5.3 ACT-UP: A Toolkit for Hampton, Cognitive Modeling Composition, Reuse and Integration

MODSIM WORLD Conference & Expo

October 13–15, 2010
Hampton, Virginia

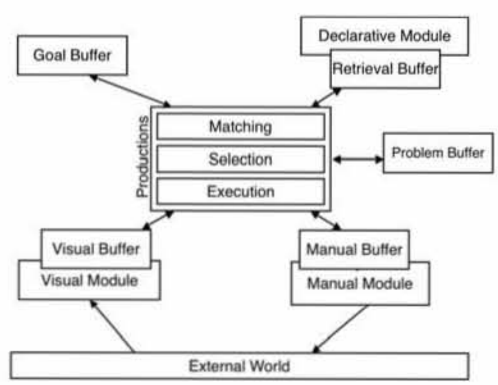
ACT-UP: A Cognitive Modeling Toolkit for Composition, Reuse and Integration

Christian Lebiere and David Reitter
Carnegie Mellon University
cl@cmu.edu; reitter@cmu.edu

MODSIM WORLD Conference & Expo

ACT-R Cognitive Architectures

- Computational implementation of unified theory of cognition
- Commitment to task-invariant mechanisms
- Modular organization
- Limited capacity
- Hybrid symbolic statistical processes



Motivations and Applications

- **Philosophy**: Unified understanding of the mind.
- **Psychology**: Account for experimental data.
- **Education**: Provide cognitive models for intelligent tutoring systems and other learning environments.
- **Human Computer Interaction**: Evaluate artifacts and help in their design.
- **Computer Generated Forces**: Provide cognitive agents to inhabit training environments & games.
- **Neuroscience**: Provide a framework for interpreting data from brain imaging.

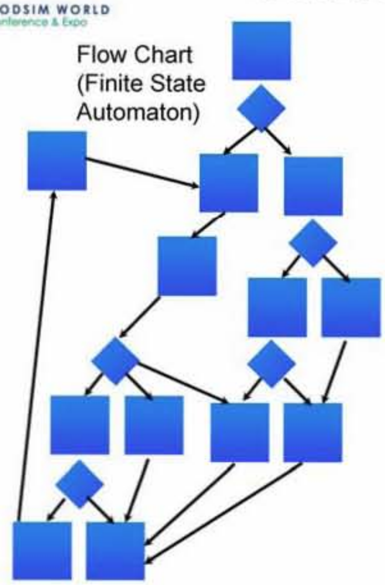
Goals

- Enable the implementation of more complex ACT-R models
- Scale up cognitive models to simulate learning / adaptation in communities (e.g., about 1,000 models in parallel)
- Treat models as hard claims
  - Evaluate each specified component against data
  - Underspecify the rest and fit free parameters

The Argument

- **Constraints:** Architectural advances require further constraints
- **Scaling it up:** Complex tasks, broad coverage of behavior (e.g., linguistic), use of microstrategies and predictive modeling may serve to motivate further architectural constraints
- **Difficulties:** ACT-R is heavily constrained already, and models are difficult to develop, reuse and exchange

