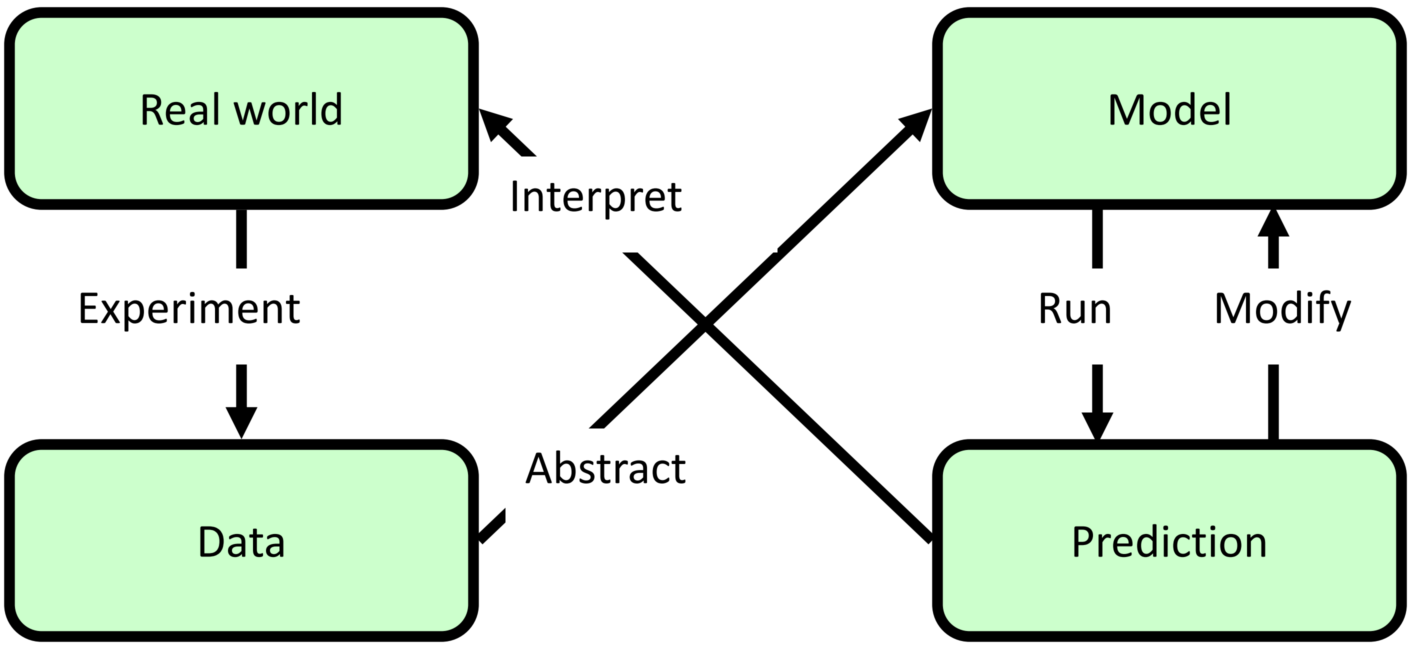
Notes on teaching ACT-R modeling

Bill Kennedy
George Mason University
September 2011

Why Model?



From Marvin L. Bittinger(2004) Calculus and Its Applications, 8^{th} Ed.

Why Model?

1. Explain (very distinct from predict)
2. Guide data collection
3. Illuminate core dynamics
4. Suggest dynamical analogies
5. Discover new questions
6. Promote a scientific habit of mind
7. Bound outcomes to plausible ranges
8. Illuminate core uncertainties
9. Offer crisis options in near-real time
10. Demonstrate tradeoffs / suggest efficiencies
11. Challenge the robustness of prevailing theory through perturbations
12. Expose prevailing wisdom as incompatible with available data
13. Train practitioners
14. Discipline the policy dialogue
15. Educate the general public
16. Reveal the apparently simple (complex) to be complex (simple)

Joshua M. Epstein (2008) JASSS

Cognitive Science

- An interdisciplinary field including parts of Anthropology, Artificial Intelligence, Education, Linguistics, Neuroscience, Philosophy, and Psychology

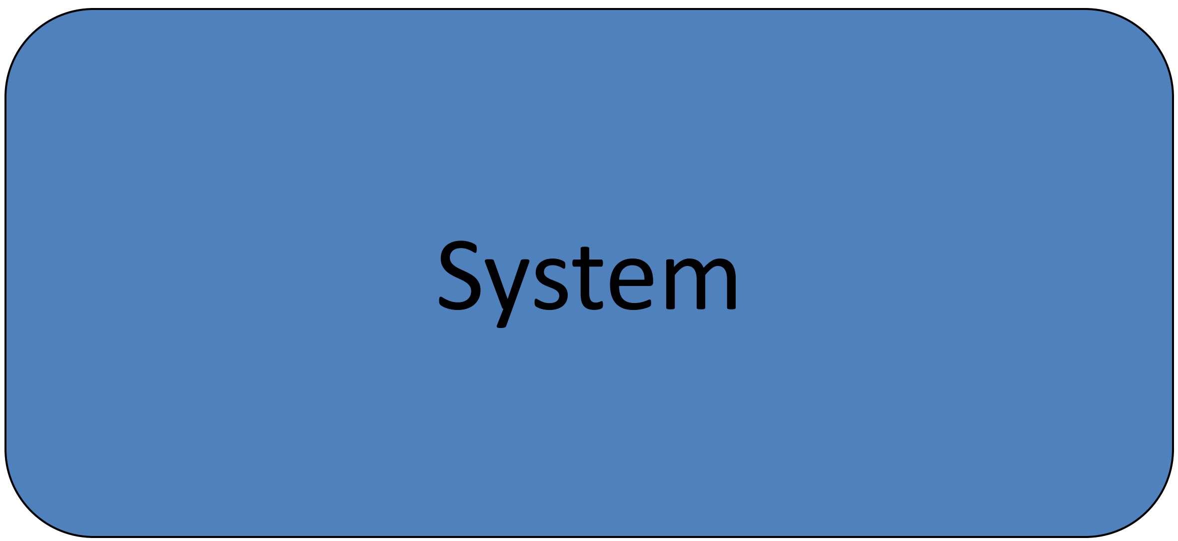
- Goal: understanding the nature of the human mind

Goals of AI & Cog Sci

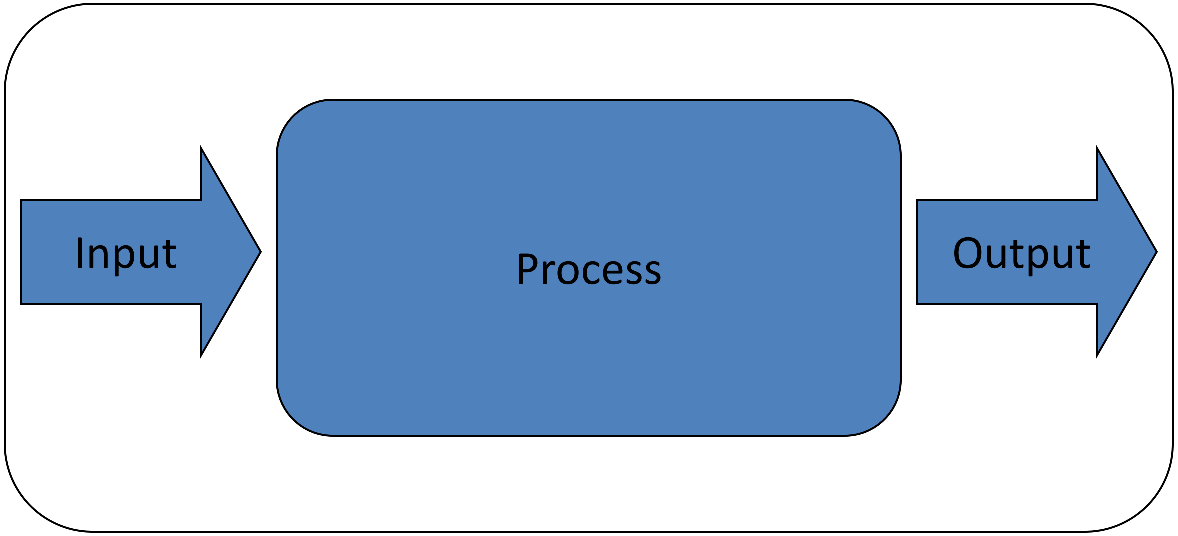
Artificial Intelligence: "scientific understanding of the mechanisms underlying thought and intelligent behavior and their embodiment in machines" (from AAAI website):
demonstrated by functionality

Cognitive Science: understanding the nature of the human mind
as demonstrated by models that match human behavior: i.e., matching human behavior

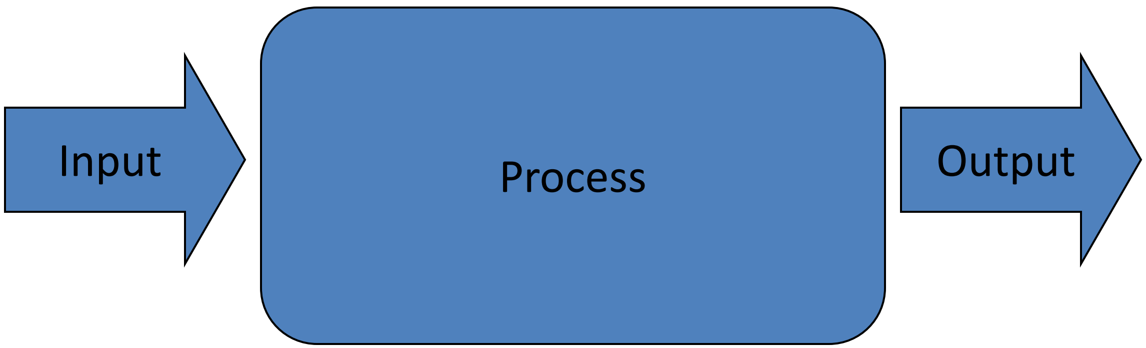
What is it?



What is it?

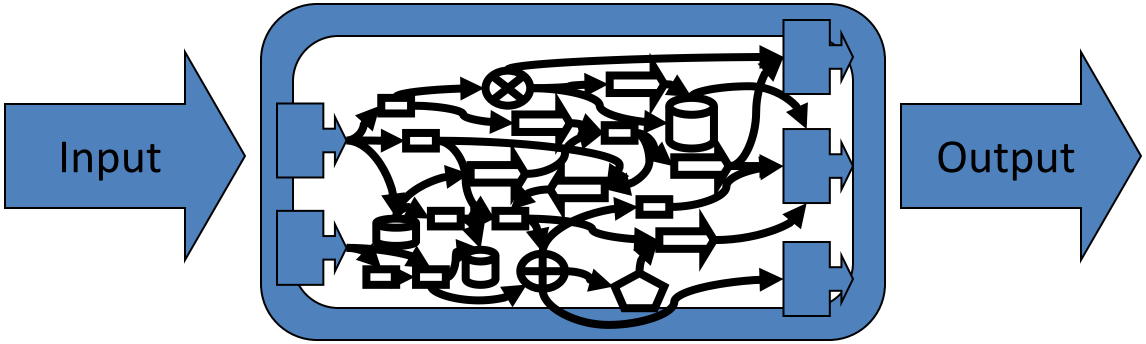


What is it?



Perception Cognition Action

What is it?



Perception Cognition Action

Components

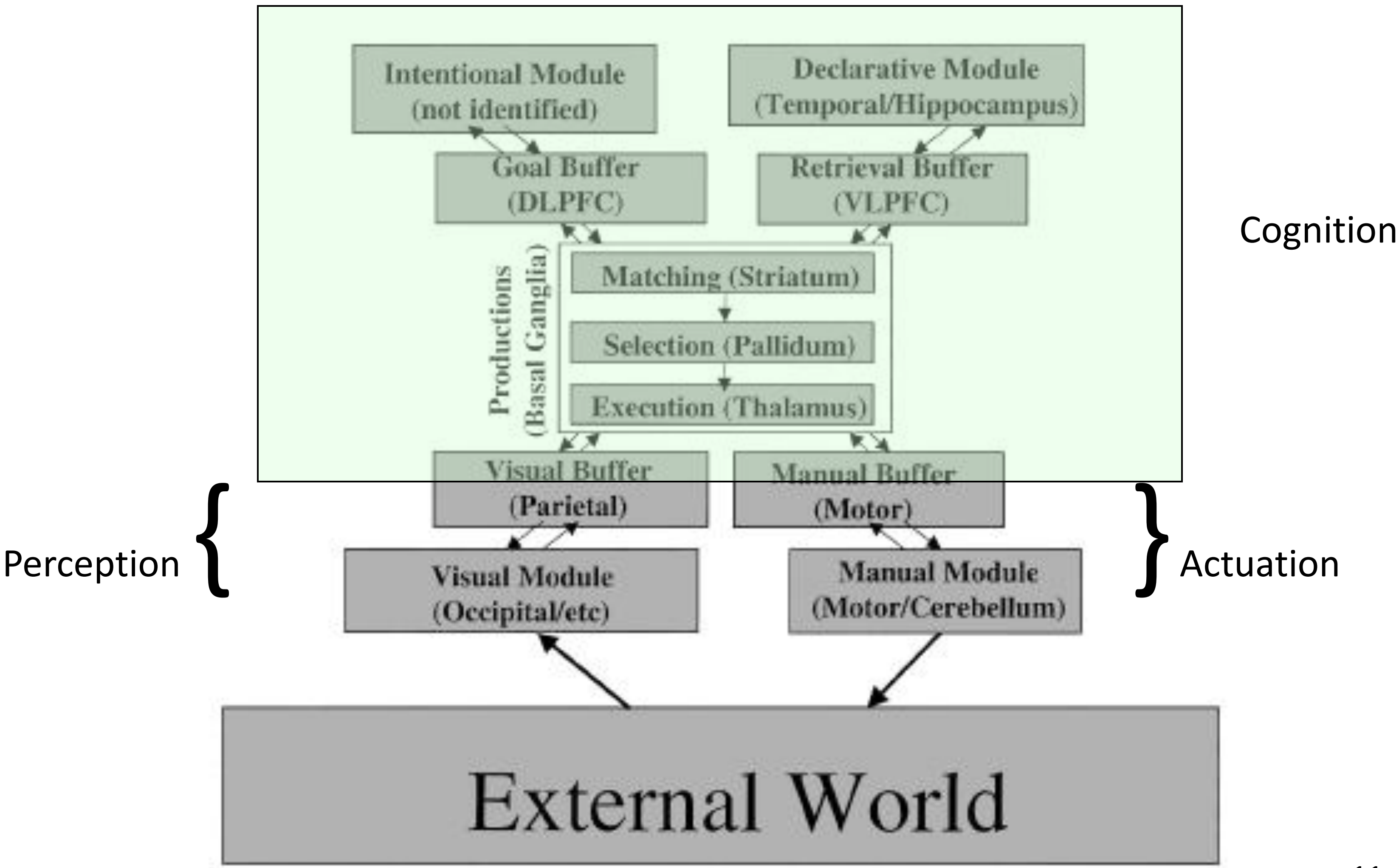
- An **architecture** is the fixed* part of a system that implements the mind functions
- Associated with an architecture, a **model** is the variable* part of a the mind's processing

------

- System's knowledge representation is on the edge between architectures and models

* with respect to a time scale (from Newell's 1990 UTC)

ACT-R Architecture



ACT-R Architecture Overview

- Procedure core
- Buffers
- Modules
- 2 types of knowledge representation:
  - declarative (facts, “chunks”)
  - procedural (deductions, “productions”)

Productions

- Procedural knowledge as if-then statements
- Basic process is: match, select, fire
- Many may match current status
- Only 1 selected to fire

Productions

- Productions use buffers in IF & THEN parts
- IF part checks buffer contents or status
- THEN part, changes buffer values or requests buffer/module action

Productions

- Useful to translate English to ACT-R

- eg: IF the goal is to count to y AND currently at x, AND x<>y, THEN remember what comes after x.

