ACT-R 5.0/HOT

(Hands-on Tutorial)

Mike Schoelles
Rensselaer Polytechnic Institute
CogWorks Laboratory
schoem@rpi.edu

Frank Ritter
Penn State University
Applied Cognition Laboratory
ritter@ist.psu.edu

Slides with * from
Introduction to ACT 5.0 Tutorial by Christian Lebiere,
http://act-r.psy.cmu.edu/tutorials/

Loading and Running ACT-R

for PCs

Using ACL

1. Launch via Start > ACL Professional > ACL Profession with Func IDE
2. Go to File > Load. Choose the file C:\ACT-R Environment\loader.lisp.
3. Many warnings appear.
4. (ACT-R 5 is loaded at this point.)
5. Using Windows Explorer, launch the ACT-R environment via C:\ACT-R Environment\Start Environment
6. Back in ACL, enter the command (start-environment) to start and (stop-environment) to stop. Note that in the ACT-R control panel, the Open Model command should not be used; Load Model should be used instead. Also, all editing must be done in an outside editor, not in the environment.

Loading and Running ACT-R

for Mac

[There should be a folder on the top level of the disk, with alias to all these parts]

Double click the MACL 5.0 icon

Choose the File: Load file... menu item

Navigate to to the HOT ACT-R folder and load "loader.lisp"

Find the Start OSX Environment and double-click it

Type to the MACL prompt "(start-environment) <CR>"

The control pane should then have buttons on it.

Tutorial Overview

- Cognitive Architecture/Modeling Overview
- ACT-R Theory Overview
  - Addition,counting and letter models
- Build a model (Dialing Model)
- ACT-R Theory Details
  - Sternberg and Building Sticks models
- Future directions for ACT-R

What is a Cognitive Architecture?

- Infrastructure for an intelligent system
- Cognitive functions that are constant over time and across different task domains
- Analogous to a building, car, or computer

Unified Theories of Cognition

- Account of intelligent behavior at the system-level
- Newell's claim
  - "You can't play 20 questions with nature and win"

Integrated Cognitive Architecture

- Cognition doesn't function in isolation
  - Interaction with perception, motor, auditory, etc. systems
- Embodied cognition
  - Represents a shift from
    - *"mind as an abstract information processing system"*
    - *Perceptual and motor are merely input and output systems*
  - Must consider the role of the environment
  - Other body processes
    - *Effects of caffeine, stress and other moderators*

Motivations for a Cognitive Architecture *

1. **Philosophy:** Provide a unified understanding of the mind.

2. **Psychology:** Account for experimental data.

3. **Education:** Provide cognitive models for intelligent tutoring systems and other learning environments.

4. **Human Computer Interaction:** Evaluate artifacts and help in their design.

5. **Computer Generated Forces:** Provide cognitive agents to inhabit training environments and games.

6. **Neuroscience:** Provide a framework for interpreting data from brain imaging.

7. All of the above

Requirements for Cognitive Architectures*

1. Integration, not just of different aspects of higher level cognition but of cognition, perception, and action.

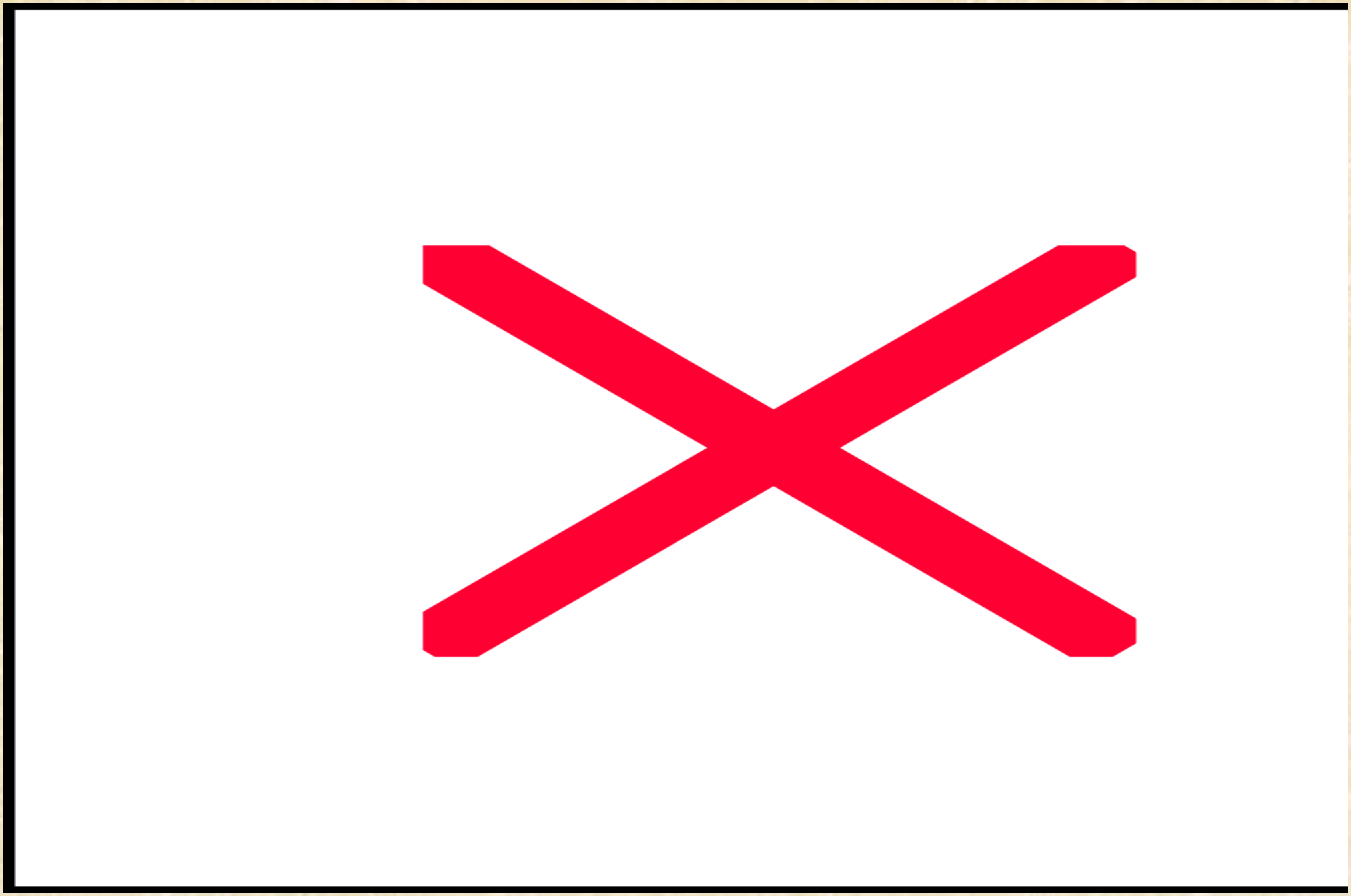
2. Systems that run in real time.