Control Structure

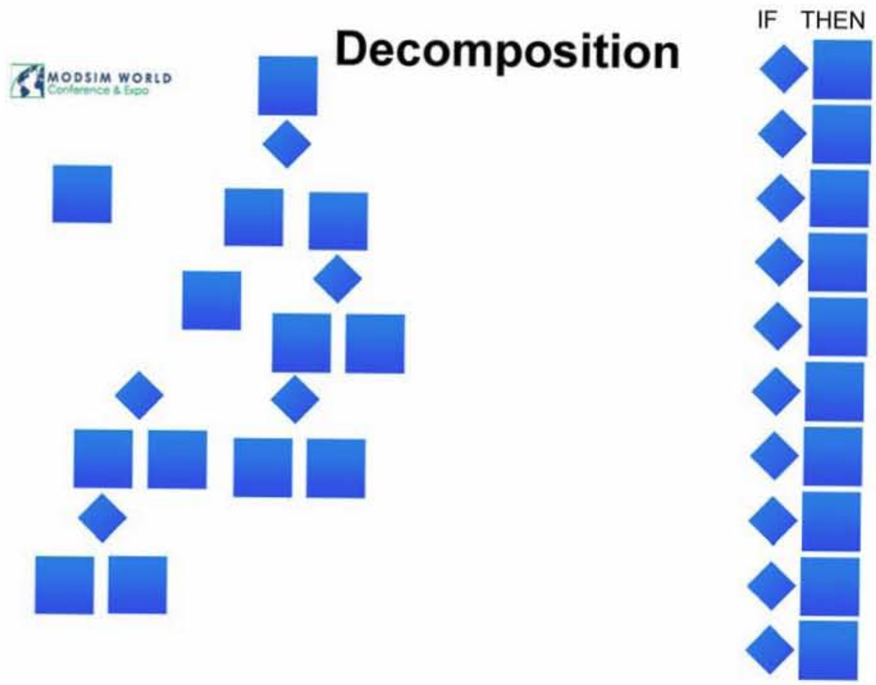


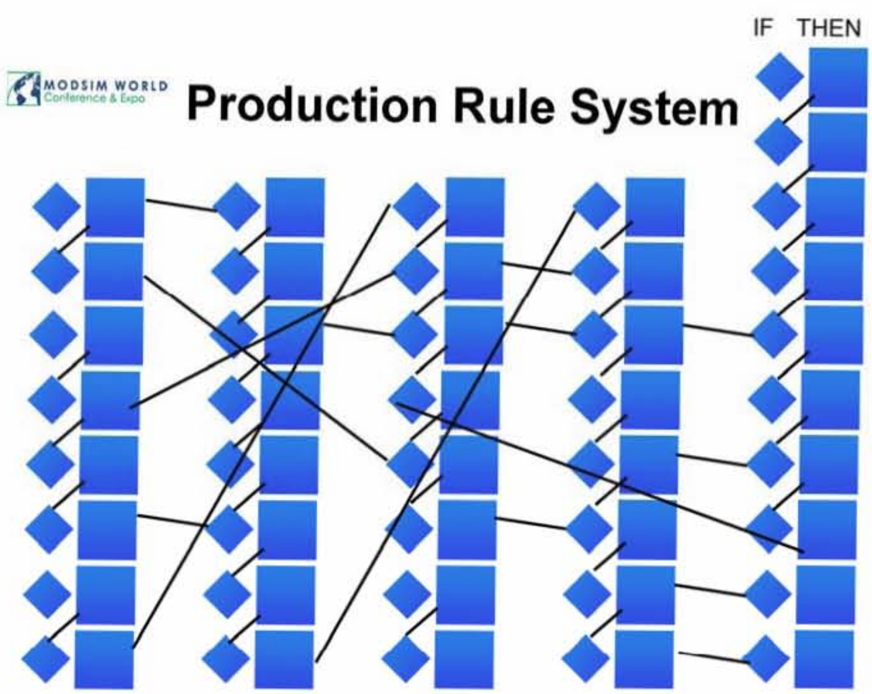
A flow-chart describes an algorithm (or a cognitive strategy)