Production Design

- eg 1: IF the goal is to count to y AND currently at x, AND x<>y, THEN remember what comes after x.
- but:
  - this production will always match and fire...
  - another production will deal with the remembered fact
  - it can work with addition of a “state” variable

Production Design

- IF the goal is to count to y AND currently at x, AND x<>y, THEN remember what comes after y.

```
(p rule-getnext
  =goal>
     isa          count
     to           y
     current      x
   - current      y
   - state        recalling-next
  ==>
  +retrieval>
     isa          next-fact
     current      x
  =goal>
    state         recalling-next
)
```

count chunk type:
to <n>
current <m>
state <w>

Production Design 2

- eg 1: IF the goal is to count to y AND currently at x, AND x<>y, THEN recall what comes after y.

- eg 2: IF the goal is to count with current = x AND “to” is not x, THEN recall what comes after x

- eg 3: IF the goal is to count with current is x AND “to” is not x, and we remembered y comes after x, THEN update current to y and recall what comes after y

Production Design (core)

```
(P increment
  =goal>
    ISA        count-from
    count       =num1
   - end       =num1
  =retrieval>
    ISA       count-order
    first       =num1
    second   =num2
 ==>
  =goal>
    count       =num2
  +retrieval>
    ISA        count-order
    first        =num2
  !output!      (=num1)
)
```

count-from chunk type:
  end <n>
  count <m>

count-order chunk type:
  first <n>
  second <m>

Production Design (start)

```
(P start
  =goal>
     ISA       count-from
     start     =num1
     count      nil
 ==>
  =goal>
     count      =num1
  +retrieval>
     ISA       count-order
     first     =num1
)
```

Production Design (stop)

```
(P stop
  =goal>
    ISA       count-from
    count     =num
    end       =num
==>
  -goal>
  !output!      (=num)
)
```

ACT-R & Lisp...

- ACT-R written in Lisp
- ACT-R uses Lisp syntax
- Parts of a model
  - Lisp code (~)
  - Parameters
  - Initialization of memory (declarative & proc)
  - Running a model

ACT-R & Lisp...syntax

- ; comments
- “(“ <function-name> <arguments> “)”

  eg: (clear-all)

  (sgp)   <= lists all parameters & settings

  (p ...) <= p function creates productions

ACT-R & Lisp…warnings/errors

- Lisp warnings

```
#|Warning: Creating chunk BUZZ of default type chunk |#
```

Undefined term, usually insignificant

```
#|Warning: Invalid chunk definition: (RED
                                      ISA
                                      CHUNK) names a
  chunk which already exists. |#
```

Some terms defined within ACT-R as chunks (~reserved words)

ACT-R & Lisp...warnings/errors

- Lisp /ACT-R error example 1:

```
> (help)
UNDEFINED-FUNCTION
Error executing command: "(help)":

Error:attempt to call `HELP' which is an undefined
  function..
```

Non-existent function call

ACT-R & Lisp...warnings/errors

- Lisp /ACT-R error example 2:

```
Error Reloading:
; loading
;    c:\documents and settings\bill
kennedy\desktop\psyc-768-s09\demo2.txt
error reloading model
error:eof encountered on stream
#<file-simple-stream
  #p"c:\\documents and settings\\bill
kennedy\\desktop\\psyc-768-s09\\demo2.txt" closed
  @ #x20b2159a>
```

Unbalanced parentheses.

ACT-R Model (outline)

```
; header info
(clear-all)
(define-model <model name>
     (sgp :<parm name> <value> <parm name> <value> ... )
     (chunk-type <isa name> <att1> <att2> ...)
     (add-dm
       (<name> isa <chunk-type> <attn> <value>
                             <attm> <value> ...)
       (<name> isa <chunk-type> <attn> <value>
                             <attm> <value> ...)
       ...
      ) ; end of add-dm
     (p ... )
     (goal-focus <chunk-name>)
) ; end of model definition
```

ACT-R Model

```
(p <production name>
   =goal>
      ISA    <chunk-type>
      <att> <value>
      ...
    =retrieval>                    ← buffer (content)
      ISA    <chunk-type>
    ?retrieval>                    ← buffer (status)
      state full
==>
    =goal>
      <att> <value>
    +retrieval>                    ← request to a module
      ISA    <chunk-type>
      <att> <value>
      ...
    -goal>                         ← explicit clearing of a buffer
    !output!      (text text =variable text =variable)
)
```

Homework: Data Fitting

- From now on the assignments will be compared to human performance
  - Mostly Response time
    - Correlation and Mean deviation
- Provides a way to compare and judge the models
- Not the only way!
  - Plausibility
  - Generality
  - Simplicity
- Make sure the model does the right thing before trying to tune it with parameters!

Subitizing

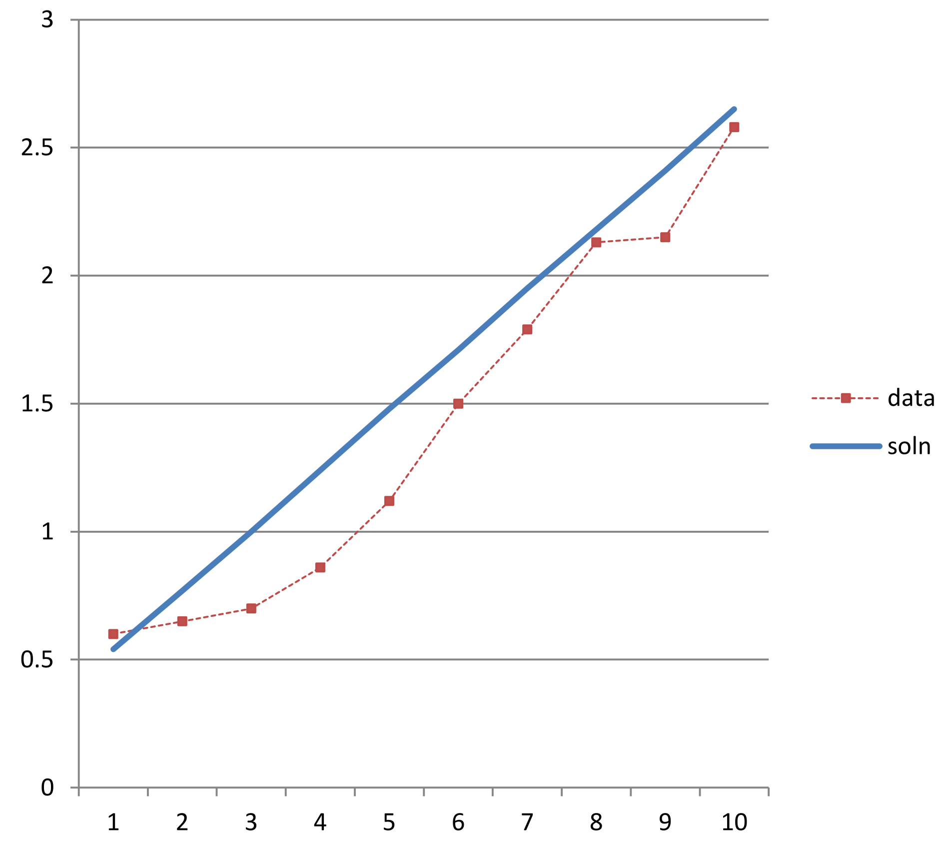
- Task: A bunch of objects appear on the display, report the number by speaking it
- Model starts with the counting facts from 0-11
- Will need to manage visual attention
  - Make sure the model gets to every item
  - Needs to know when its done
  - Given 10 finsts with a long duration to start
    - Do not have to use that if you do not want to
- Should not need to adjust parameters to get a reasonable fit to the data

Solution Model

CORRELATION: 0.980

MEAN DEVIATION: 0.230

Items	Current	Original
1	0.54 (T)	0.60
2	0.77 (T)	0.65
3	1.00 (T)	0.70
4	1.24 (T)	0.86
5	1.48 (T)	1.12
6	1.71 (T)	1.50
7	1.95 (T)	1.79
8	2.18 (T)	2.13
9	2.41 (T)	2.15
10	2.65 (T)	2.58



Mechanism

- Find, attend, count, repeat
- Linear in the number of items
- Any other solutions?

Memory's Subsymbolic Representation

33

Memory’s Subsymbolic Representation

- At symbolic level
  - chunks in DM
  - retrieval process

```
(p add-ones
  =goal>
     isa        add-pair
     one-ans busy
     one1      =num1
     one2      =num2
  =retrieval>
     isa        addition-fact
     addend1 =num1
     addend2 =num2
     sum       =sum
==>
  =goal>
     one-ans  =sum
     carry      busy
  +retrieval>
     isa         addition-fact
     addend1 10
     sum        =sum
)
```

- When turned on, :esc t,
  - retrieval based on chunk’s “activation”

Memory's Subsymbolic Representation: Activation

- Activation drives both latency and probability of retrieval
- Activation for chunk i:

$$A_i = B_i + \varepsilon_i$$

- Retrieved *if* activation above a threshold (retrieval threshold :rt default 0, often -1)
- Latency calculated from activation

Memory's Subsymbolic Representation: Activation

Activation for chunk i:

$$A_i = B_i + \varepsilon_i$$

B_i = "Base-level activation"

ε_i = noise contribution

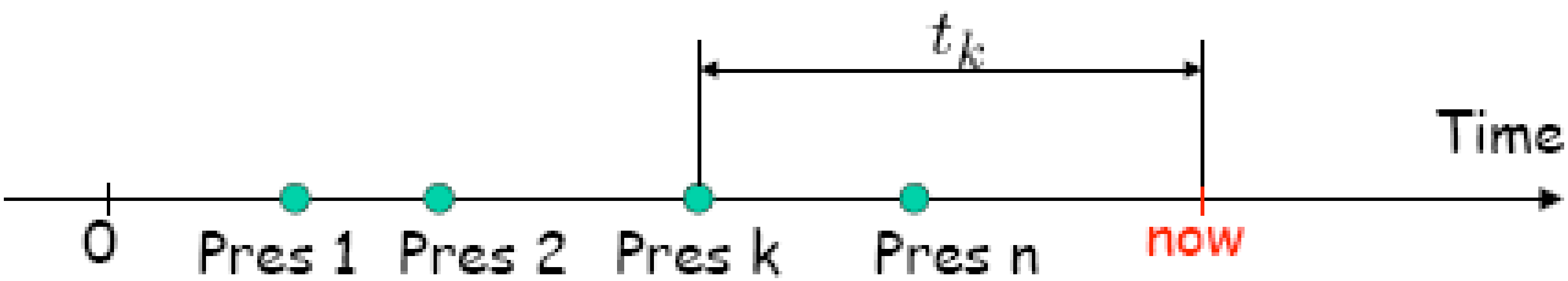
36

Memory's Subsymbolic Representation: Base-level Activation

- Base-level activation
  - Depends on two factors of the history of usage of the chunk: recency & frequency
  - represented as the log of odds of need (Anderson & Schooler, 1991)
  - due to math representation, can be negative
  - includes every previous use
  - affected most by most recent use

37

Memory's Subsymbolic Representation: Base-level Activation



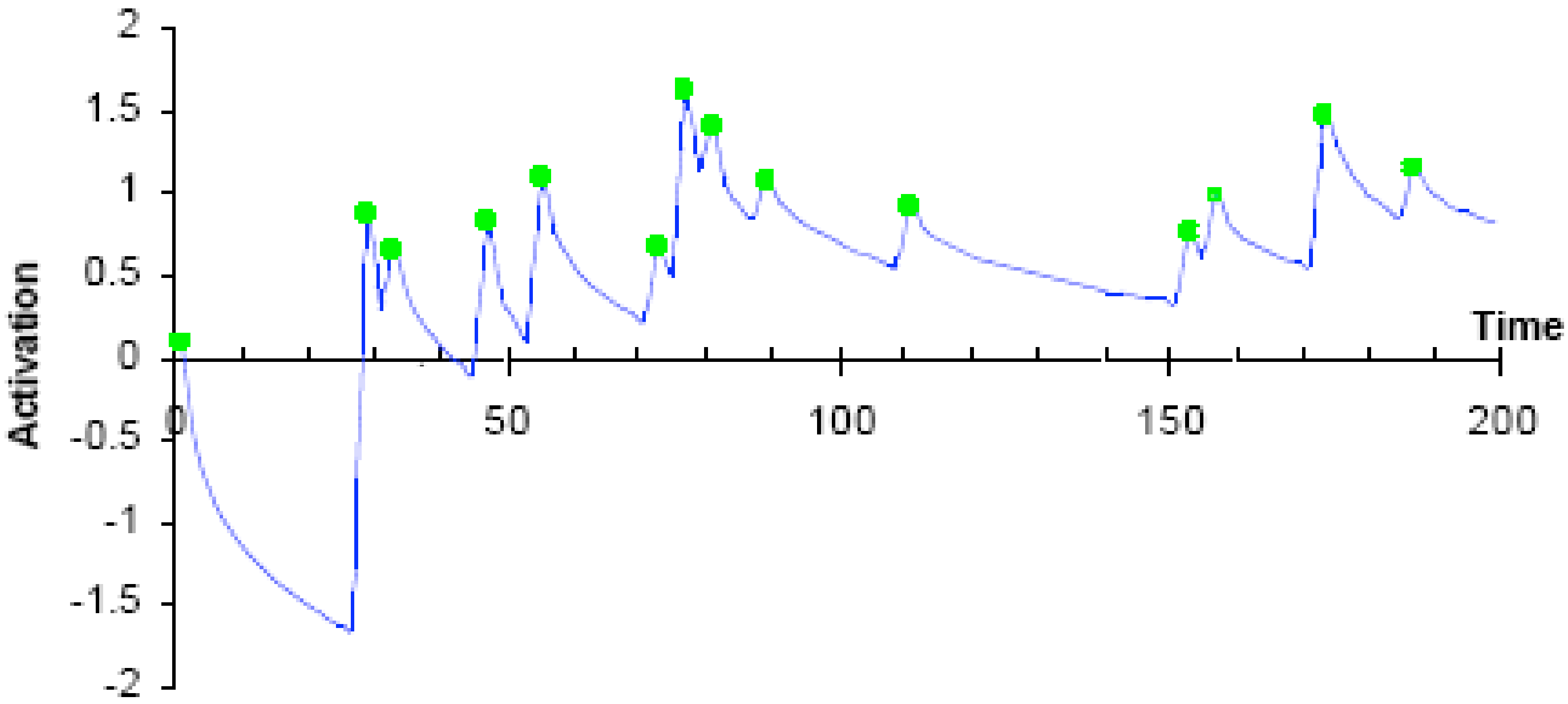
$$B_i = \ln\left(\sum_{k}^{n} t_k^{-d}\right)$$

t_k — time since the k-th presentation of the chunk i

d — decay parameter (sgp :bll 0.5)