3. Robust behavior in the face of error, the unexpected, and the unknown.

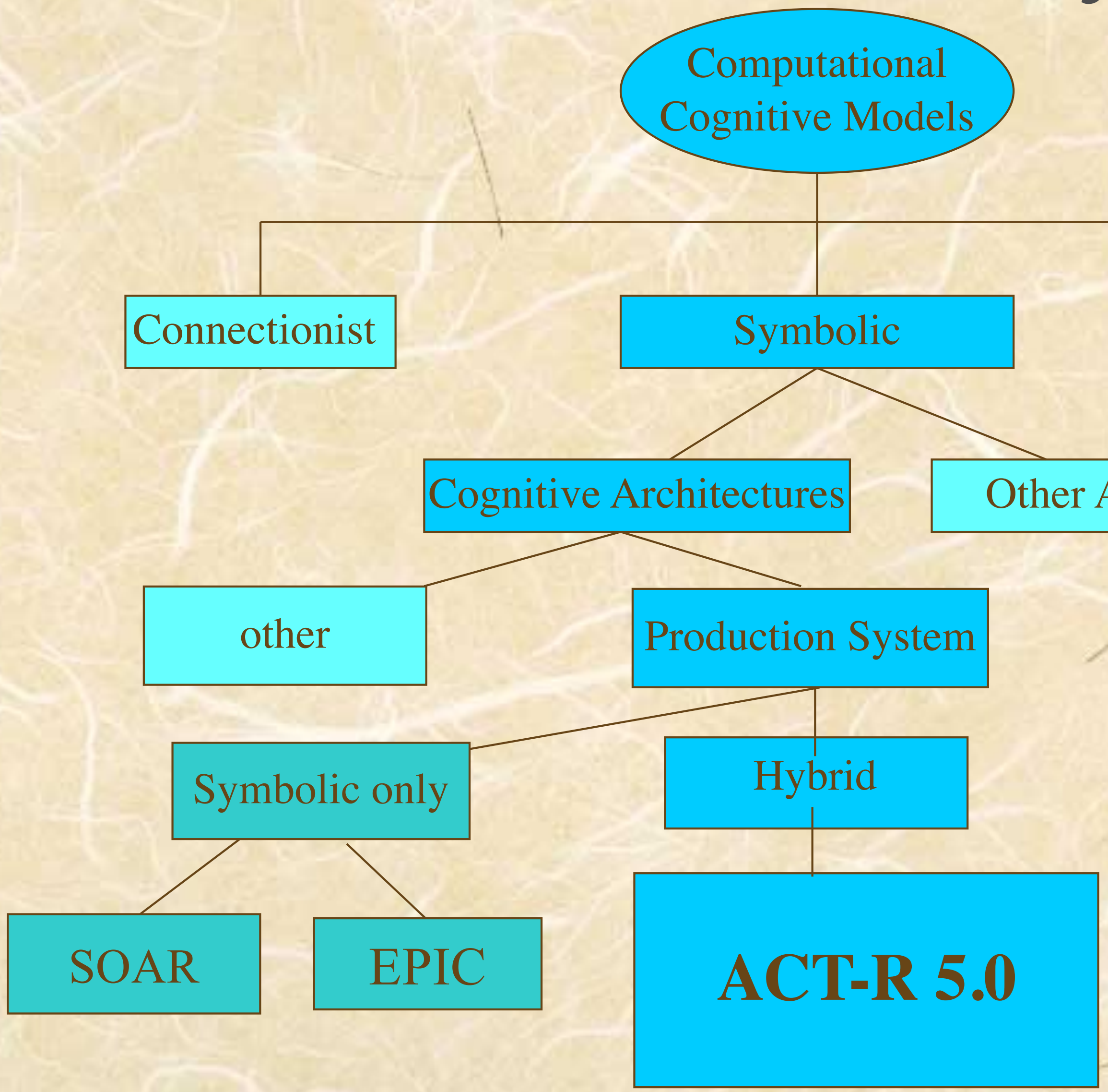
4. Parameter-free predictions of behavior.

5. Learning.

Newell’s Time Scale of Human Activity (amended)



Taxonomy


Computational Cognitive Models
Connectionist
Symbolic
Mathematical
Cognitive Architectures
Other AI
other
Production System
Symbolic only
Hybrid
SOAR
EPIC
ACT-R 5.0

Other Cognitive Architectures

- Soar
  - Production rule system
    - *Organized in terms of operators associated with problem spaces*
    - *Goal oriented*
      - Sub-goaling
    - *Learning mechanism - Chunking*

- EPIC
  - Parallel firing of production rules
  - Well developed visual and motor system

ACT-R Overview

- Modules (buffers)
- Knowledge Representation
- Symbolic/Sub-symbolic
- Performance/Learning

History of the ACT-framework*

Predecessor	HAM	(Anderson & Bower 1973)
Theory versions	ACT-E	(Anderson, 1976)
	ACT*	(Anderson, 1978)
	ACT-R	(Anderson, 1993)
	ACT-R 4.0	(Anderson & Lebiere, 1998)
	ACT-R 5.0	(Anderson & Lebiere, 2001)
Implementations	GRAPES	(Sauers & Farrell, 1982)
	PUPS	(Anderson & Thompson, 1989)
	ACT-R 2.0	(Lebiere & Kushmerick, 1993)
	ACT-R 3.0	(Lebiere, 1995)
	ACT-R 4.0	(Lebiere, 1998)
	ACT-R/PM	(Byrne, 1998)
	ACT-R 5.0	(Lebiere, 2001)
	Windows Environment	(Bothell, 2001)
	Macintosh Environment	(Fincham, 2001)

~ 100 Published Models in ACT-R 1997-2002*

I. Perception & Attention
1. Psychophysical Judgements
2. Visual Search
3. Eye Movements
4. Psychological Refractory Period
5. Task Switching
6. Subitizing
7. Stroop
8. Driving Behavior
9. Situational Awareness
10. Graphical User Interfaces

II. Learning & Memory
1. List Memory
2. Fan Effect
3. Implicit Learning
4. Skill Acquisition
5. Cognitive Arithmetic
6. Category Learning
7. Learning by Exploration and Demonstration
8. Updating Memory & Prospective Memory
9. Causal Learning