Decision-making points and states

Not easy to reuse: it fails to capture generalizations

Computer Science: pre-Object Orientation, pre-Functional Programming

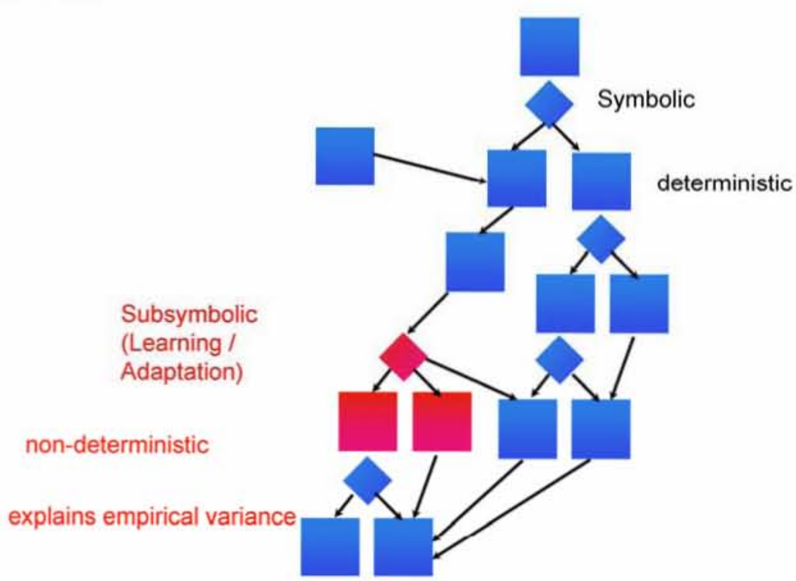

MODSIM WORLD
Conference & Expo
Decomposition
IF
THEN


IF
THEN
MODSIM WORLD
Conference & Expo
Production Rule System

The Argument

- **Constraints:** Architectural advances require further constraints
- **Scaling it up:** Complex tasks, broad coverage of behavior (e.g., linguistic), use of microstrategies and predictive modeling may serve to motivate further architectural constraints
- **Difficulties:** ACT-R is heavily constrained already, and models are difficult to develop, reuse and exchange
- **We need to produce models at a higher abstraction level**
  – However, we'd like to leverage successful cognitive modules, describing memory retention, cue-based retrieval, routinization, reinforcement learning

Cognitive Strategy



Priming Model

Crucial request of a chunk from declarative memory

- Only a small portion of the model explains the behavioral data at hand
- The rest explains that the task can be accomplished in principle with a parallel architecture and with specific cognitive representations (chunk types)

Production Systems vs. assembly language

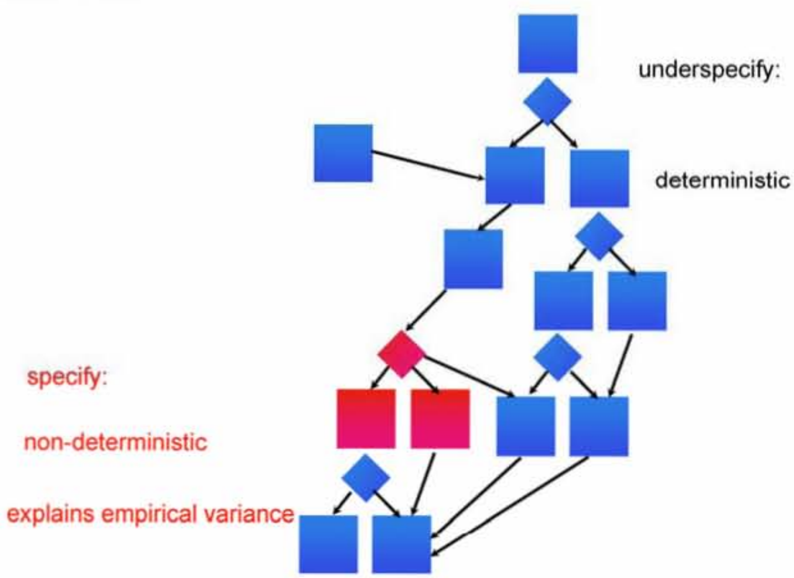
```
evensum:    clr.l     D1        ;Zero-out
                                ;Accumulator
sumloop:    add.l     D0,D1     ;Add current
                                ;counter value to
accumulator
            subq      #1,D0     ;Decrement
                                ;counter by one
            bne       sumloop   ;until it
                                ;reaches zero
            muls      #2,D1     ;Double sum to account
                                ;for even numbers
            rts                 ;Return
                                ;to caller
```