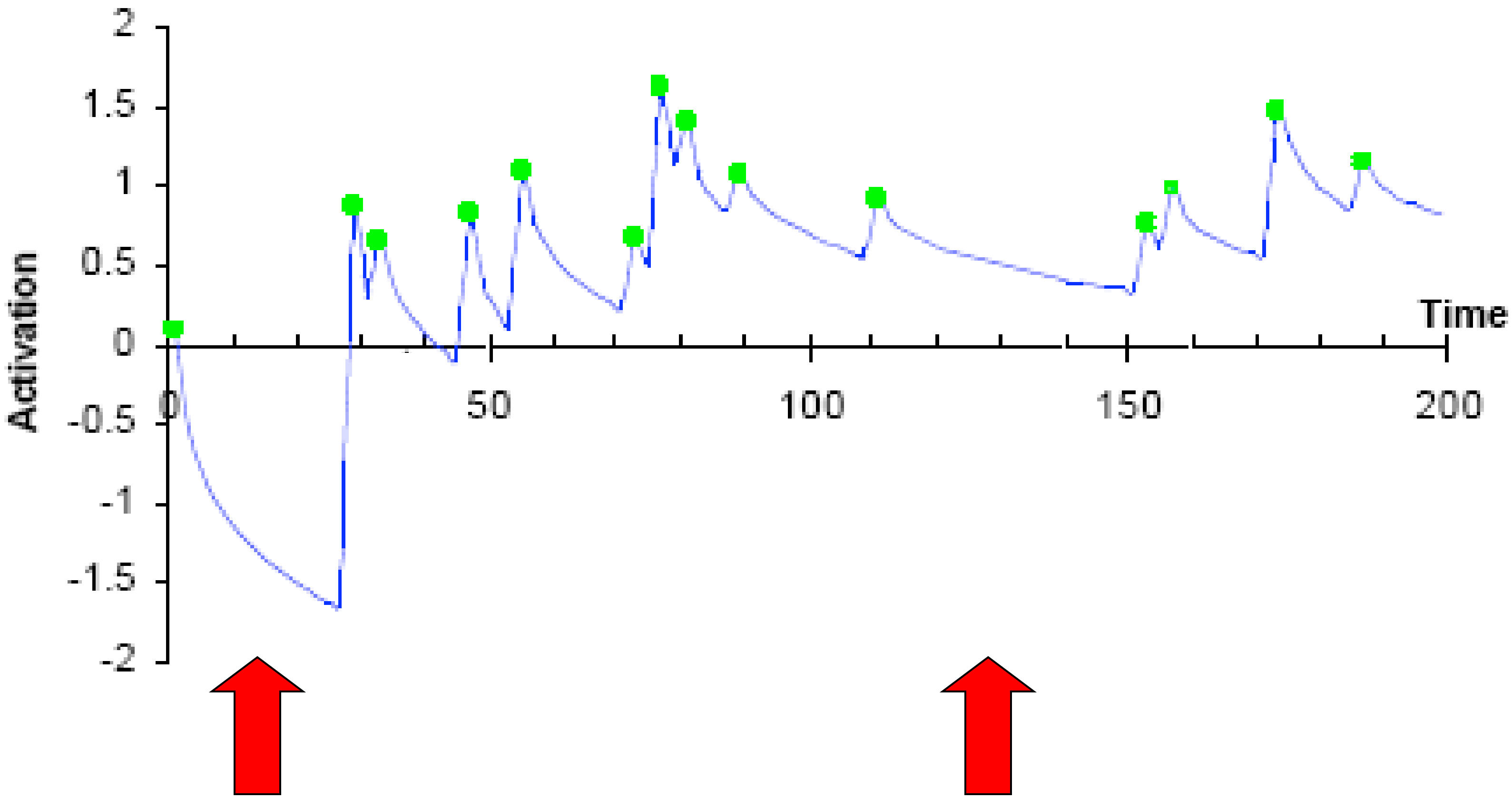
Mathematically transform the ages to compute the current strength of a memory

Memory's Subsymbolic Representation: Base-level Activation



39

Memory’s Subsymbolic Representation: Base-level Activation



With use, decays less

40

Memory's Subsymbolic Representation: Base-level Activation

- Chunk events affecting activation ("event presentations")
  - chunk creation
  - cleared from a buffer and entered into DM
  - cleared from a buffer and already in DM
  - retrieved from DM (credited when cleared)

41

Memory’s Subsymbolic Representation: Base-level Activation

- Base-level activation calculation called “Base-Level Learning”

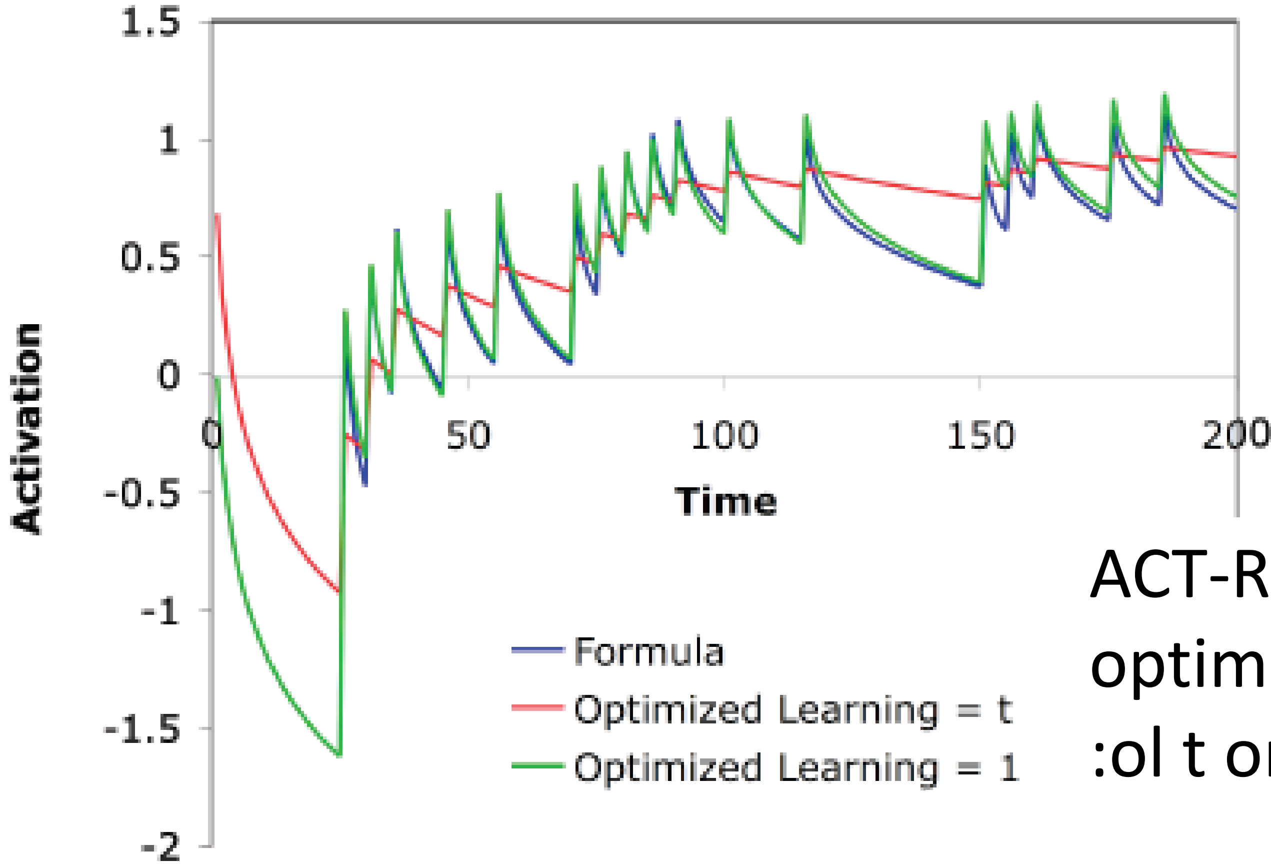
- Key parameter, :bll
  - the exponent in the formula
  - normal value: a half, i.e., 0.5

Memory's Subsymbolic Representation: Base-level Activation

- Full representation requires keeping data on every previous use...
- Alternate: "Optimized" Learning
- If previous chunk events are evenly spaced, a simpler formula is possible
- Keeping even 1 previous event is effective

43

Memory's Subsymbolic Representation: Base-level Activation



ACT-R parameter optimized learning: :ol t or :ol 1

44

Memory's Subsymbolic Representation: Activation

Activation for chunk i:

$$A_i = B_i + \varepsilon_i$$

B_i = "Base-level activation"

ε_i = noise contribution

45

Memory's Subsymbolic Representation: Activation Noise

- ε_i = noise contribution
- 2 parts: permanent & instantaneous
- both ACT-R parameters :pas & :ans
- usually only adjust :ans
- :ans setting varies, from 0.1 to 0.7
- noise in model sometimes nec'y to match noise of human subjects...

46

Memory's Subsymbolic Representation: Latency(s)

- Activation also affects latency (two ways)
- Latency = $F * e^{-A}$

  A is activation

  F is "latency factor" (ACT-R parameter :lf ~0.5)

- Threshold setting affects latency of retrieval failure

  (must wait until latency of threshold passes!)

47

Memory’s Subsymbolic Representation

- Activation = base-level and noise
- Base-level dependent of recency & frequency of previous chunk “presentations”
- Retrieval only when activation above “retrieval threshold”
- Activation <u>and</u> threshold affect latency
- Many parameters :esc, :rt, :bll, :ol, :ans

Memory II:
Other Sources of Activation

- Previously, chunk’s activation over time

- Now, add the effect of context (two types)

Other Sources: Spreading Activation & Partial Matching

- Activation (previous):

  A_i = Base Level Activation + noise

  $= B_i + \varepsilon_i$

- the effect of context (new):

  $A_i = B_i + \varepsilon_i + SA + PM$

Spreading Activation

- Learn multiple similar facts, e.g.,

  A hippie is in the park

  A lawyer is in the cave

  A debutante is in the bank

  A hippie is in the cave

  A lawyer is in the church

Spreading Activation

- Tests (seen before Y/N?)

A lawyer is in the park

A hippie is in the cave

Spreading Activation

- Reponses time increases linearly as number of persons and locations increase, i.e., “fanning out” of activation

- Foils take longer than targets to decide

Spreading Activation

- The <u>context</u> affects retrievals
- Contents of other buffers contribute to retrieval activation calculation for chunks in DM
- Affects response time

Spreading Activation

- Consider: several matching chunks in memory
- How decide which?
- Activation based on base (recency & frequency) PLUS small context effect
- Retrieval based on parts of chunk separates exact matches from non-matches

Spreading Activation

- Activation (previous):

  A_i = Base Level Activation + noise

  $= B_i + \varepsilon_i$

- add <u>context</u>: effect of other buffers' chunks

  $$A_i = B_i + \varepsilon_i + \sum_{\text{buffers}(k)} \sum_{\text{slots}(j)} (W_{kj} S_{ji})$$

Spreading Activation

- add context: effect of other buffers' chunks

$$A_i = B_i + \varepsilon_i + \sum_{\text{buffers}(k)} \sum_{\text{slots}(j)} (W_{kj} S_{ji})$$

W_{kj} is weighting of slot j in buffer k (normalized)

S_{ji} is the strength of the association between slot j and chunk i

Spreading Activation

- add context: effect of other buffers' chunks

$$A_i = B_i + \varepsilon_i + \sum_{\text{buffers}(k)} \sum_{\text{slots}(j)} (W_{kj} S_{ji})$$

W_{kj} is weighting of slot j in buffer k (normalized)

(default is 1 for goal, 0 for others)

S_{ji} is the strength of the association between

slot j and chunk i (S_{ji}=0 or S-ln(fan_j))

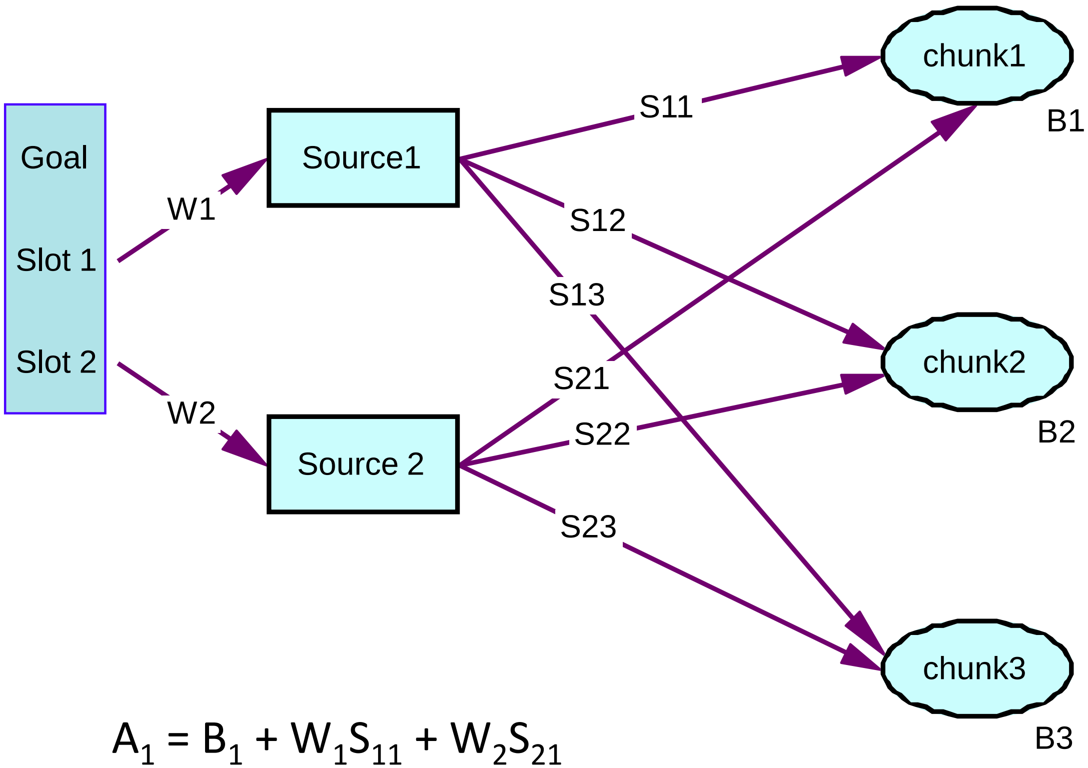
Spreading Activation

Fan Effect (Anderson 1974)

- Fan effect: number of associations “fanning out” from a chunk

- Other buffers hold chunks
- Chunk has slots with other chunks
- How many uses of a chunk affects its A_i

Spreading Activation: Fan Effect



$A_1 = B_1 + W_1 S_{11} + W_2 S_{21}$
$A_2 = B_2 + W_1 S_{12} + W_2 S_{22}$
$A_3 = B_3 + W_1 S_{13} + W_2 S_{23}$

Spreading Activation: Fan Effect

- Retrievals based on matching & activation
- Now, other buffers affect retrieval
- But, activation diluted by similar chunks

- Effect:

  Similar but non-matches slow retrievals

Other Sources: Partial Matching

Other Sources: Partial Matching

- Provides ACT-R a mechanism to explain errors of commission, retrieving wrong chunk
- (previous activation mechanism explained errors of omission, $A_i < :RT$)

Partial Matching

- add context: effect of <u>similar chunks</u>

$$A_i = B_i + \boldsymbol{\varepsilon}_i + \sum_{\text{buffers(k)}} \sum_{\text{slots(j)}} (W_{kj}\, S_{ji)}) + \sum_{\text{retrieval slots}} PM_{li}$$

P is weighting of slot

M_{li} is the similarity between values in slot $_l$ of retrieval and slot $_i$ of chunk

Partial Matching

- add context: effect of similar chunks

$$A_i = B_i + \varepsilon_i + \sum_{\text{buffers(k)}} \sum_{\text{slots(j)}} (W_{kj} S_{ji)}) + \sum_{\text{retrieval slots}} PM_{li}$$

P is weighting of slots (all equal)

M_{li} is the similarity between values in slot $_l$ of retrieval and slot $_i$ of chunk

Partial Matching

- Effect is can retrieve a wrong but similar chunk (... IF chunk hierarchy supports it)

- Retrieval of wrong chunk supports errors of commission, taking wrong action vice no action

ACT-R Modeling

- ACT-R Model Development
- ACT-R Model Debugging

67

ACT-R Model Development

1. Plan overall model to work in stages.

2. Start simple then add details to your model.

3. Write simple productions using simple chunks.

4. Run the model (with own trace) frequently to test progress (eg. with every new or changed production).

68

ACT-R Model Development

5. Start with productions doing one thing at a time (i.e., reference goal + one buffer) and use multiple productions. Combine later.