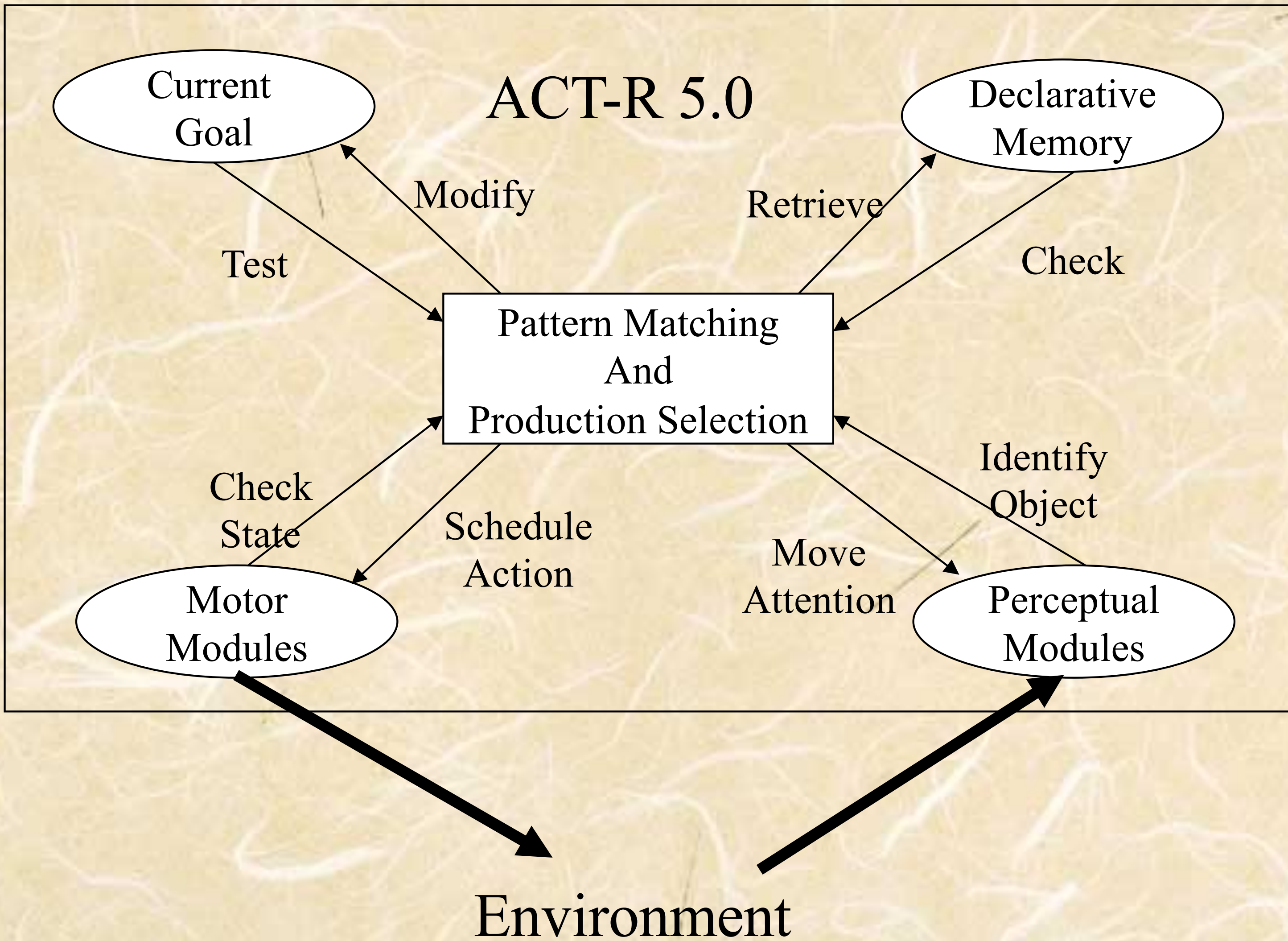
III. Problem Solving & Decision Making
1. Tower of Hanoi
2. Choice & Strategy Selection
3. Mathematical Problem Solving
4. Spatial Reasoning
5. Dynamic Systems
6. Use and Design of Artifacts
7. Game Playing
8. Insight and Scientific Discovery

IV. Language Processing
1. Parsing
2. Analogy & Metaphor
3. Learning
4. Sentence Memory

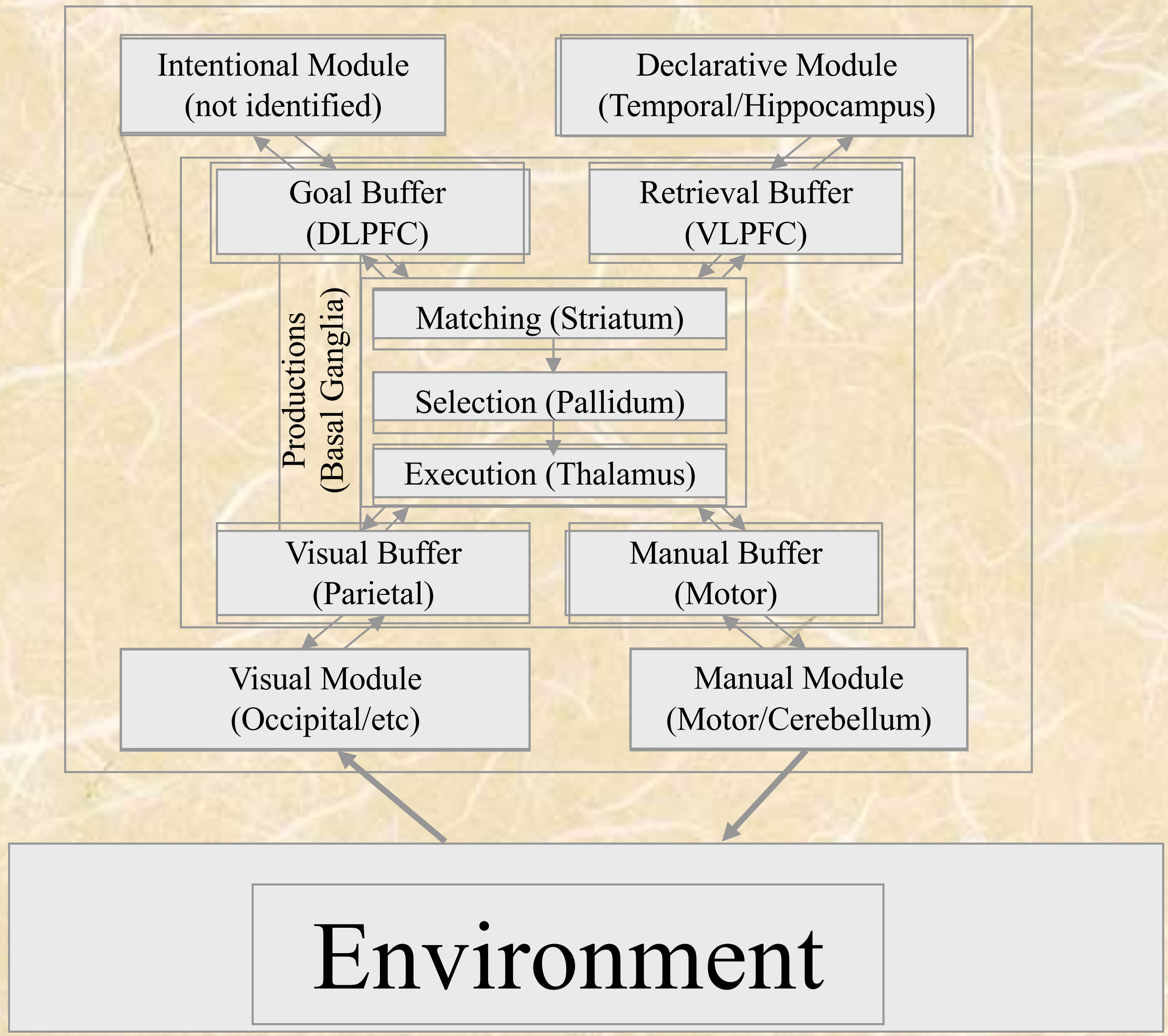
V. Other
1. Cognitive Development
2. Individual Differences
3. Emotion
4. Cognitive Workload
5. Computer Generated Forces
6. fMRI
7. Communication, Negotiation, Group Decision Making

Visit http://act.psy.cmu.edu/papers/ACT-R_Models.htm link.