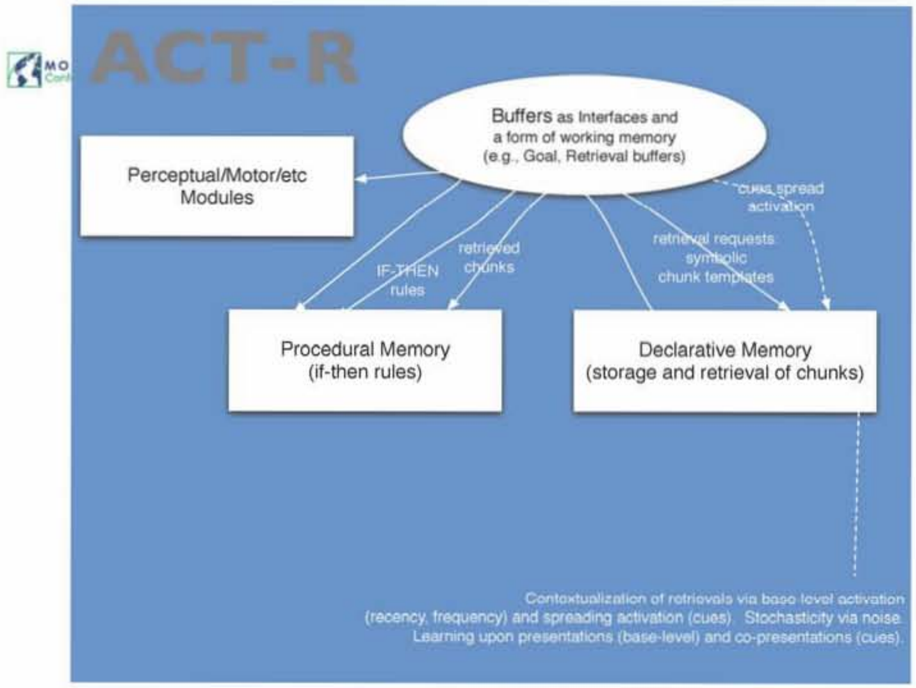
~1990

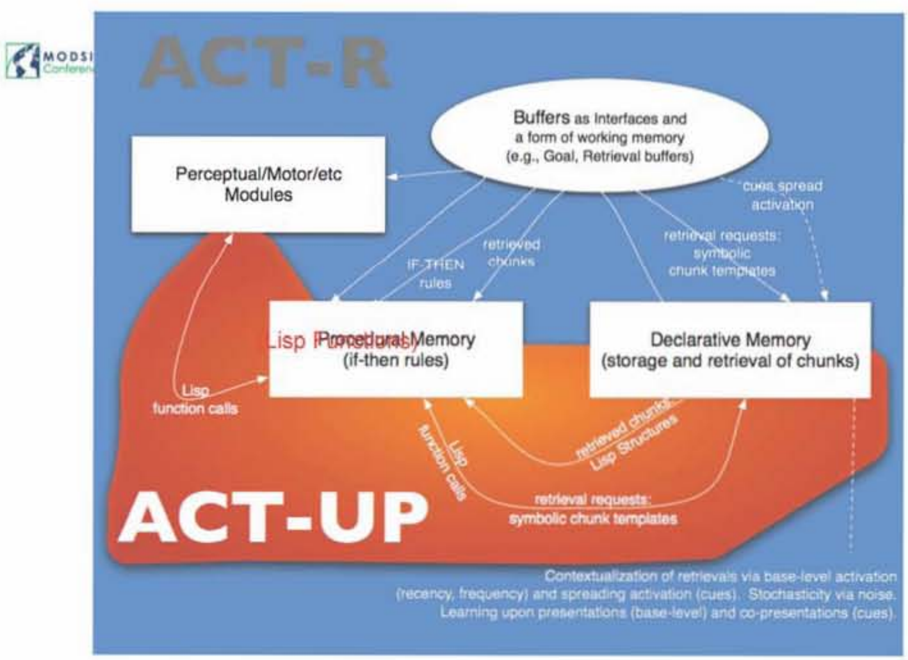
The Argument

- **Constraints:** Architectural advances require further constraints
- **Scaling it up:** Complex tasks, broad coverage of behavior (e.g., linguistic), use of microstrategies and predictive modeling may serve to motivate further architectural constraints
- **Difficulties:** ACT-R is heavily constrained already, and models are difficult to develop, reuse and exchange
- **Abstraction:** To implement those, we need to produce models at a higher abstraction level
- **Underspecification is the key to focus on verifiable claims, and to avoid overfitting by fitting free parameters to data**