6. Use state variables to rigorously control sequencing until model works, then remove as many as possible.

69

ACT-R Model Development

7. With each buffer request, consider a production for handling the failure.

8. In using loops, consider preps to start and how to leave loop.

70

ACT-R Debugging Process

- Run ACT-R up to problem...
  - set time limit
  - change production to stop at problem step
- Check "why not" of expected production
- Check buffers & buffer status
- Check visicon
- Causes ...

71

ACT-R Code Debugging

Stops unexpectedly/expected production not firing:

- Conditions not met (use "Why not" to identify which)
- Conditions over-specified with unnec'y variable tests which don’t match
- Logic mismatch among conditions
- nil will not match =variable
- …

72

ACT-R Code Debugging

Stops unexpectedly/expected production not firing (continued):

- Typo on variable name, i.e., not same ref.
- Wrong slot referenced in LHS
- Time ran out
- Production not in memory
- Error on loading (production ignored)
- Production overwritten by duplicate naming (warning)

73

ACT-R Code Debugging

Wrong production firing:

- Firing production also meets current conditions
- Conditions do not meet expected production LHS

Production firing repeatedly:

- LHS not changed by firing, i.e., still valid

74

ACT-R Code Debugging

Buffer unexpectedly empty:

- Not filled
- Implicit clearing (on LHS but not RHS)

Buffer with unexpected chunk:

- Previous production to fill it didn't fire
- Sensor handling not as expected
- Buffer not updated/cleared as expected

75

ACT-R Code Debugging

Retrieval unsuccessful:

- Expected chunk not in memory
- Retrieval specification unintended
  - overly specific (too many slots specified)
  - unintended chunk type
- Expected chunk's activation too low
- Wrong chunk retrieved
  - under specified (too few slots specified)
  - partial matching effect (intended)

ACT-R Code Debugging

Timing too slow:

- Combine productions
- Driven by retrieval failures and :RT too low

Timing too fast:

- Driven by retrieval failures and :RT too high

77

Unit 4: Zbrodoff's Experiment

- alpha arithmetic, eg: A + 2 = C: correct?
- possible addends: 2, 3, or 4
- learning over:
  - stimuli set (24)
  - repetition (5)
  - blocks (2)   = 192 trials

Model Design

- Given model that counts to answer
- Process: read problem, count, answer
- Already creates saves chunks of answers
- Strategy?

Model Design - start

- Basic strategy: learn to retrieve answer
- How: attempt retrieval

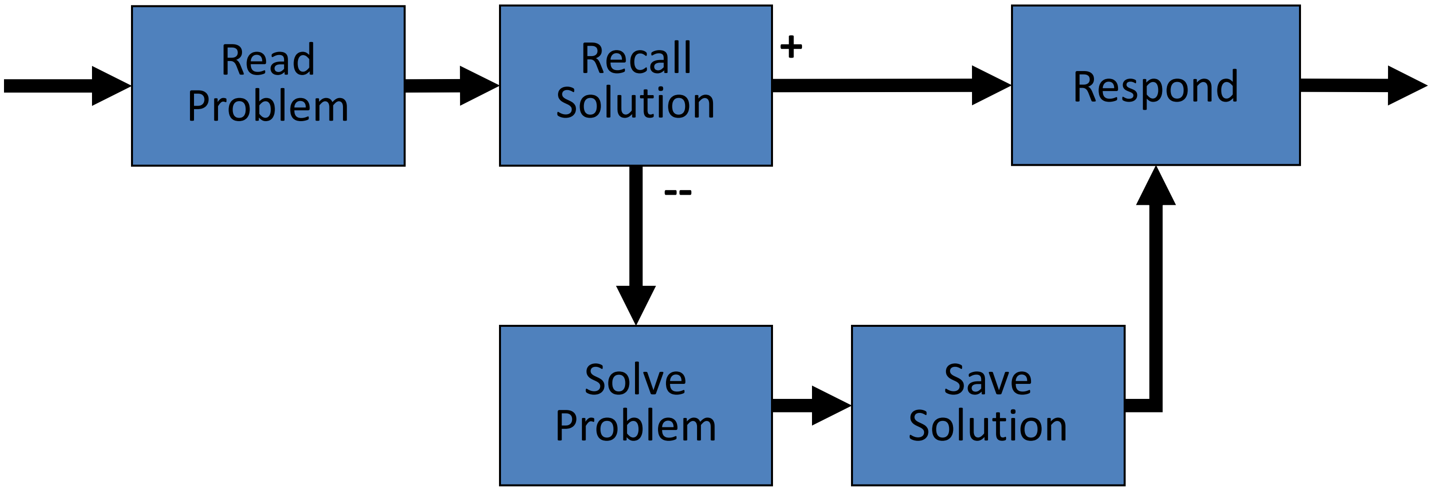
  if successful, answer based on it

  if fails, resort to counting

  (another basic process...)
- After model runs, adjust :RT

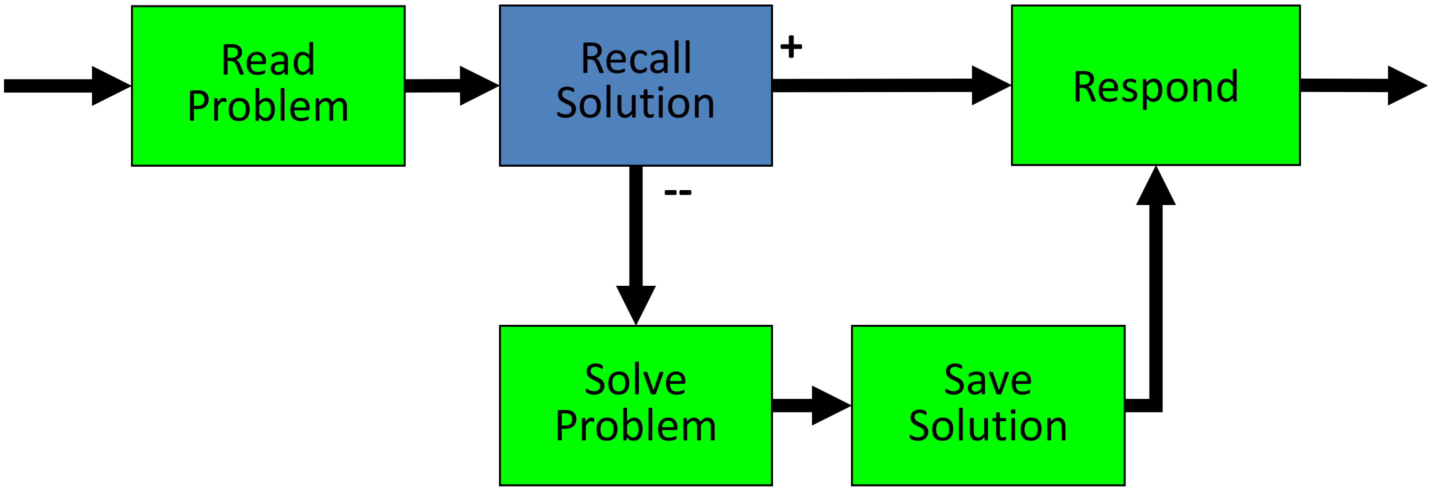
Model Design – basic process

Instance-based action/declarative learning process:



Model Coding

Ziegler homework: ~80% given



Parameter Tweaking

- After model runs, adjust retrieval threshold to control when learning occurs vs. problem solving

Unit 5: Siegler Experiment

- Experiment is 4 yr olds answering addition problems: 1+1 up to 3+3 (answers from 0-8 & “other”)
- Environment starts with problem stuffed into imaginal buffer as “text”
- Goal: beat 0.976 / 0.058
- Strategy?

Model Development Process

1. get working for one case
2. expand for all cases
3. adjust parameters

Model / Coding Issue

- Note: one <> “one” <> “1” <> 1,

  (symbol, strings, and a constant)

- Facts stored using different representations

```
Number fact:   (one ISA number value 1 name "one")
Addition fact: (f11 ISA plus-fact addend1 one
     addend2 one sum two)
```

- problem input & response as strings, eg., “one”

Model / Coding Issue

- Now possible to access to chunk names

  eg., =retrieval

  like previous =visual-location

- Or, ("old school") add slot with value, eg:

  (four ISA number value 4 name "four" ref four)

Model Design - basics

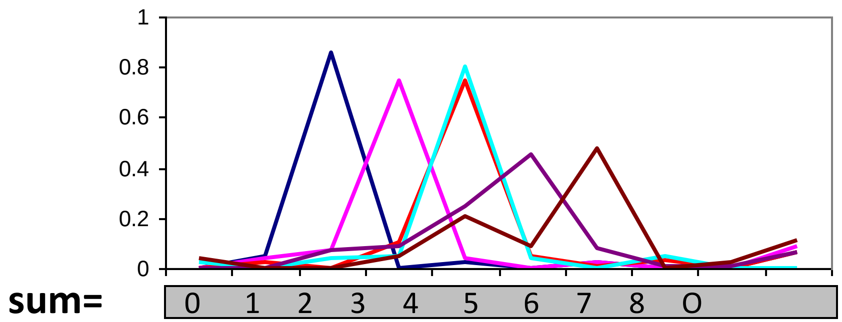
- Given chunk types and functions to set base activation and set similarities

- Basic process: encode, retrieve, respond

Parameter Tweaking

- Set base level activations for successful retrievals throughout a run
- Adjust :ans to get “other” responses
- Set similarities to generate error distribution to match human subjects

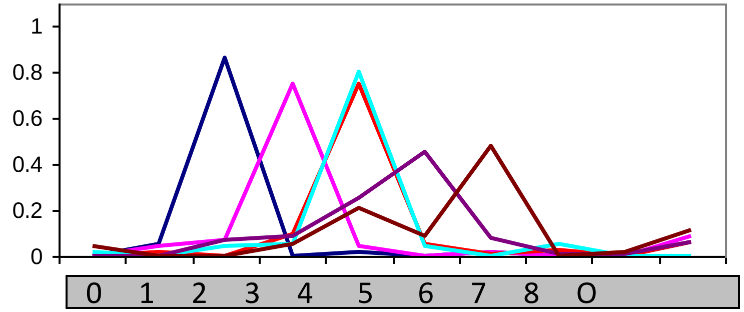
Siegler Experiment Data



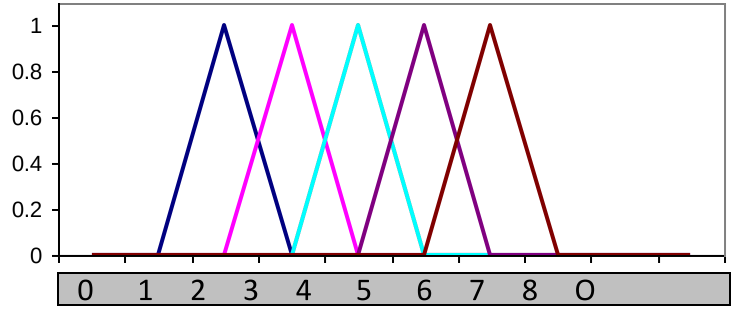
	zero	one	two	three	four	five	six	seven	eight	other
1+1	0	0.05	0.86	0	0.02	0	0.02	0	0	0.06
1+2	0	0.04	0.07	0.75	0.04	0	0.02	0	0	0.09
1+3	0	0.02	0	0.1	0.75	0.05	0.01	0.03	0	0.06
2+2	0.02	0	0.04	0.05	0.8	0.04	0	0.05	0	0
2+3	0	0	0.07	0.09	0.25	0.45	0.08	0.01	0.01	0.06
3+3	0.04	0	0	0.05	0.21	0.09	0.48	0	0.02	0.11

Siegler Data & Models

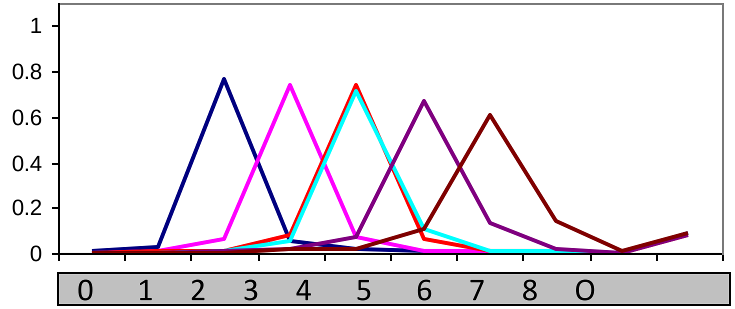
Experimental Data



Perfect Behavior
(0.942 / 0.125)

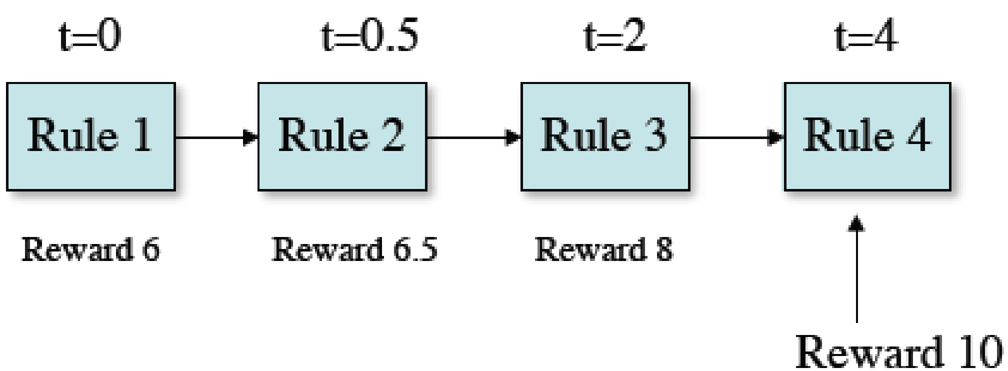


My Model
(0.960 / 0.059)