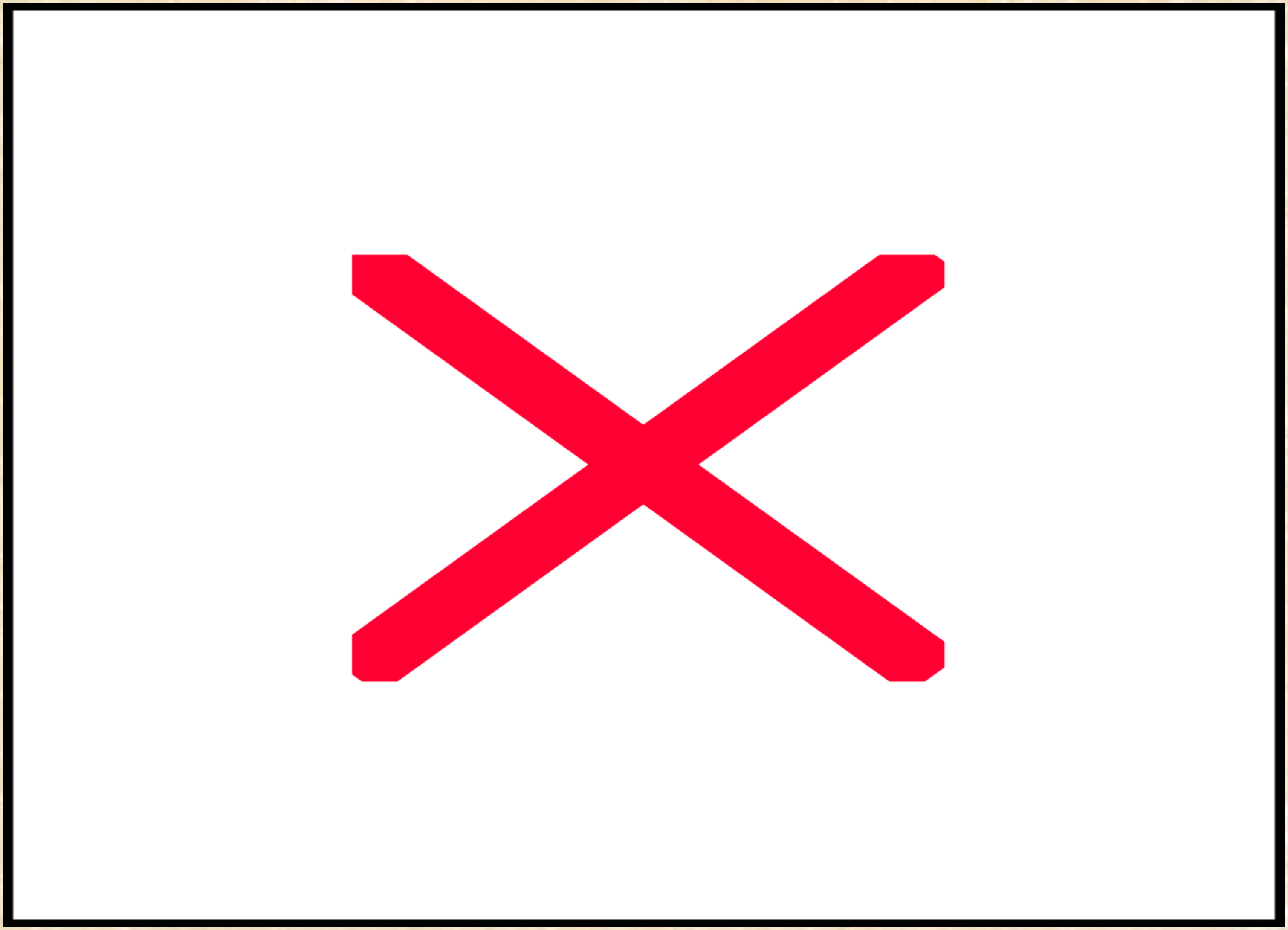
ACT-R 5.0 Architecture



ACT-R 5.0 Mapping to the Brain*



ACT-R: Assumption Space*



Interactive Session

- Load and run Addition model

Addition model exercise

In this exercise you will load a simple model and run it to see how a model runs. You will also get some experience with the interface.

1. Open model

Click on the "Open Model" button on the Environment Control Pane, and select the Addition model. This will open up the model so that you can see it and its parts.

You should be able to see the working memory elements in the model (window "Chunk"), the productions (Production window). There are three further windows, Chunk Type, Command, and Miscellaneous, that we will cover later.

You should briefly examine the chunk and production contents. You may note that there about 11 pieces of working memory, and just 4 rules in this system.

2. Run the model

You can run the model using the Lisp command line, but we will use the environment because it provides a recognition-based interface rather than a recall-based interface.

You should first click on "Reset"; thi s will reset the model and make it ready to run. You can do this to a model that has run as well, or has been stopped in the middel of a run.

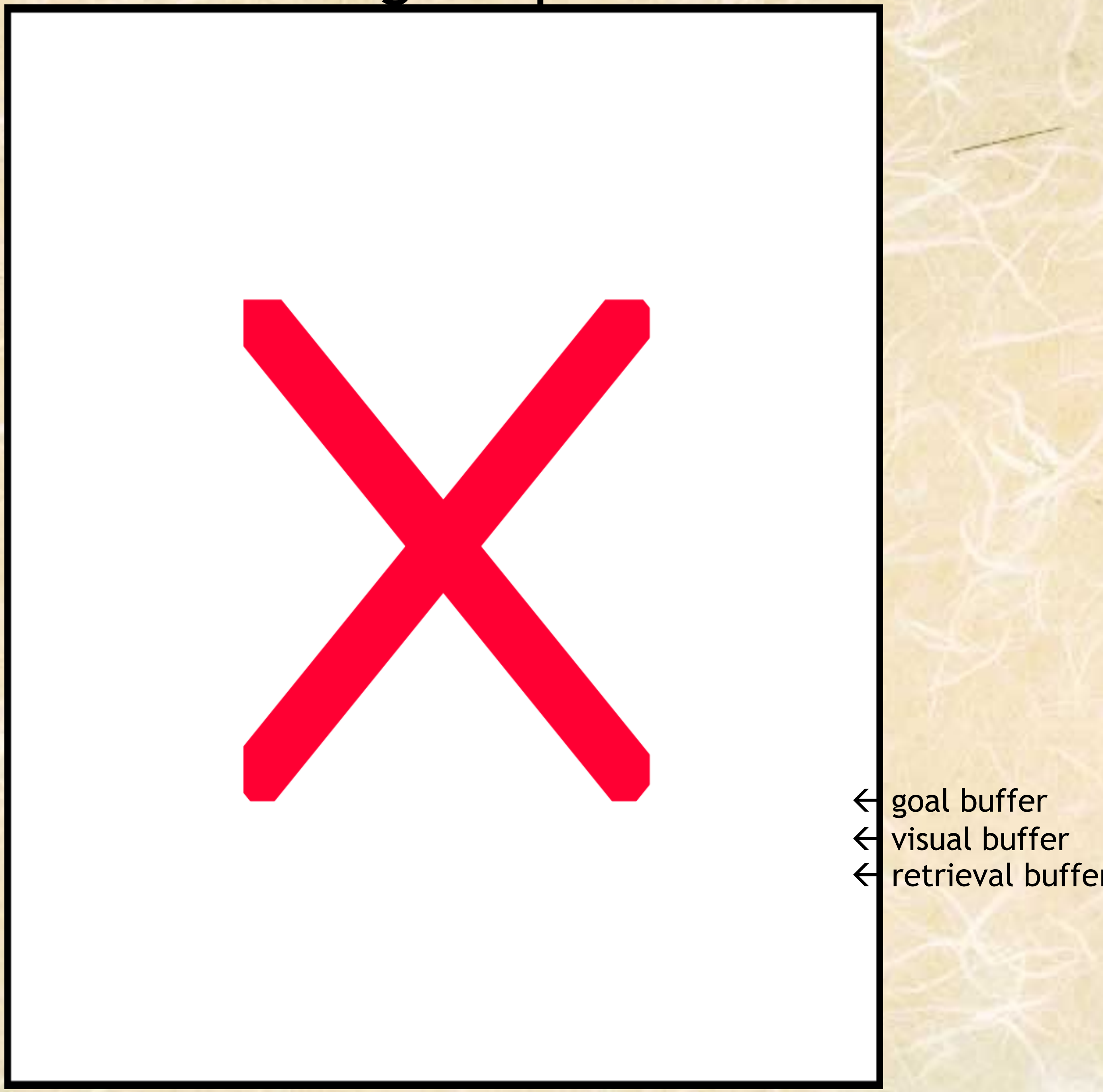
You can run the model by clicking on the "Run" button. A trace of the model will appear in the (Lisp) "Listener" window. You can see how the order that rules are selected and fired, as well as when chunks are retrieved from memory by the rules.