Underspecified Models




ACT-R
Buffers as Interfaces and
a form of working memory
(e.g., Goal, Retrieval buffers)
Perceptual/Motor/etc
Modules
cues spread
activation
retrieval requests:
symbolic
chunk templates
retrieved
chunks
IF-THEN
rules
Procedural Memory
(if-then rules)
Declarative Memory
(storage and retrieval of chunks)
Contextualization of retrievals via base-level activation
(recency, frequency) and spreading activation (cues). Stochasticity via noise.
Learning upon presentations (base-level) and co-presentations (cues).


ACT-R
Buffers as Interfaces and
a form of working memory
(e.g., Goal, Retrieval buffers)
Perceptual/Motor/etc
Modules
cues spread
activation
retrieval requests:
symbolic
chunk templates
retrieved
chunks
IF-THEN
rules
Procedural Memory
(if-then rules)
Lisp Functions)
Declarative Memory
(storage and retrieval of chunks)
Lisp
function calls
Lisp
function calls
retrieved chunks:
Lisp Structures
retrieval requests:
symbolic chunk templates
ACT-UP
Contextualization of retrievals via base-level activation
(recency, frequency) and spreading activation (cues). Stochasticity via noise.
Learning upon presentations (base-level) and co-presentations (cues).

ACT-UP

- A stand-alone system on the basis of Common Lisp
- targets an audience that can write simple Lisp programs (unlike, e.g., CogTool)
- Toolbox approach to ACT-R
  - light-weight: it's a Lisp library
  - does not produce production rules (ACT-R/Lisa, ACT-Simple, CogTool)
- Not aimed at implementing all constraints of ACT-R 6 (unlike Java ACT-R, Python ACT-R)

Declarative Memory

- `define-chunk-type'
  - types are optional
- `make-count-order'
- `learn-chunk'
- `defrule'
- `retrieve-chunk'
- `count-order-second'

ACT-UP Code

```
;; ACT-R parameters
(setq *lf* .05)
(setq *rt* -1)

;;;; Defining chunk type

(define-chunk-type count-order first second)
```

ACT-UP is not ACT-R 6...

- ACT-UP Interface is synchronous
  - Serial execution
  - Deterministic strategies defined as programs
- Parallelism (e.g., perceptual/motor modules) possible [not implemented]
- Non-deterministic rule choice is possible
  - Reinforcement-learning as in ACT-R 6

PM / Utility learning

- `choose-coin'
- calls either `decide-heads or `decide-tails'
- `assign-reward' reinforces the decision
- Exact production rules are underspecified,
  - but decision-making point is explicit
- Choice model replicates ACT-R and empirical results

```
;; Experimental environment
(defun toss-coin ()
  (if (< (random 1.0) .9) 'heads 'tails))

;; The Model
;;;; Rules that return the choice as symbol heads or tails

(defrule decide-tails ()
  :group choose-coin
  'tails)

(defrule decide-heads ()
  :group choose-coin
  'heads)
```

Debugging

```
(defrule count-model (arg1 arg2)
  "Count from ARG1 to ARG1.
ARG1 is the starting point and ARG2 is the ending point.
Each increment is 1 unit."
  (speak arg1)
  (if (not (eq arg1 arg2))
      (let ((p (debug-detail (retrieve-chunk (list :chunk-type 'count-order
                                                   :first arg1)))))
        (if p
            (count-model (count-order-second p) arg2)))
      ;; else return end point
      arg2))
```

