorize state
- repeat
#1: Paired Associates

## Instructions

To respond:

- wait-for visual-change
- read word
- recall number for word
- if success type number
- wait-for visual-change
- read number
- memorize state
- repeat

**start respond**
#1: Paired Associates

- **Instructions**

  to respond:
  - wait-for visual-change
  - read word
  - recall number for word
  - if success type number
  - wait-for visual-change
  - read number
  - memorize state
  - repeat

  many aspects of the instructions make use of well-practiced skill knowledge
#2: Fan Effect

## Challenges

- tutorial model accounts only for the testing stage
  - chunks are already in memory, activated strongly
- a supermodel needs to account for the studying stage
  - chunks must be learned, and activated sufficiently
  - how do we activate them all evenly?
    - any discrepancy can alter the data pattern drastically
- randomize study sequence, 10x/sentence
- randomize test sequence

### Human Data:

<table>
<thead>
<tr>
<th>1.11, 1.17, 1.22</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.17, 1.20, 1.22</td>
</tr>
<tr>
<td>1.15, 1.23, 1.36</td>
</tr>
<tr>
<td>1.20, 1.22, 1.26</td>
</tr>
<tr>
<td>1.25, 1.36, 1.29</td>
</tr>
<tr>
<td>1.26, 1.47, 1.47</td>
</tr>
</tbody>
</table>
#2: Fan Effect

## Instructions

**to study-sentences:**
- wait-for visual-change
- read person
- read location
- memorize state
- repeat

**to recall-sentences:**
- wait-for visual-change
- read person
- read location
- recall location for person as recalled-location
- compare location to recalled-location
- if success type k
- if failure type d
- repeat

start study-sentences
(...and then later...)
start recall-sentences
#3: Tracking & Choice

**Task**
- tracking: follow arrow with the mouse
- choice: identify left/right arrows

**Challenges**
- multitasking: tracking & choice learned separately, must be interleaved
  - answer: threaded cognition
- eye movements: effects arise from distance of arrow from target area
  - answer: EMMA eye-movement model
#3: Tracking & Choice

## Instructions

**to track-target:**
- move-mouse-to target
- repeat

**to respond-to-arrow:**
- read arrow
- if failure repeat
- compare arrow literal ‘<’
- if success punch left-pinkie
- if failure punch left-middle
- repeat

start **track-target** and **respond-to-arrow**
Challenges

- **Base-level learning**
  - should be on -- it’s a core learning mechanism
  - affects instruction following
    - need to rehearse instructions, or they’re gone later
  - affects Paired Associate model
    - need to memorize state chunks enough to retrieve later
  - affects Fan Effect model
    - tutorial model uses (set-base-levels ...)
    - rehearsing each chunk to achieve the same base level is difficult in practice -- study has to be done just right
Challenges

- Production compilation
  - should be on -- it’s a core learning mechanism
  - affects the Paired Associate model
    - forms specific rules for specific pairs
  - affects the Tracking model
    - reduces tracking loop from 7 to ~3 rule firings (good!)
    - how does compilation interact with threads?
  - affects the Fan Effect model
    - eliminates retrievals --> no more fan effect (bad!)
    - this is a general issue of keeping some retrievals
      - e.g., rehearsal productions shouldn’t be compiled
Challenges

- **Other parameters**
  - **activation noise (:ans)**
    - Paired: 0.5
    - Fan: none
  - **expected gain noise (:egs)**
    - Paired: 0.1
    - Fan: none
  - **retrieval threshold (:rt)**
    - Paired: –1.7
    - Fan: 0.0
Results

Currently... not great

<table>
<thead>
<tr>
<th>Task</th>
<th>Check</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paired</td>
<td>true</td>
<td>~.96</td>
</tr>
<tr>
<td>Tracking</td>
<td>true</td>
<td>~.70</td>
</tr>
<tr>
<td>Fan</td>
<td>true</td>
<td>bad!</td>
</tr>
</tbody>
</table>

- Paired model is robust
- Tracking model is somewhat robust
- Fan model is very sensitive
Reflections

- Some of this retraces previous work
  - like Fu et al.'s “over-the-shoulder” instructions

- Some new things
  - the “experimenter finger”
  - no chunk types (not a new idea though)
    - variable slots & instruction following
  - new integrated testing environment

- The big challenge
  - Bonnie John has said, ‘Why can’t we just turn on all the learning mechanisms at once?’
  - this is a test of this question — there’s no easy answer
Reflections

The big benefit: model testing & fidelity
- let’s say we propose a new theory of...
  - sequential actions, like Memory for Goals
    - every goal must be rehearsed, then retrieved
  - basal ganglia, a la Andrea Stocco’s work
    - especially: no two variables on the LHS
  - etc. etc.
- when integrated into a cognitive supermodel,
  we immediately test its effects across many domains
Moving Forward

- After a lot of work, we have...
  a single model of 3 tasks, sort of
- But now we have the infrastructure to start rigorously pursuing a cognitive supermodel
Moving Forward

- After a lot of work, we have... a single model of 3 tasks, sort of.
- But now we have the infrastructure to start rigorously pursuing a cognitive supermodel.

For this each domain uses a different model.

Can we do this with the same model?

### Task Check Score
---
Unit1-Count true ----
Unit1-Addition true ----
Unit1-Semantic true ----
Unit1-Tutor true ----
Unit2-Demo true ----
Unit3-Sperling true ----
Unit4-Paired true 0.98
Unit5-Siegler true 0.96
Unit5-Grouped true ----
Unit5-Fan true 0.86
Unit6-BST true 0.73
Unit7-Paired true 0.99
Moving Forward

- After a lot of work, we have... a single model of 3 tasks, sort of
- But now we have the infrastructure to start rigorously pursuing a cognitive supermodel

Here’s what we have now

<table>
<thead>
<tr>
<th>Task</th>
<th>Check</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paired</td>
<td>true</td>
<td>~.96</td>
</tr>
<tr>
<td>Tracking</td>
<td>true</td>
<td>~.70</td>
</tr>
<tr>
<td>Fan</td>
<td>true</td>
<td>bad!</td>
</tr>
</tbody>
</table>
Moving Forward

- After a lot of work, we have... a single model of 3 tasks, sort of
- But now we have the infrastructure to start rigorously pursuing a cognitive supermodel

Here’s what we want...

<table>
<thead>
<tr>
<th>Task</th>
<th>Check</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fan Effect</td>
<td>true</td>
<td>0.86</td>
</tr>
<tr>
<td>Paired Assoc.</td>
<td>true</td>
<td>0.94</td>
</tr>
<tr>
<td>PRP</td>
<td>true</td>
<td>0.98</td>
</tr>
<tr>
<td>Driving Exp 1</td>
<td>true</td>
<td>0.87</td>
</tr>
<tr>
<td>Driving Exp 2</td>
<td>true</td>
<td>0.82</td>
</tr>
<tr>
<td>Driving Exp 3</td>
<td>true</td>
<td>0.84</td>
</tr>
<tr>
<td>ATC Exp 1</td>
<td>true</td>
<td>0.90</td>
</tr>
<tr>
<td>Document Editing</td>
<td>true</td>
<td>0.78</td>
</tr>
<tr>
<td>Web Browsing</td>
<td>true</td>
<td>0.91</td>
</tr>
<tr>
<td>Choosing Coffee</td>
<td>true</td>
<td>0.99</td>
</tr>
<tr>
<td>etc. etc. etc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>