3. Inspect the model

Click on "Declarative viewer" in the Control Pane to bring up an inspector window for the declarative memory elements. If you scroll, you can find the chunks a-j and second-goal. Pay most attention to their structure, and note that they have several parameters. These parameters are used to compute how fast they are used and if they can be retrieved. With learning and use, the activation, for example, goes up. These are covered later in this tutorial.

The Procedural viewer provides a view onto the rules.

ACT-R: Knowledge Representation*



Chunks: Example*

(CHUNK-TYPE NAME SLOT1 SLOT2 SLOTN)

(FACT3+4

isa ADDITION-FACT

ADDEND1 THREE

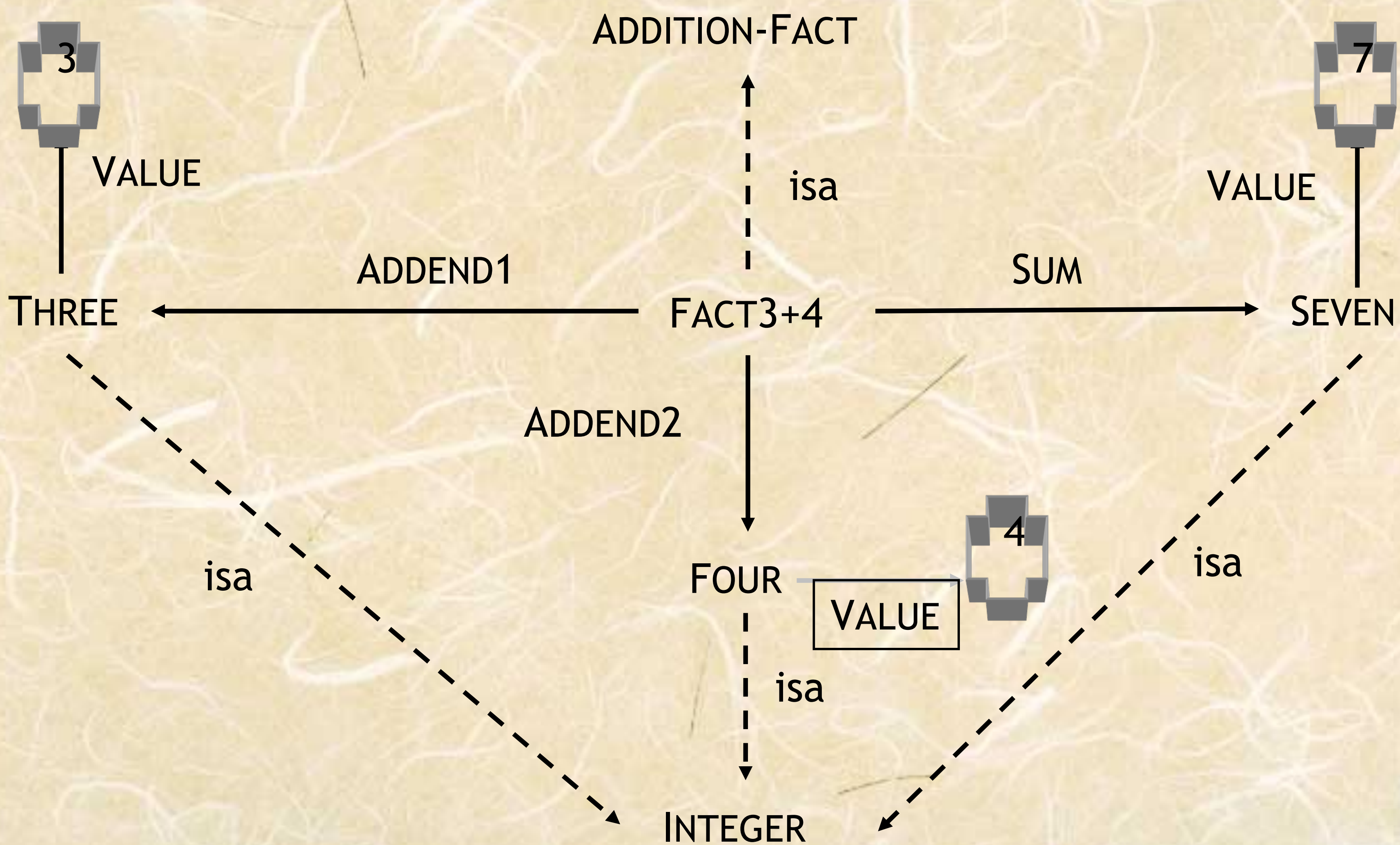
ADDEND2 FOUR

SUM SEVEN)

Chunks: Example*

```
(CLEAR-ALL)
(CHUNK-TYPE addition-fact addend1 addend2 sum)
(CHUNK-TYPE integer value)
(ADD-DM (fact3+4
                isa addition-fact
                addend1 three
                addend2 four
                sum seven)
        (three
                isa integer
                value 3)
        (four
                isa integer
                value 4)
        (seven
                isa integer
                value 7)
```

Chunks: Example*



A Production is*

1. The greatest idea in cognitive science.

2. The least appreciated construct in cognitive science.

3. A 50 millisecond step of cognition.

4. The source of the serial bottleneck in otherwise parallel system.

5. A condition-action data structure with “variables”.

6. A formal specification of the flow of information from cortex to basal ganglia and back again.

Productions*

Key Properties ⇒

- modularity
- abstraction
- goal/buffer factoring
- conditional asymmetry

Structure of productions

	(p name
condition part	Specification of Buffer Tests
delimiter	==>
action part	Specification of Buffer Transformations
	)

Interactive Session

- Load and run Counting model

Count model

This model works much like the previous model, but prints out its count.

1. Open the model

Either quit and restart your Lisp, or else click on "Close Model".

Open the Count model by clicking on "Open Model" and then selecting the Count model.

Run the model to see its trace, and examine its rules and chunks.

2. Using the Stepper

Click on "Stepper", and a stepper window should appear.

Reset the model, and then click on the run button. This starts the stepper. You can now step through the model by clicking on the "Step" button on the Stepper.

As you step through the model, you should be able to see most of the mechanisms in ACT-R now, the productions, how they are matched, the chunks, and how they are retrieved, and the buffers (click on Buffer Viewer to see the buffers and their contents).

- 3. Checking on a rule that does not fire.
- After you have run the model a few steps, click on the Procedural Viewer. Select a rule in the dialogue box, and see why it does not fire.
- 4. Edit the model
- Look at the model and consider how to have it count backwards.
- You can change the production rules in the Production window. After you make changes, save the model (it will automatically increment). Close the model and reopen it to try your new model.