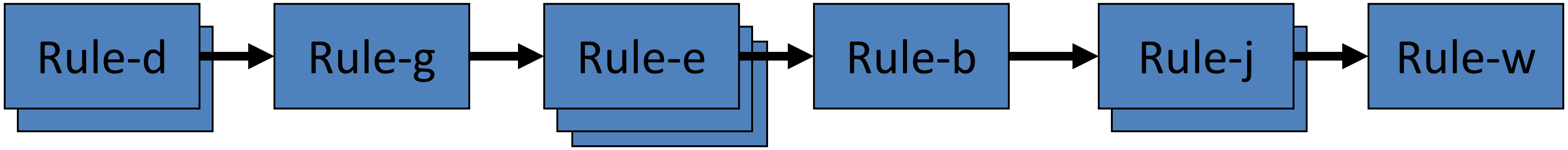


Procedural Knowledge "Reward"

- A reward value is propagated backwards through rule firings and depreciated by time



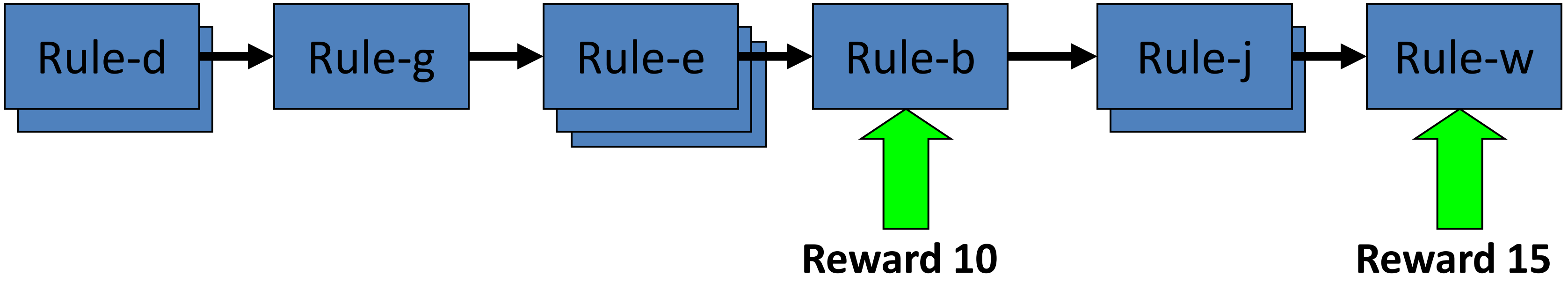
Procedural Knowledge “Reward”



Procedural Knowledge “Reward”

- Eg: (spp Rule-b :reward 10)

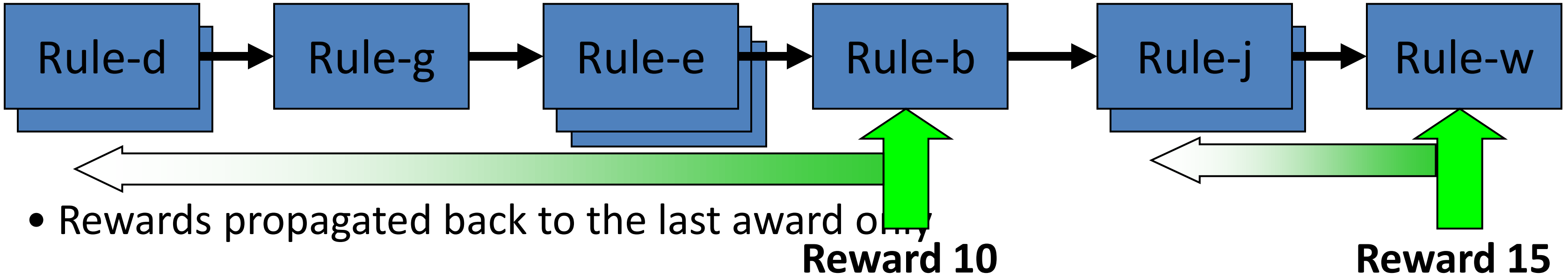
  (spp Rule-w :reward 15)



Procedural Knowledge "Reward"

- Eg: (spp Rule-b :reward 10)

  (spp Rule-w :reward 15)



- Rewards propagated back to the last award only

Why?

- Multiple rules' LHS meet current conditions and appropriate strategy is a balance between the them.

- Rule selection balance can be learned

- To turn on utility learning: **`(sgp :ul t)`**

Learning Utility

- Difference Learning Equation:

$$U_i(n) = U_i(n-1) + \alpha[\, R_i(n) - U_i(n-1)\,]$$

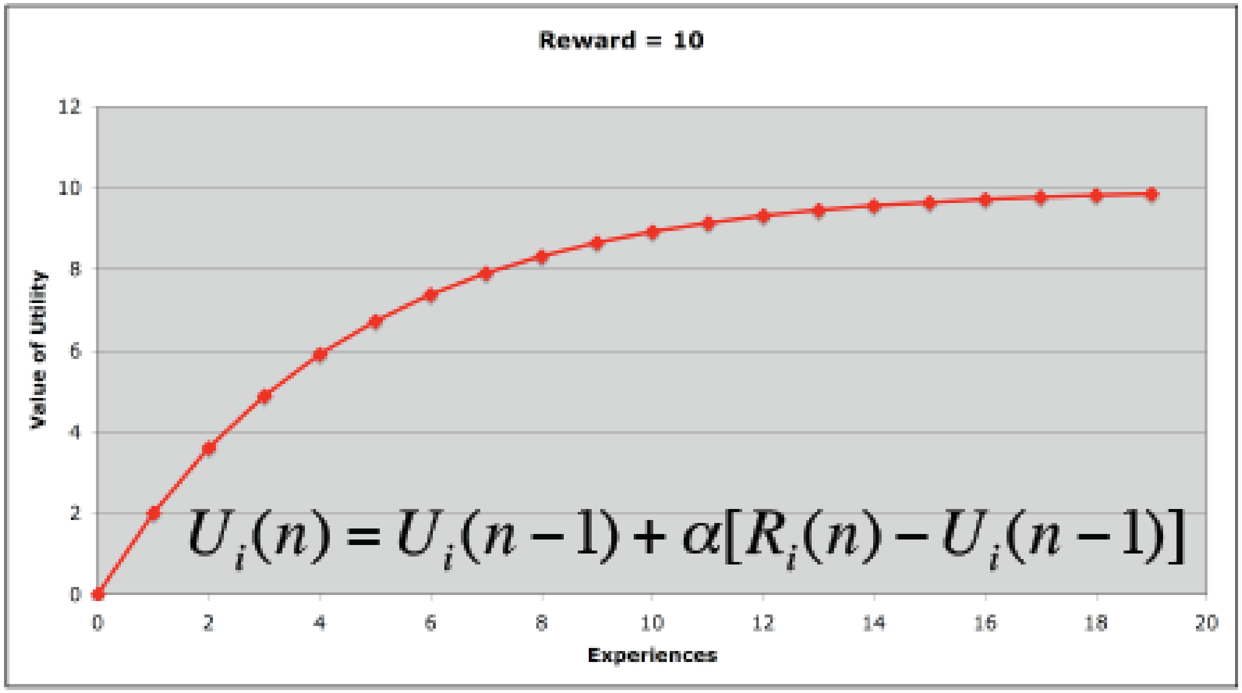
$U_i(n)$ is utility of rule $_i$ at n^{th} firing

$R_i(n)$ is reward for rule $_i$ at n^{th} firing

α is "learning rate" **`(sgp :alpha 0.2)`**

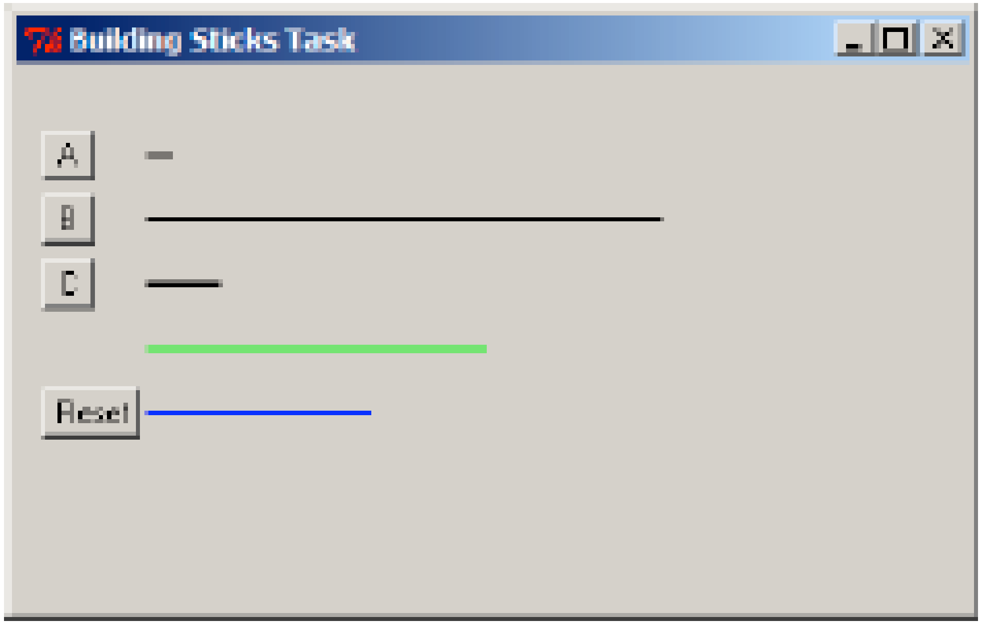
Learning Utility

- Difference Learning Equation effect:

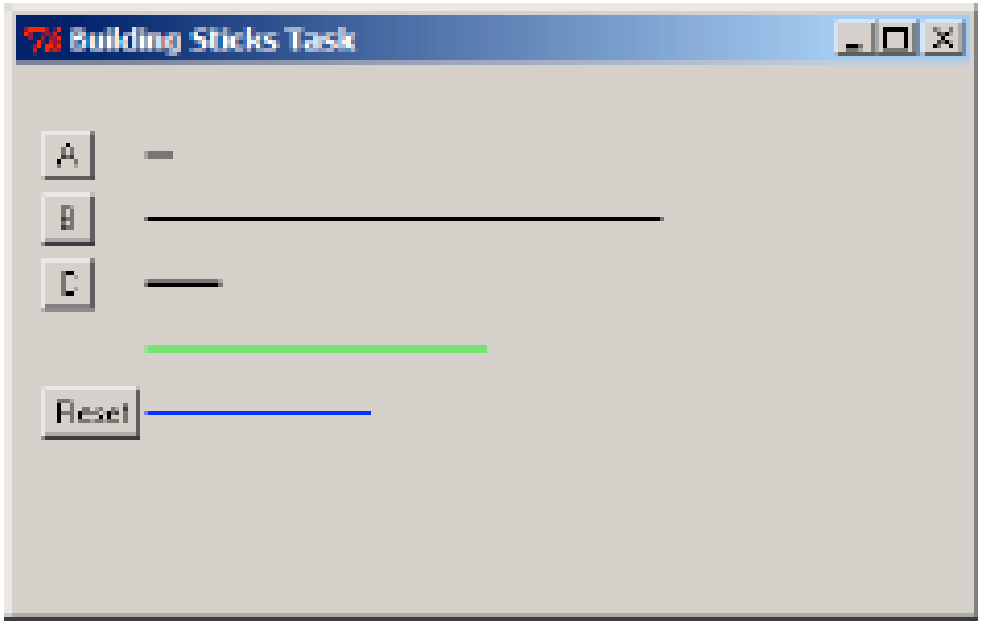


Example: Building Sticks Task

- Match a given length by a combination of 3 other given lengths
- Two strategies: building up or subtracting (“undershoot”/“overshoot” based on first move)



Example: Building Sticks Task



- Searching for combination of A,B, & C that equals target length (green)
- If current is too long, length is subtracted

Example: Building Sticks Task

Demo

A=15, B=200, C=41

Goal: 103

Example: Building Sticks Task

Demo

A=15, B=200, C=41

Goal: 103 = B – 2C – A

Example: Building Sticks Task

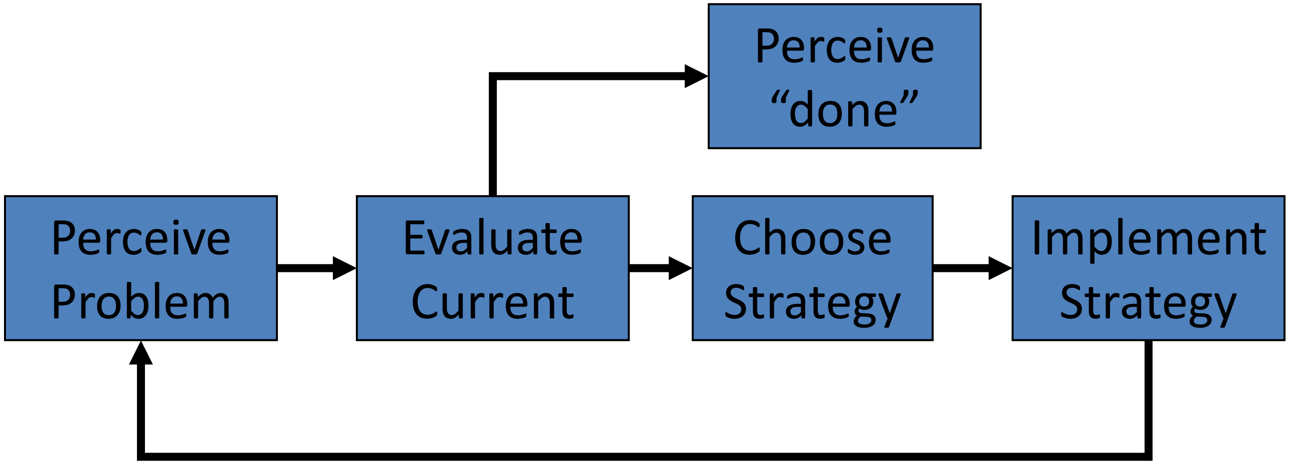
Human performance data:

available lengths a	b	c	Goal	%Overshoot
15	250	55	125	20
10	155	22	101	67
14	200	37	112	20
22	200	32	114	47
10	243	37	159	87

...