Debugging

```
CL-USER> (debug-detail (do-it 1))

make-match-chunk (make-TYPE*): No such chunk in DM.  Returning new chunk (not in DM) of name LOSE
Presentation of chunk LOSE (MP: NIL t=72761.26.  M: MODEL521436, t=0.
Implicitly creating chunk of name LOST.
Presentation of chunk LOST (MP: NIL t=72761.26.  M: MODEL521436, t=0.
Implicitly creating chunk of name BLANK.
Presentation of chunk BLANK (MP: NIL t=72761.305.  M: MODEL521436, t=72761.305.
make-match-chunk (make-TYPE*): No such chunk in DM.  Returning new chunk (not in DM) of name HAVE
Presentation of chunk HAVE (MP: NIL t=72761.445.  M: MODEL521436, t=72761.445
Implicitly creating chunk of name HAD.
Presentation of chunk HAD (MP: NIL t=72761.445.  M: MODEL521436, t=72761.445.
Group PAST-TENSE-MODEL with 1+0 matching rules, choosing rule PTMODEL (Utility 5.0709996)
Group FORM-PAST-TENSE with 3+0 matching rules, choosing rule STRATEGY-WITHOUT-ANALOGY (Utility 5.225957)
retrieve-chunk:
   spec: (CHUNK-TYPE PASTTENSE VERB GET)
  cues: NIL
  pmat: NIL
filtered 0 matching chunks.
retrieved none out of 0 matching chunks.
NIL
Assigning reward 3.9
Assigning reward 3.853125 to STRATEGY-WITHOUT-ANALOGY. STRATEGY-WITH-ANALOGY remains best regular rule in group FORM-PAST-TENSE.
Assigning reward 0.0 to PTMODEL. Best regular rule among alternatives in group PAST-TENSE-MODEL!
NIL
CL-USER>
```

MODSIM WORLD Conference & Expo

Implemented Models

- 10 Classic models implemented:
  - count, addition, siegler, zbrodoff, paired, fan, sticks, semantic, choice, past-tense

* past-tense not yet complete

MODSIM WORLD Conference & Expo

Efficiency

- Sentence production (syntactic priming) model
  - 30 productions in ACT-R, 720 lines of code
  - 82 lines of code in ACT-UP (3 work-days)
  - ACT-R 6: 14 sentences/second
  - ACT-UP: 380 sentences/second

Scalability

- Language evolution model
  - Simulates domain vocabulary emergence (ICCM 2009, JCSR 1010)
  - 40 production rules in ACT-R (could not prototype)
  - 8 participants interacting in communities
- In larger community networks: 1000 agents, 84M interactions (about 1 minute sim. time each), 37 CPU hours

Rapid prototyping/Reuse

- Dynamic Stocks&Flows model (JAGI 2010)
  - Competition entry, model written in < 1 person-month
  - Instance-based learning (IBL, Gonzales&Lebiere 2003)
  - Blending (Wallach&Lebiere 2003)
  - free parameters (timing) estimated from example data
  - Model generalized to novel conditions
    - (.... NOT. but it did so better than others.)
- Same IBL/blending micro-strategy was re-used directly in a *Lemonade Stand Game* entry to a 2009 competition (BRIMS 2010)

MODSIM WORLD Conference & Expo

Drawbacks

- Less established code-base than ACT-R 6
- Lisp
- Lack of architectural timing predictions from rule matching
- Lack of parallelism (planned: fall 2010)
- lack of perception/motor modules
  - Will be available in ACT/Simple-style interface (Salvucci&Lee 2003)

MODSIM WORLD Conference & Expo

Beta-Test

- **Limited Release** of ACT-UP test version
  - comes with 10 example models
  - 4 tutorials (paralleling the ACT-R 6 ones)
  - Full API documentation plus *How-do-I...* document
- Testing period: Fall 2010
- Task: implement 1-2 models of your own
- Review letter requested (journal-review style)