The Modules(reprise)

- Cognition
- Memory
- Vision
- Motor
- Audition
- Speech

ACT-R 5.0 Buffers*

1. Goal Buffer (=goal, +goal)
   -represents where one is in the task
   -preserves information across production cycles

2. Retrieval Buffer (=retrieval, +retrieval)
   -holds information retrieval from declarative memory
   -seat of activation computations
3. Visual Buffers
   -location (=visual-location, +visual-location)
   -visual objects (=visual, +visual)
   -attention switch corresponds to buffer transformation
4. Auditory Buffers (=aural, +aural)
   -analogous to visual
5. Manual Buffers (=manual, +manual)
   -elaborate theory of manual movement include feature preparation, Fitts law, and device properties
6. Vocal Buffers (=vocal, +vocal)
   -analogous to manual buffers but less well developed

Cognition

- Executive Control - Production System
- Serial
- Parallel at sub-symbolic level
  - Utility selects production to fire
  - Utility = benefit - cost
    - *Benefit = probability of success * value of achieving goal*

Production System Cycle

- Match conditions of all rules to buffers
- Those that match enter the conflict set
- Conflict resolution selects a rule to fire
- Action side of rule initiates changes to one or more buffers
- If no production can match and no action is in progress then quit else repeat

Goal directed

- Represents what you are trying to do
- A declarative memory element that is the focus of “internal” attention

Memory Module

- Activation based
  - Frequency and recency
  - Contextual cues
- Cognition
  - Requests retrieval
    - *Specifies constraints*
    - *Partial matching*
- Memory
  - Parallel search of memory to match constraints
  - Calculates activation of matching chunks
  - Returns most active chunk

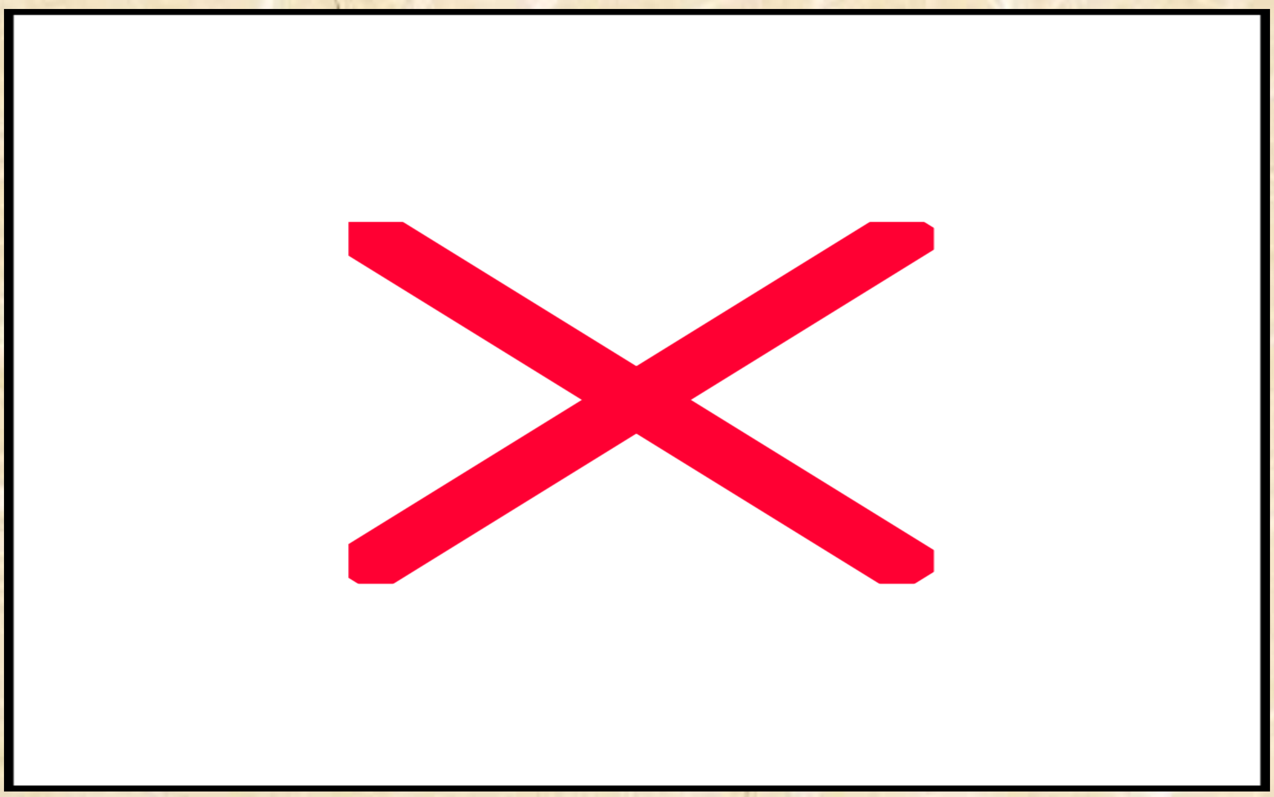
Vision Module

- ACT-R's "eyes"
- Dorsal "where" system
- Ventral "what" system

“Where” System

- Cognition
  - Requests “pre-attentive” visual search
  - Specifies a set of constraints
  - Attribute/value pairs
    - *Properties or spatial location*
      - e.g. color red, screen-x greater-than 150
- “Where” system
  - Returns a “location” chunk
    - *Specifies location of an object whose features satisfy the constraints*
- Onsets
- Features are held in vision module’s memory

Vision Module Memory



“What” System

- Cognition
  - Requests “move attention”
  - Provides “location” chunk
- “Where” System
  - Shifts visual attention to that location
  - Encodes object at that location
    - *Added to Declarative Memory*
    - *Episodic representation of visual scene*
  - Places encoding in “visual” buffer
  - Calculates latency
    - *EMMA*

Motor Module

- ACT-R's Hands
- Based on EPIC's Manual Motor Processor
- Movement Styles
- Phased Processing

Movement Styles

- Ply - moves a device (e.g. mouse) to a given location
- Punch - pressing a key below finger
- Peck - directed movement of finger to a location followed by keystroke
- Peck-recoil - same as peck but finger moves back
- Point-hand - moves hand to a new location

Phased Processing (1)

- Preparation Phase
  - Hierarchical feature preparation
    - *Style->hand->finger*
  - Prep time depends on
    - *Complexity of movement*
    - *Number of features*
  - State buffer set to prep busy

Phased Processing (2)

- Initiation (fixed 50 ms)
- Execution
  - *Time depends on*
    - Type of movement
      - Minimum execution time
    - Distance
      - Fitt's Law
- Allow overlapping of preparation and execution

Interactive Session

- Load and run Letter model

Device Interface

- Simulated device with which ACT-R interacts
  - Contains graphical objects
- Typically a Window
  - Can be entire screen
- Interaction
  - Constructing vision system's iconic memory (sets of features) from graphical objects
  - Handle mouse and keyboard actions

Audition Module

- Simulated perception of audio
- Memory of features
  - Temporal-extent - sound events
- Tones, digits, and speech
- Attributes
  - Onset, duration, delay, recode time

Audition Module Processing

- Parallels vision system
- Cognition
  - Specifies a set of constraints
  - Attribute/value pairs
- Audition
  - Returns a "location" chunk
- Cognition
  - Requests shift of auditory attention providing the "location"chunk
- Audition
  - Encodes the sound

Sub-symbolic level

- Sub-symbolic learning allow the system to adapt to the statistical structure of the environment
- Production Utilities are responsible for determining which productions get selected when there is a conflict.
- Chunk Activations are responsible for determining which (if any chunks) get retrieved and how long it takes to retrieve them.
- Chunk Activations have been simplified in ACT-R 5.0 and a major step has been taken towards the goal of parameter-free predictions by fixing a number of the parameters.