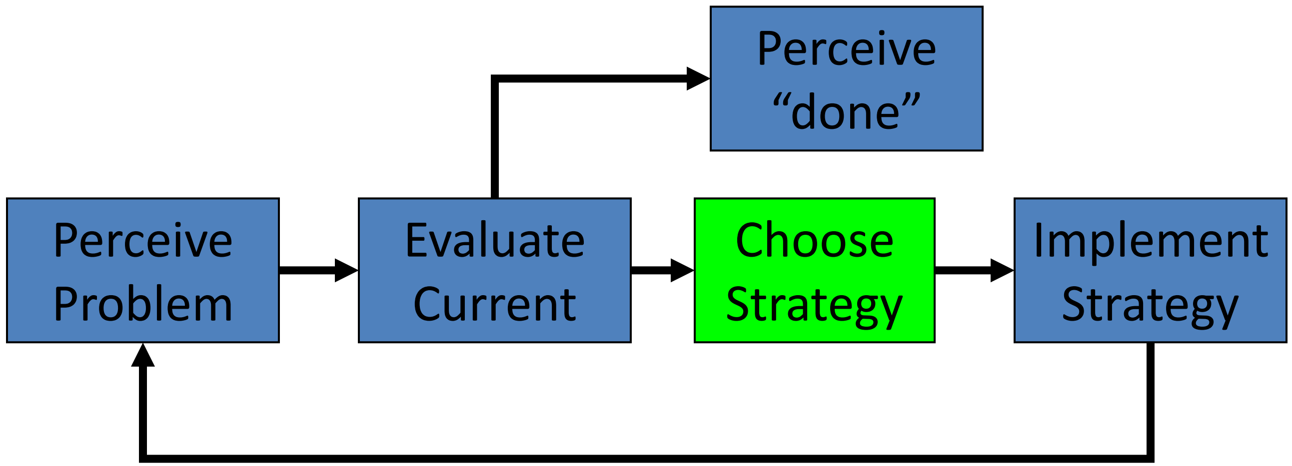
Example: Building Sticks Task

- BST model has 27 productions



Example: Building Sticks Task

- BST model has 27 productions



Example: Building Sticks Task

- BST model has 27 productions
- Key productions decide which length to use based on whether current length is too long ("over") or too short ("under")

```
decide-over
decide-under
force-over
force-under
```

Example: Building Sticks Task

```
(p decide-under
   =goal>
     isa      try-strategy
     state    choose-strategy
     strategy nil
   =imaginal>
     isa      encoding
     over     =over
     under    =under

   !eval! (< =under (- =over 25))
==>
   =imaginal>
   =goal>
     state    prepare-mouse
     strategy under
   +visual-location>
     isa      visual-location
     kind     oval
     screen-y 85)
```

```
(p force-under
   =goal>
     isa      try-strategy
     state    choose-strategy
   - strategy under
==>
   =goal>
     state    prepare-mouse
     strategy under
   +visual-location>
     isa      visual-location
     kind     oval
     screen-y 85)
```

Example: Building Sticks Task

- (collect-data 100)→ corr: 0.803
- Utilities start → end

decide-over	13 → 13.15
decide-under	13 → 11.15
force-over	10 → 12.15
force-under	10 → 6.59

Unit 6 Homework

Model design:

productions

- start by detecting “choose” in window
- generate key-press (heads <u>and</u> tails)
- read actual result
- note matches and non-matches

settings :ul t (to turn on utility learning)

:egs noise parameter

set initial utilities

AND establish rewards for specific productions

Unit 6 Homework

Model results (collect-data 100):

Corr: / Mean Dev.

official answer: 0.991 / 0.012

Unit 6 Homework

Model results (collect-data 100):

	Corr: / Mean Dev.
official answer:	0.991 / 0.012
my best:	0.998 / 0.010

Unit 6 Homework

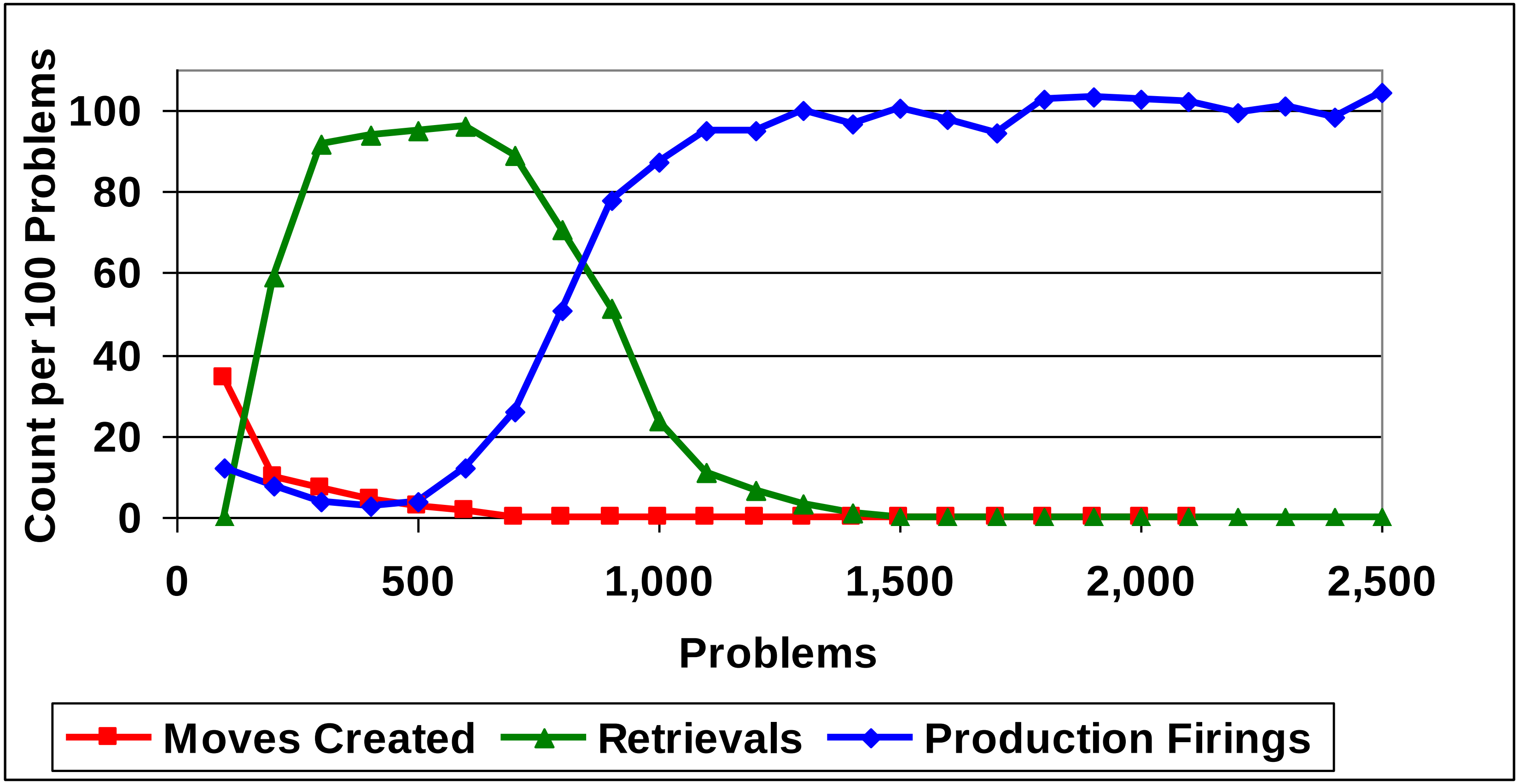
Model results (collect-data 100):

	Corr: / Mean Dev.
official answer:	0.991 / 0.012
my best:	0.998 / 0.010
another run:	0.990 / 0.030
run 200:	0.990 / 0.024

Learning New Rules

- Anderson (1982) suggested a 3 stage learning process:
  - “Cognitive” (problem solving, chunking)
  - “Associative” (retrieval of solutions)
  - “Autonomous” (new procedural knowledge)

Learning New Rules



from Kennedy & Trafton (2007) Long-term Learning, CSR

Learning New Rules

- How? compiling rules that fire sequentially

Abstract:

Rule A→B followed by rule B→C

...

Learning New Rules

- How? compiling rules that fire sequentially

Abstract:

Rule A→B followed by rule B→C combined into a new rule: A→B&C

Learning New Rules

```
(p rule1
   =goal>
      isa      goal
      state    nil
==>
   =goal>
      state    start
)

(p rule2
   =goal>
      isa      goal
      state    start
==>
   =goal>
      add1  zero
)
```

Learning New Rules

```
(p rule1
   =goal>
      isa      goal
      state    nil
==>
   =goal>
      state   start
)

(p rule2
   =goal>
      isa      goal
      state    start
==>
   =goal>
      add1  zero
)
```

A

→

B

B

→

C

Learning New Rules

```
(p rule1
   =goal>
      isa      goal
      state    nil
==>
   =goal>
      state   start
)

(p rule2
   =goal>
      isa      goal
      state    start
==>
   =goal>
      add1  zero
)
```

```
(p rule1+2
   =goal>
      state   nil
==>
   =goal>
      state  start
      add1 zero
)
```

A
→
B
C

Learning New Rules

```
(p rule1
   =goal>
      isa      goal
      state    nil
==>
   =goal>
      state   start
)

(p rule2
   =goal>
      isa      goal
      state    start
==>
   =goal>
      add1  zero
)
```

```
(p rule1+2
     =goal>
      state    nil
==>
     =goal>
       state  start
       add1 zero
     )
```

Learning New Rules

- Is that all?

Realistic example:

- Rule 1 initiates retrieval of a chunk
- Rule 2 harvests the retrieved chunk
- New Rule 1+2 built with chunk information "built in" i.e., no retrieval involved

Learning New Rules

Realistic example (specifics):

- Recall the two column addition exercise
- Add by counting from first number, the second number of times
- Consider a specific case: start with 3 and need to count 2 steps to get sum

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     =arg1
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA    order
    first      =arg1
   =goal>
    sum    =arg1
    count      zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         =sum
    count       =count
   =retrieval>
    ISA    order
    first      =sum
    second     =new
==>
   =goal>
    sum    =new
   +retrieval>
    ISA    order
    first      =count
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     =arg1
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       =arg1
   =goal>
    sum     =arg1
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         =sum
    count       =count
   =retrieval>
    ISA     order
    first       =sum
    second      =new
==>
   =goal>
    sum     =new
   +retrieval>
    ISA     order
    first       =count
```

<u>Retrieved chunk</u>:

ISA order
first three
second four

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     =arg1
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA    order
    first       =arg1
   =goal>
    sum    =arg1
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         =sum
    count       =count
   =retrieval>
    ISA    order
    first       =sum
    second      =new
==>
   =goal>
    sum    =new
   +retrieval>
    ISA    order
    first       =count
```

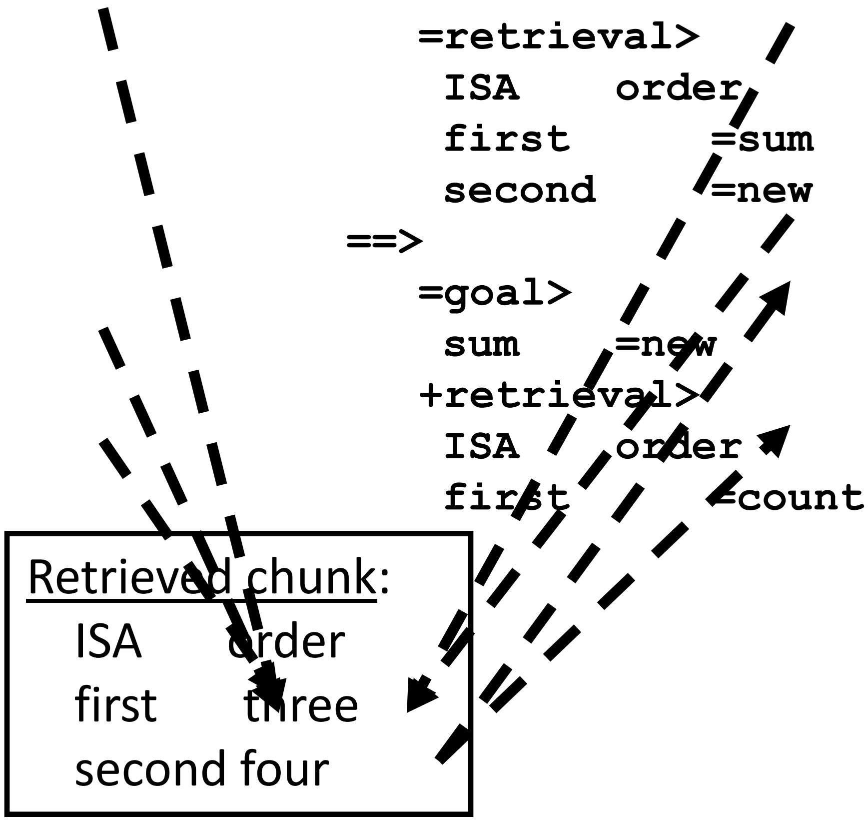
<u>Retrieved chunk</u>:

ISA order
first three
second four

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     =arg1
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA    order
    first      =arg1
   =goal>
    sum    =arg1
    count      zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         =sum
    count       =count
   =retrieval>
    ISA    order
    first       =sum
    second      =new
==>
   =goal>
    sum    =new
   +retrieval>
    ISA    order
    first       =count
```



Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA          add
    addend1      three
    addend2      =arg2
    sum          nil
==>
   +retrieval>
    ISA     order
    first        three
   =goal>
    sum     three
    count        zero
```

```
(P increment-sum
   =goal>
    ISA          add
    sum          three
    count        =count
   =retrieval>
    ISA     order
    first        three
    second       four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first        =count
```

Retrieved chunk:

ISA order

first three

second four

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         three
    count       =count
   =retrieval>
    ISA     order
    first       three
    second      four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       =count
```

Retrieved chunk:

ISA order
first three
second four

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         three
    count       =count
   =retrieval>
    ISA     order
    first       three
    second      four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       =count
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA          add
    addend1      three
    addend2      =arg2
    sum          nil
==>
   +retrieval>
    ISA     order
    first        three
   =goal>
    sum     three
    count        zero
```

```
(P increment-sum
   =goal>
    ISA          add
    sum          three
    count        =count
   =retrieval>
    ISA     order
    first        three
    second       four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first        =count
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA          add
    addend1      three
    addend2      =arg2
    sum          nil
==>
   +retrieval>
    ISA      order
    first        three
   =goal>
    sum      three
    count        zero
```

```
(P increment-sum
   =goal>
    ISA          add
    sum          three
    count        =count
   =retrieval>
    ISA      order
    first        three
    second       four
==>
   =goal>
    sum      four
   +retrieval>
    ISA      order
    first        =count
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         three
    count       =count
   =retrieval>
    ISA     order
    first       three
    second      four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       =count
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         three
    count       =count
   =retrieval>
    ISA     order
    first       three
    second      four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       =count
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         three
    count       =count
   =retrieval>
    ISA     order
    first       three
    second      four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       =count
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA        add
    addend1    three
    addend2    =arg2
    sum        nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA        add
    sum        three
    count      zero
   =retrieval>
    ISA     order
    first       three
    second      four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       zero
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         three
    count       zero
   =retrieval>
    ISA     order
    first       three
    second      four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       zero
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA          add
    sum          three
    count        zero
   =retrieval>
    ISA      order
    first        three
    second       four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       zero
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA          add
    addend1      three
    addend2      =arg2
    sum          nil
==>
   +retrieval>
    ISA      order
    first        three
   =goal>
    sum      three
    count        zero
```

```
(P increment-sum
   =goal>
    ISA          add
    sum          three
    count        zero
   =retrieval>
    ISA      order
    first        three
    second       four
==>
   =goal>
    sum      four
   +retrieval>
    ISA      order
    first        zero
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA          add
    addend1      three
    addend2      =arg2
    sum          nil
==>
   +retrieval>
    ISA     order
    first        three
   =goal>
    sum     three
    count        zero
```

```
(P increment-sum
   =goal>
    ISA          add
    sum          three
    count        zero
   =retrieval>
    ISA     order
    first        three
    second       four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first        zero
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first       three
   =goal>
    sum     three
    count       zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         three
    count       zero
   =retrieval>
    ISA     order
    first       three
    second      four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first       zero
```