Parameters

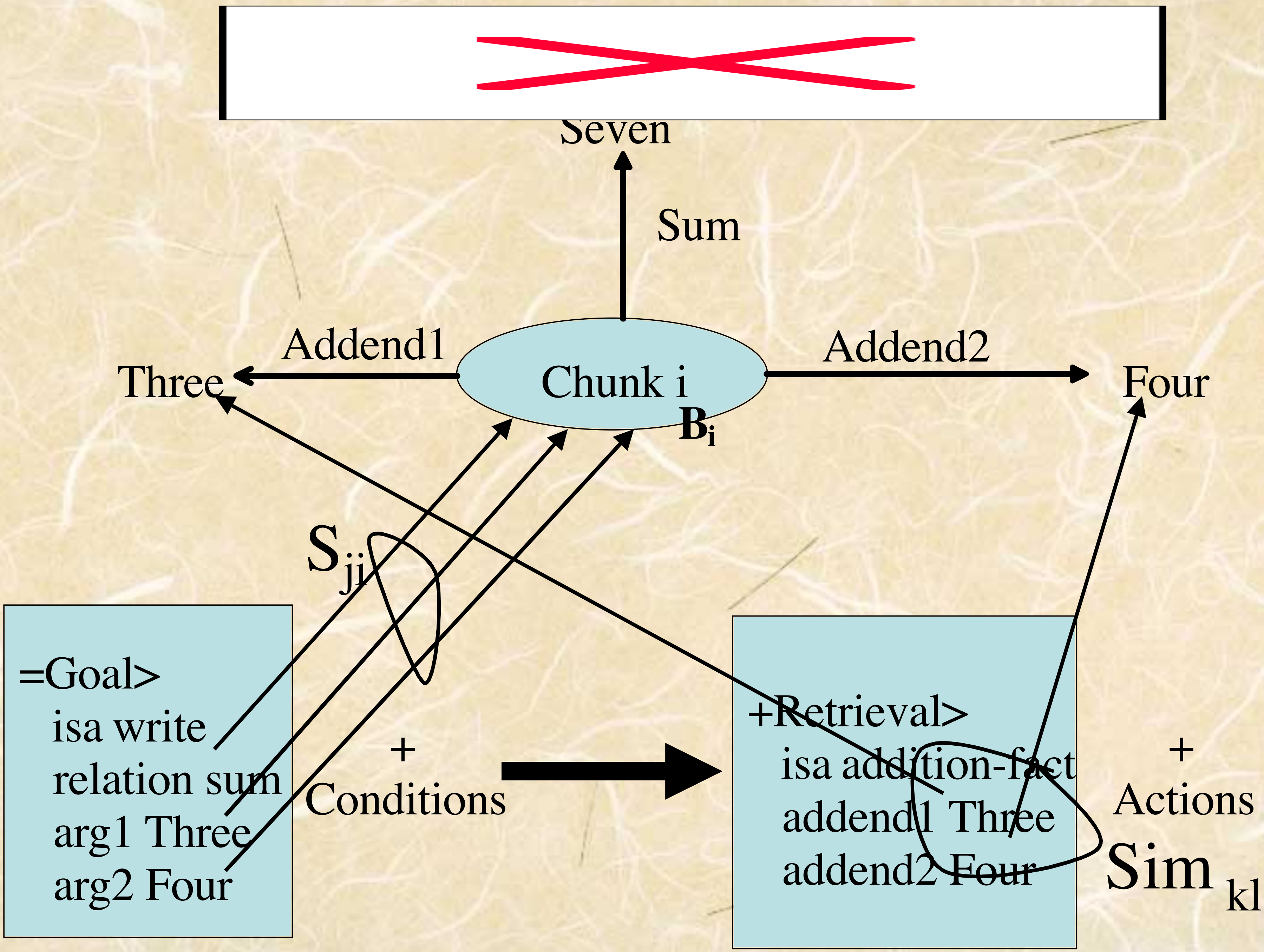
- Noise
  - Utility and activation
- Learning
  - Activation - frequency and recency
  - Utility - probability and cost
- Thresholds
  - Utility and activation

Build Dialing Model

Detailed ACT-R theory

- Activation equation
- Production Utility equation
- Production Compilation

Activation*



Seven
Sum
Addend1
Three
Chunk i
B_i
Addend2
Four
S_{ji}
=Goal>
isa write
relation sum
arg1 Three
arg2 Four
+
Conditions
+Retrieval>
isa addition-fact
addend1 Three
addend2 Four
+
Actions
Sim_{kl}

Chunk Activation*

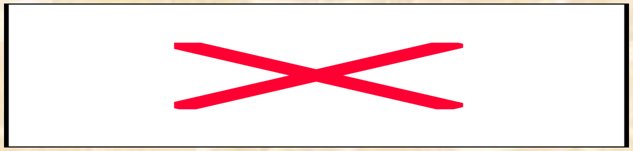
$$\text{activation} = \text{base activation} + \left(\text{source activation} * \text{associative strength}\right) + \left(\text{mismatch penalty} * \text{similarity value}\right) + \text{noise}$$

Activation makes chunks available to the degree that past experiences indicate that they will be useful at the particular moment:

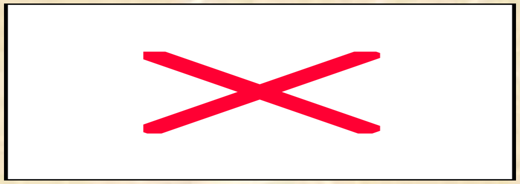
- Base-level: general past usefulness
- Associative Activation: relevance to the general context
- Matching Penalty: relevance to the specific match required
- Noise: stochastic is useful to avoid getting stuck in local minima

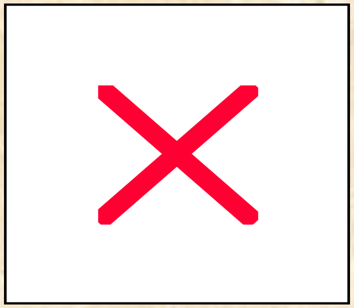
Activation, Latency and Probability*

- Retrieval time for a chunk is a negative exponential function of its activation:



- Probability of retrieval of a chunk follows the Boltzmann (softmax) distribution:





- The chunk with the highest activation is retrieved provided that it reaches the retrieval threshold τ
- For purposes of latency and probability, the threshold can be considered as a virtual chunk

Base-level Activation*

activation = base activation

$$A_i = B_i$$

The base level activation B_i of chunk C_i reflects a context-independent estimation of how likely C_i is to match a production, i.e. B_i is an estimate of the log odds that C_i will be used.

Two factors determine B_i:

- frequency of using C_i
- recency with which C_i was used

$$B_i = \ln\left(\frac{P(C_i)}{P(\overline{C_i})}\right)$$

Source Activation*

$$+ \left(\text{source activation} \quad * \quad \text{associative strength} \right)$$

$$+ \sum_j W_j \quad * \quad S_{ji}$$

The source activations W_j reflect the amount of *attention* given to elements, i.e. fillers, of the current goal. ACT-R assumes a *fixed capacity* for source activation

$W = \Sigma W_j$ reflects an individual difference parameter.

Associative Strengths*

$$+ \left(\text{source activation} \quad * \quad \text{associative strength} \right)$$

$$+ \quad \sum W_j \quad * \quad S_{ji}$$

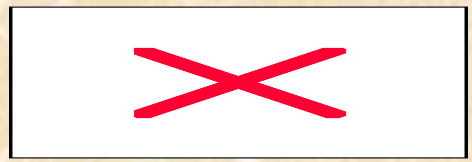
The association strength S_{ji} between chunks C_j and C_i is a measure of how often C_i was needed (retrieved) when C_j was element of the goal, i.e. S_{ji} estimates the log likelihood ratio of C_j being a source of activation if C_i was retrieved.

$$S_{ji} = \ln \left(\frac{P(N_i | C_j)}{P(N_i)} \right)$$

$$= S - \ln(P(N_i | C_j))$$

Partial Matching*

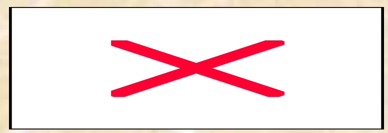
$$+\left(\text{mismatch penalty} * \text{similarity value}\right)$$



- The mismatch penalty is a measure of the amount of control over memory retrieval: MP = 0 is free association; MP very large means perfect matching; intermediate values allow some mismatching in search of a memory match.
- Similarity values between desired value k specified by the production and actual value l present in the retrieved chunk. This provides generalization properties similar to those in neural networks; the similarity value is essentially equivalent to the dot-product between distributed representations.

Noise*

+ noise



- Noise provides the essential stochasticity of human behavior
- Noise also provides a powerful way of exploring the world
- Activation noise is composed of two noises:
  - A permanent noise accounting for encoding variability
  - A transient noise for moment-to-moment variation

Base-Level Learning*

Based on the Rational Analysis of the Environment
(Schooler & Anderson, 1997)

Base-Level Activation reflects the log-odds that a chunk will be needed. In the environment the odds that a fact will be needed decays as a power function of how long it has been since it has been used. The effects of multiple uses sum in determining the odds of being used.

Base-Level Learning Equation

$\approx n(n / (1-d)) - d*n(L)$

Note: The decay parameter d has been set to .5 in most ACT-R models

Interactive Session

- Load and run Sternberg model

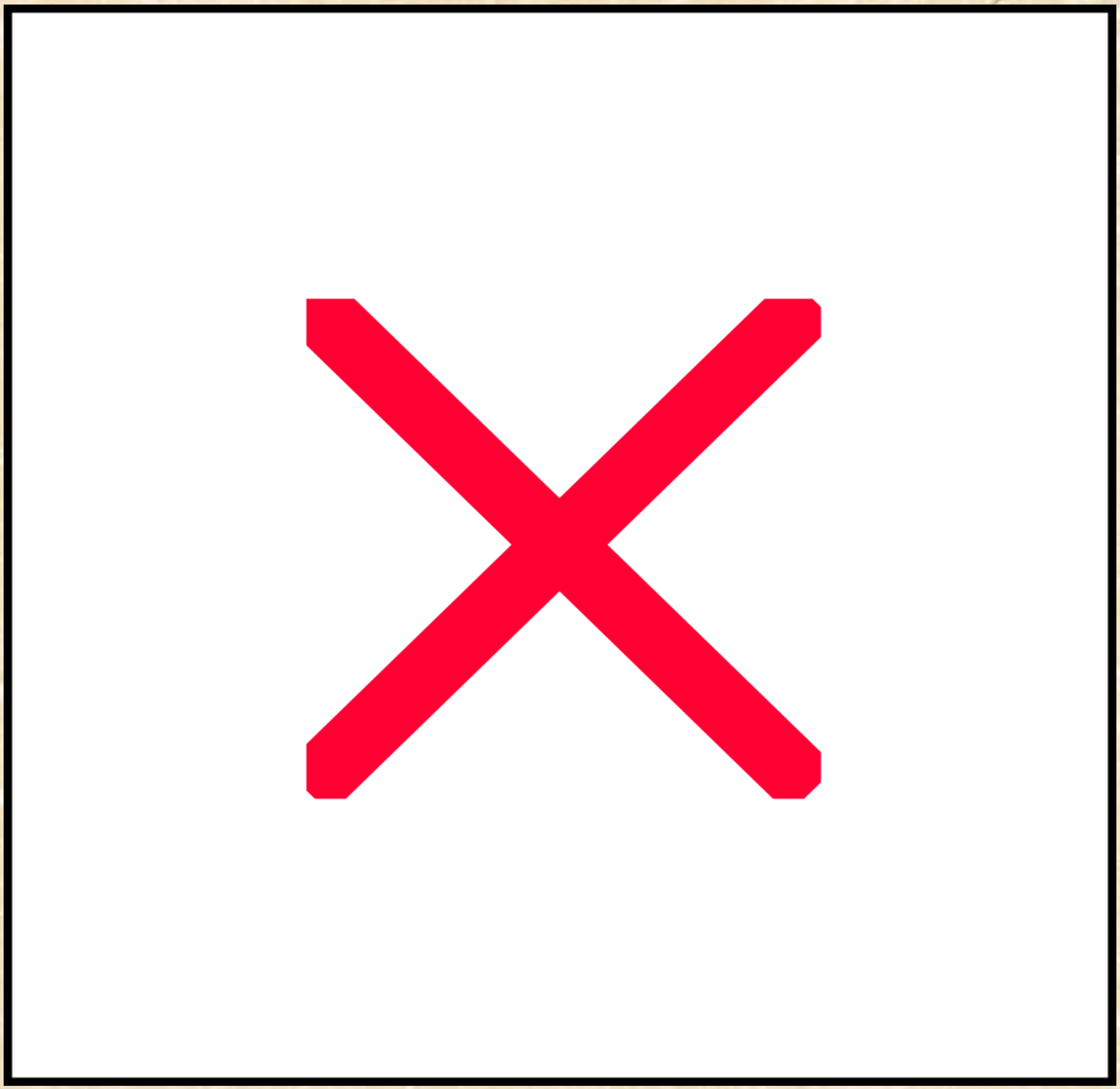
Production Utility*

P is expected probability of success
G is value of goal
C is expected cost

t reflects noise in evaluation
and is like temperature in
the Bolztman equation

α is prior successes
m is experienced successes
β is prior failures
n is experienced failures

Decay of Experience*



Note: Such temporal weighting is critical in the real world.

Interactive Session

- Load and run Building Sticks model

Production Compilation: The Basic Idea*

```
(p read-stimulus
   =goal>
      isa goal
      step attending
      state test
   =visual>
      isa text
      value =val
==>
   +retrieval>
      isa goal
      relation associate
      arg1 =val
      arg2 =ans
   =goal>
      relation associate
      arg1 =val
      step testing)

(p recall
   =goal>
      isa goal
      relation associate
      arg1 =val
      step testing
   =retrieval>
      isa goal
      relation associate
      arg1 =val
      arg2 =ans
==>
   +manual>
      isa press-key
      key =ans
   =goal>
      step waiting)

(p recall-vanilla
   =goal>
      isa goal
      step attending
      state test
   =visual>
      isa text
      value "vanilla
==>
   +manual>
      isa press-key
      key "7"
   =goal>
      relation associate
      arg1 "vanilla"
      step waiting)
```

Production Compilation: The Principles*

1. **Perceptual-Motor Buffers:** Avoid compositions that will result in jamming when one tries to build two operations on the same buffer into the same production.

2. **Retrieval Buffer:** Except for failure tests proceduralize out and build more specific productions.

3. **Goal Buffers:** Complex Rules describing merging.

4. **Safe Productions:** Production will not produce any result that the original productions did not produce.

5. **Parameter Setting:**
Successes = P***initial-experience***
Failures = (1-P) ***initial-experience***
Efforts = (Successes + Efforts)(C + *cost-penalty*)

Future Directions

- ACT-R 6.0
  - Design goals
    - *More modular*
    - *Consistent and uniform syntax*
    - *Consistent treatment of buffers*
    - *Parameter simplification*
- Model behavior in more complex real world environments

More Information

- ACT-R Home Page: http://act.psy.cmu.edu

Acknowledgements

- Support for this tutorial was provided by grants from the US Office of Naval Research, # N00014-03-1-0248 , and by the Naval Warfare Systems Center, # N6600-01-1-8916.