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA         add
    addend1     three
    addend2     =arg2
    sum         nil
==>
   +retrieval>
    ISA     order
    first        three
   =goal>
    sum     three
    count        zero
```

```
(P increment-sum
   =goal>
    ISA         add
    sum         three
    count       zero
   =retrieval>
    ISA     order
    first        three
    second       four
==>
   =goal>
    sum     four
   +retrieval>
    ISA     order
    first        zero
```

Learning New Rules

LHS

```
(P initialize-add-count
  =goal>
   ISA        add
   addend1    three
   addend2    =arg2
   sum        nil
==>
  +retrieval>
   ISA     order
   first       three
  =goal>
   sum     three
   count       zero
```

```
(P increment-sum
  =goal>
   ISA         add
   sum         three
   count       zero
  =retrieval>
   ISA     order
   first       three
   second      four
==>
  =goal>
   sum     four
  +retrieval>
   ISA     order
   first       zero
```

RHS

Learning New Rules

```
(P initialize-add-count
   =goal>
    ISA          add
    addend1      three
    addend2      =arg2
    sum          nil
==>
   +retrieval>
    ISA      order
    first        six
   =goal>
    sum      six
    count        zero
```

```
(P initalize+increment
   =goal>
    ISA          add
    addend1      three
    addend2      =arg2
    sum          nil
==>
   =goal>
    count        zero
    sum          four
   +retrieval>
    ISA          order
    first        zero
```

```
(P increment-sum
   =goal>
    ISA          add
    sum          three
    count        zero
   =retrieval>
    ISA          order
    first        three
    second       four
==>
   =goal>
    sum          four
   +retrieval>
    ISA          order
    first        zero
```

Learning New Rules

NOTICE:

1. New rule is very specific due to adding the retrieved chunk’s information to the two parent rules

Learning New Rules

NOTICE:

2. Very simply written rules can become more sophisticated with production compilation (i.e., *write simple rules that get the desired behavior and let the system learn to be fast enough to match data*...)

Learning New Rules

- Utility of new rule?
- Intent: gradual rule use
  (contrary to Soar philosophy)

Learning New Rules

- Previous utility rule

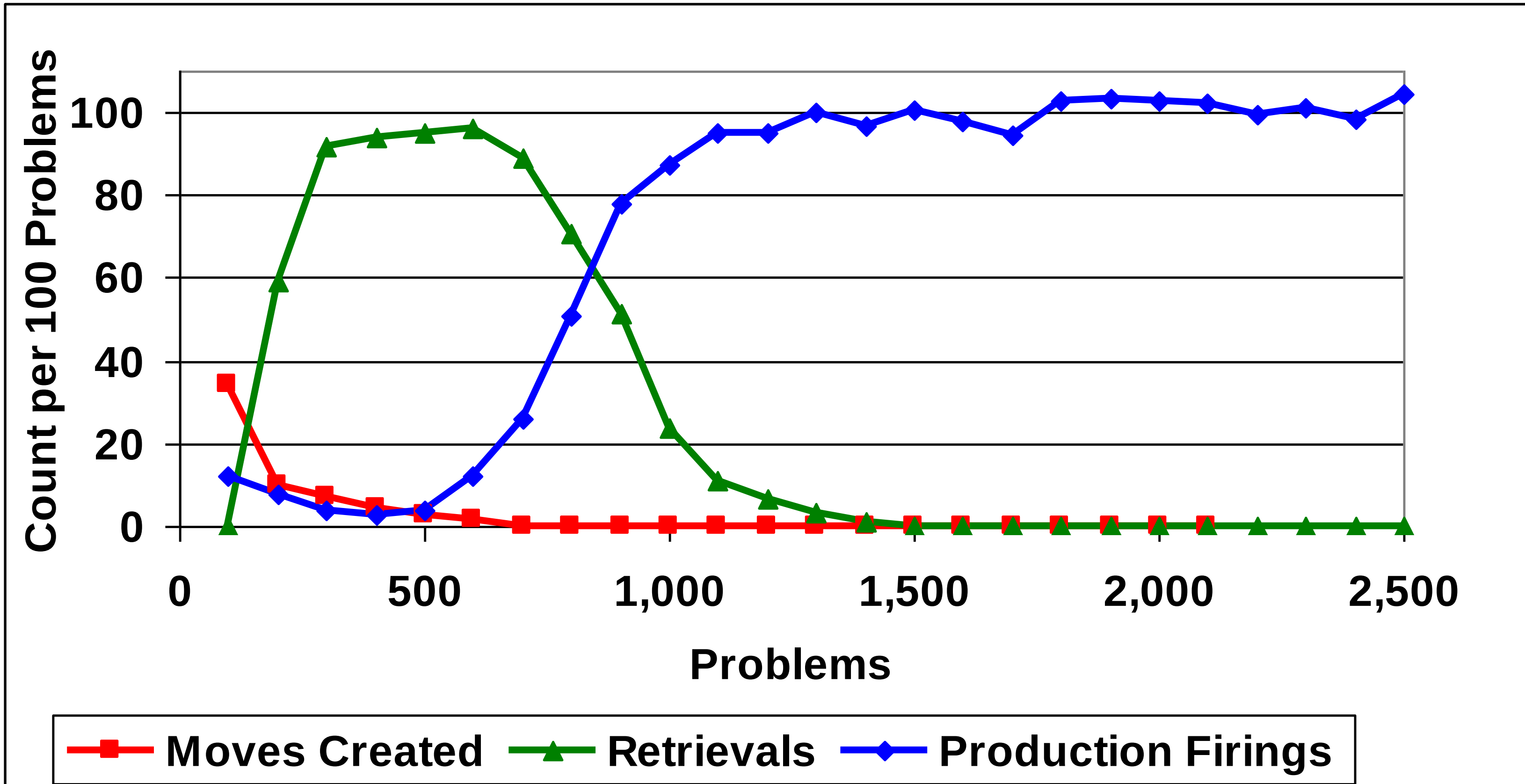
  $U_i(n) = U_i(n\text{-}1) + a\,[\,(R_i(n) - U_i(n\text{-}1)\,]$

- Utility for new rule

  $U_i(n) = U_i(n\text{-}1) + a\,[\,(U_{1st\ parent}(n) - U_i(n\text{-}1)\,]$

- Starts at 0 (default) and is increased with re-creation until with noise gets selected. Then gets reward more directly and can surpass parent.

Learning New Rules



from Kennedy & Trafton (2007) Long-term Learning, CSR

Learning New Rules

- Rule compilation conditions:
  - no conflicts
  - no “externalities”, i.e., reliance on outside world
  - must be turned on: **`(sgp :epl t)`**

    (default is nil)

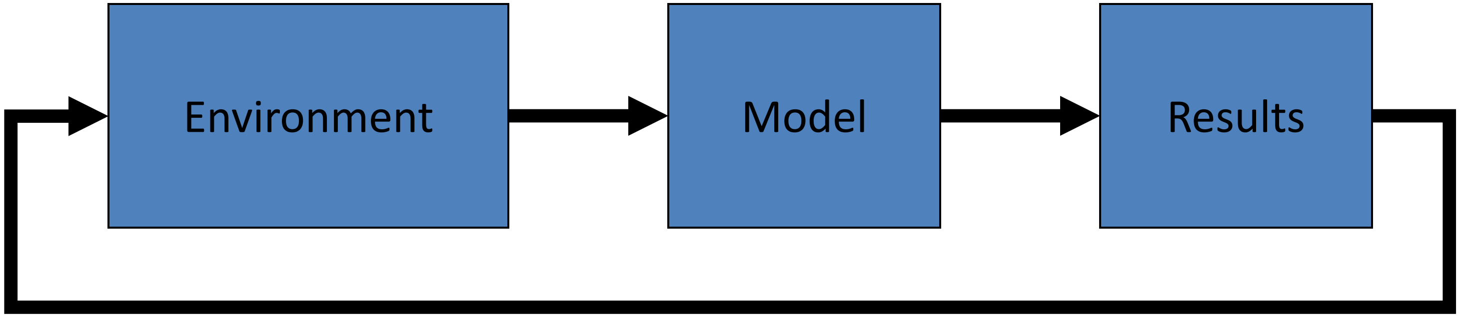
Review

- Learning new procedural knowledge:
  - 2 sequentially-firing rules combine into a new rule
  - production learning turned on with **`(sgp :epl t)`**
  - utility set like reward

- Use: learned new rules eventually take over and can be much faster than parents

Unit 7 Homework

- Model design?



Unit 7 Homework

- Given:

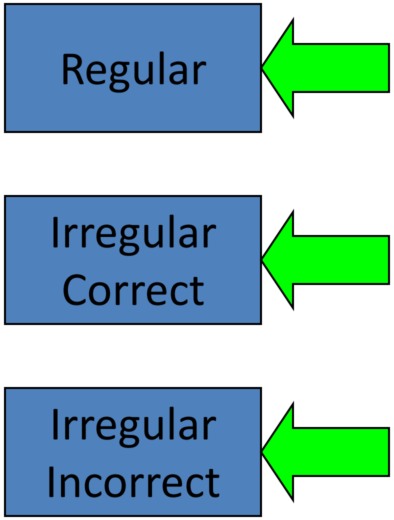
Regular

Irregular Correct

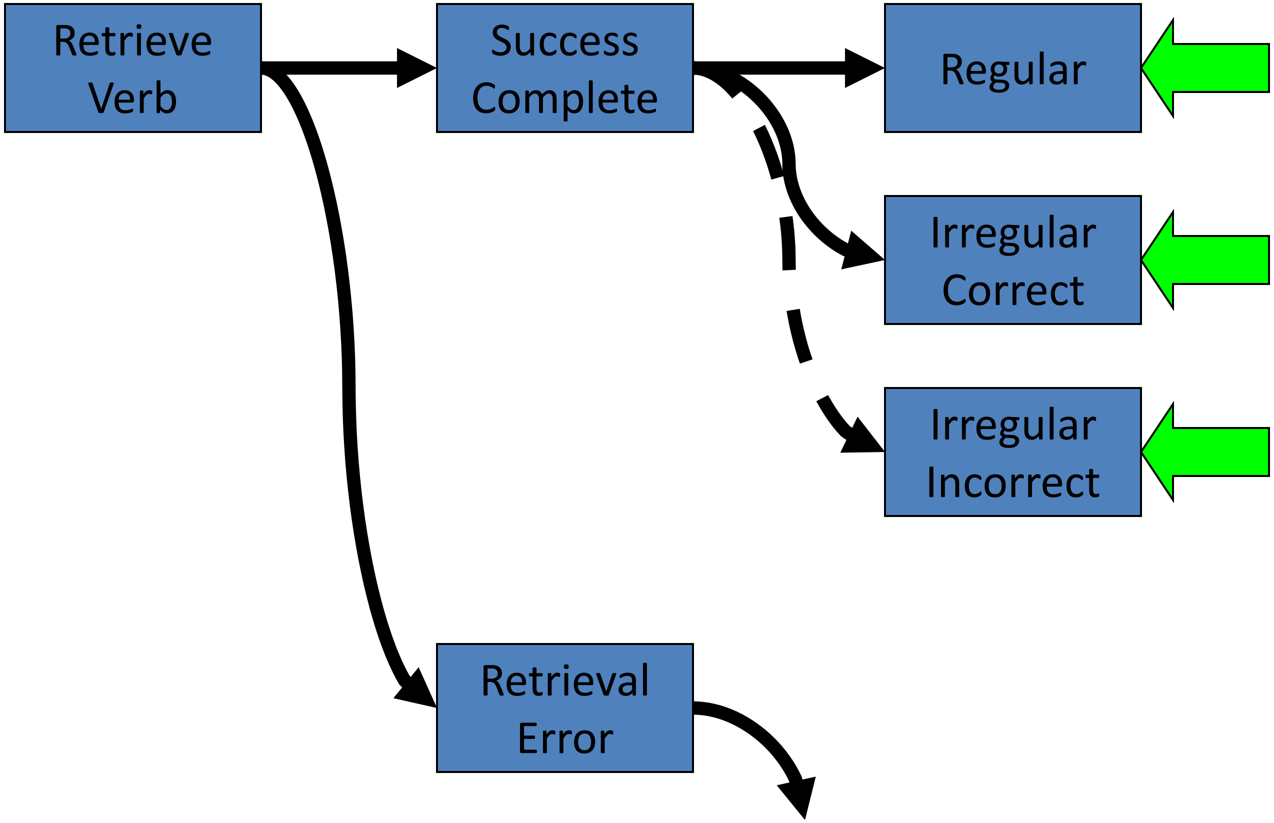
Irregular Incorrect

Unit 7 Homework

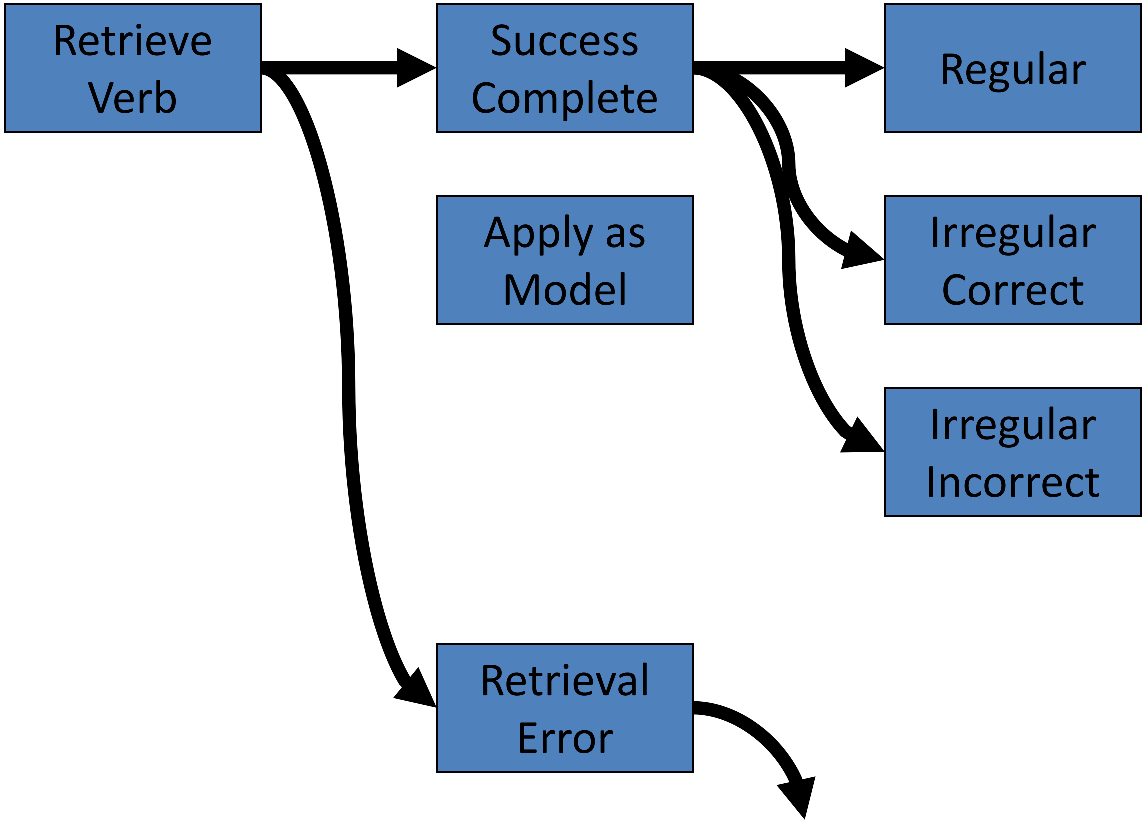
- Given:



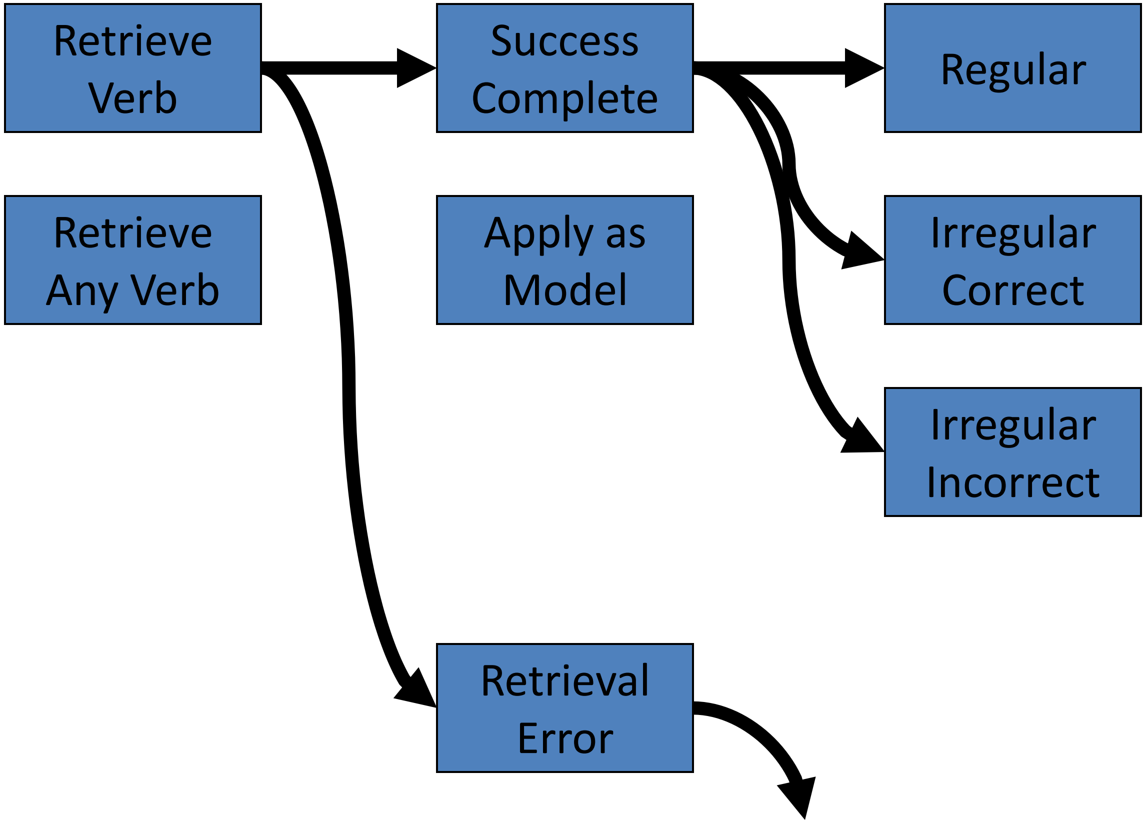
Unit 7 Homework



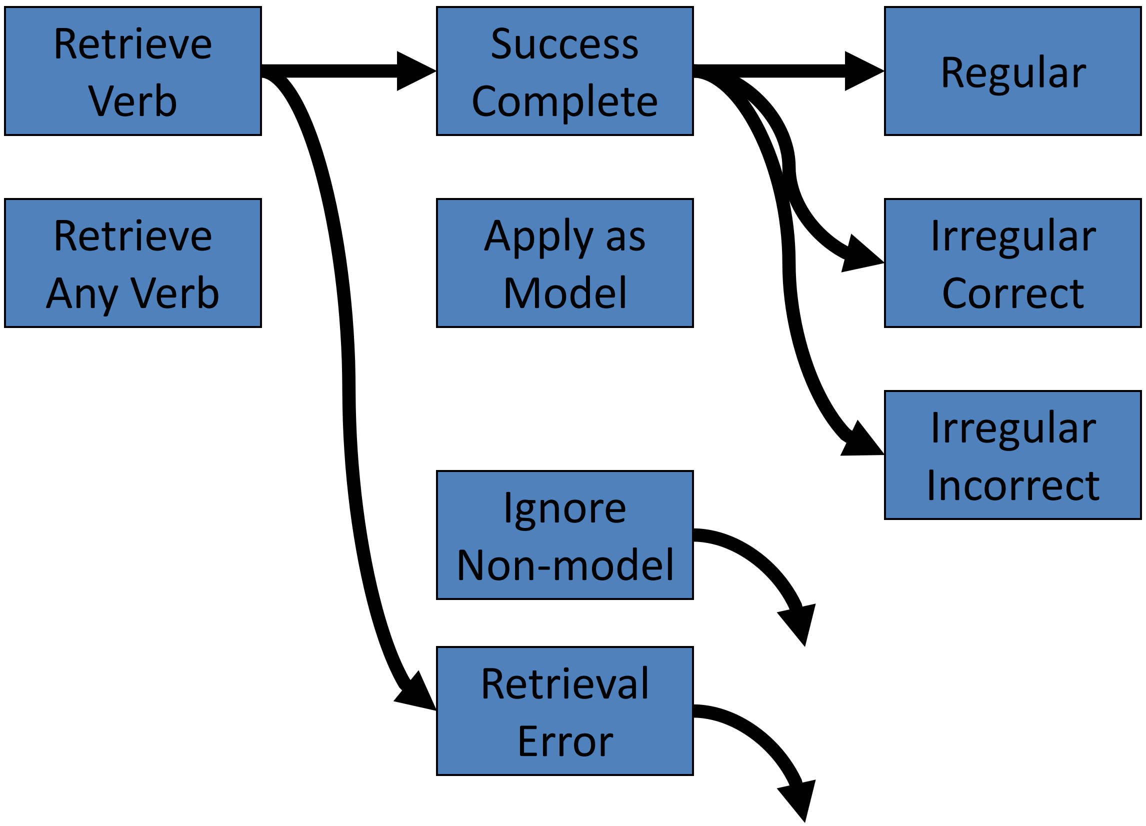
Unit 7 Homework



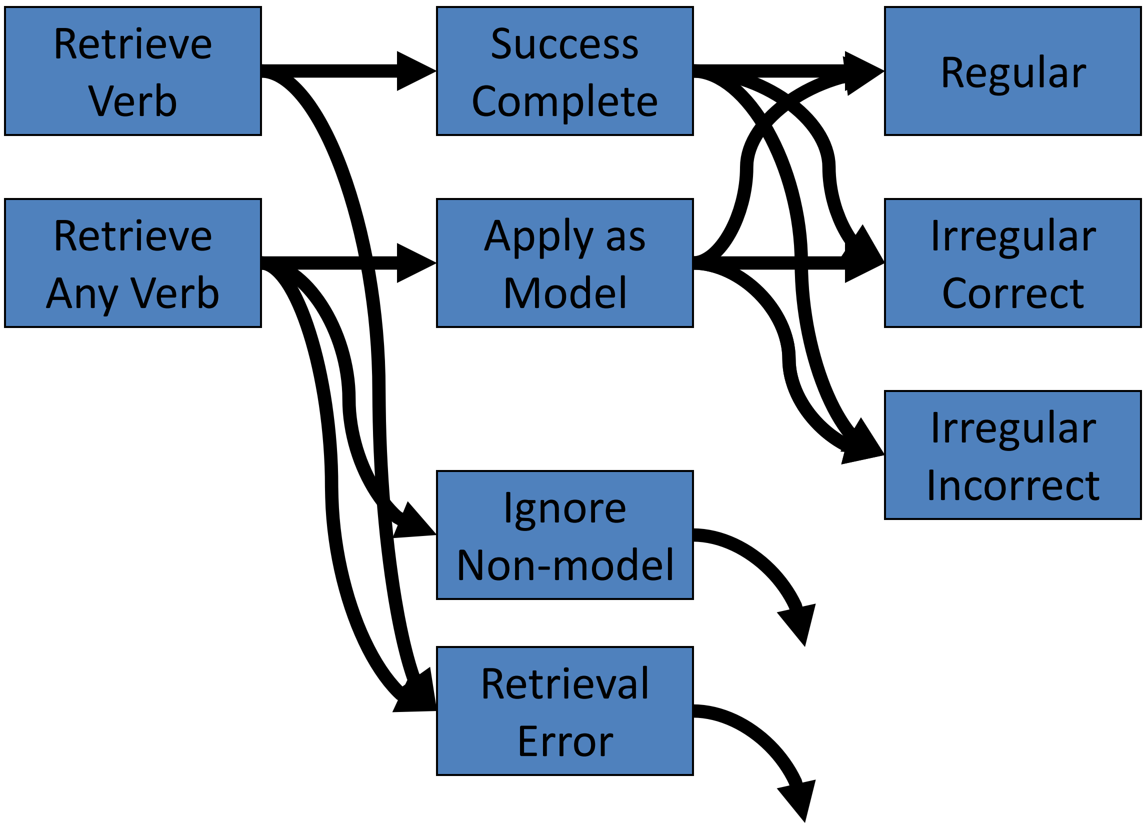
Unit 7 Homework



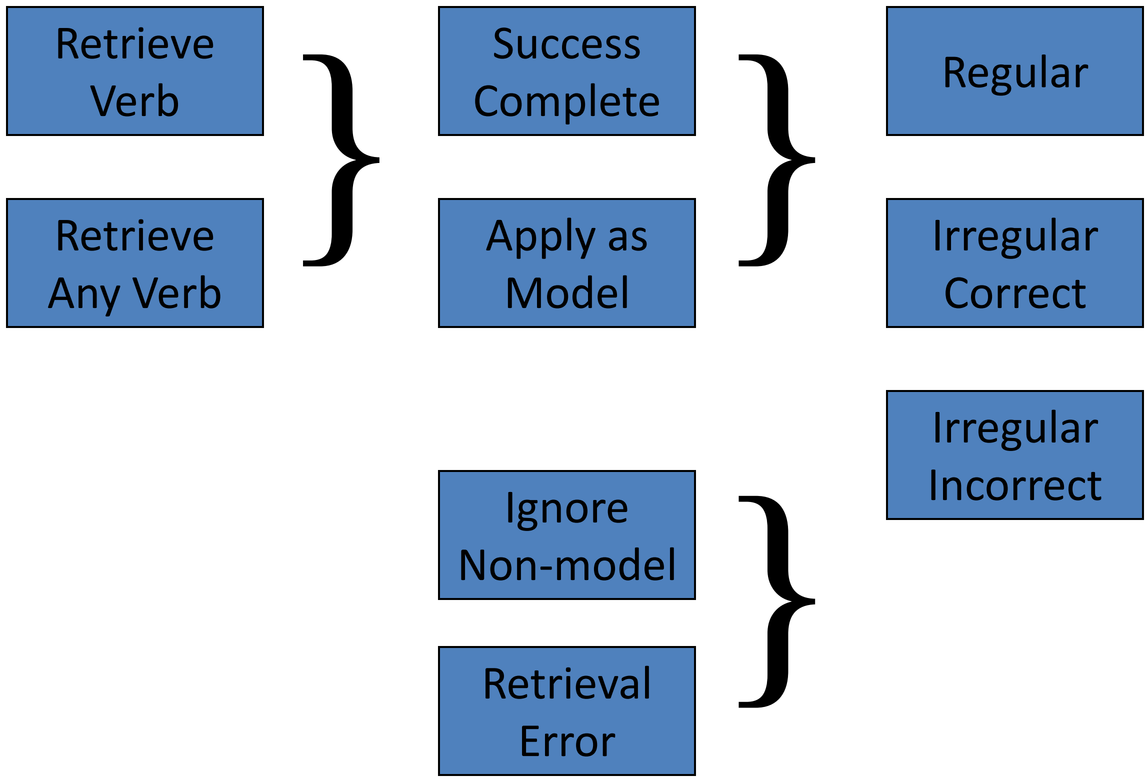
Unit 7 Homework



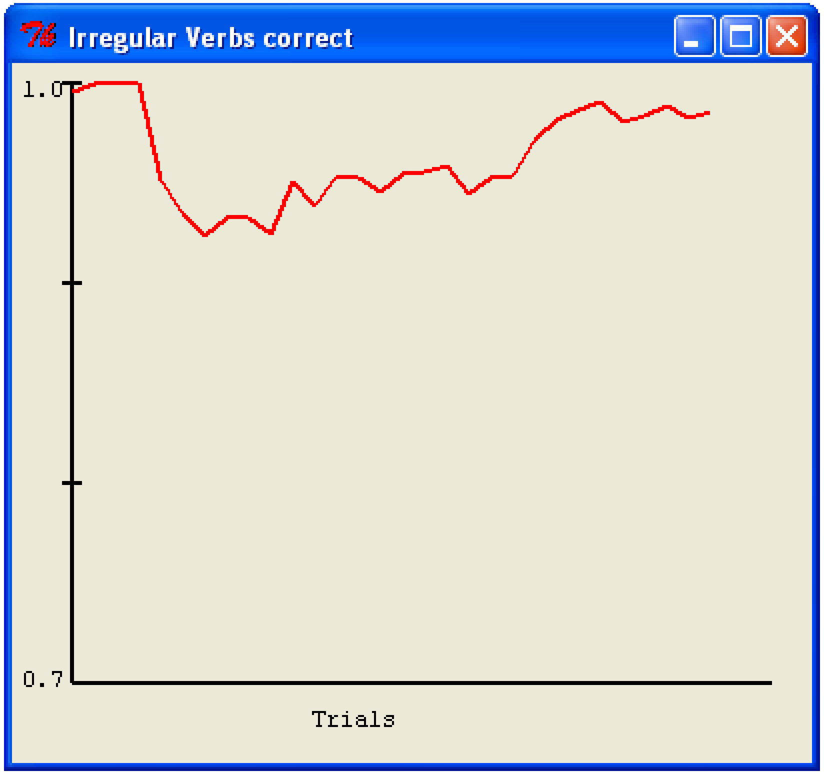
Unit 7 Homework

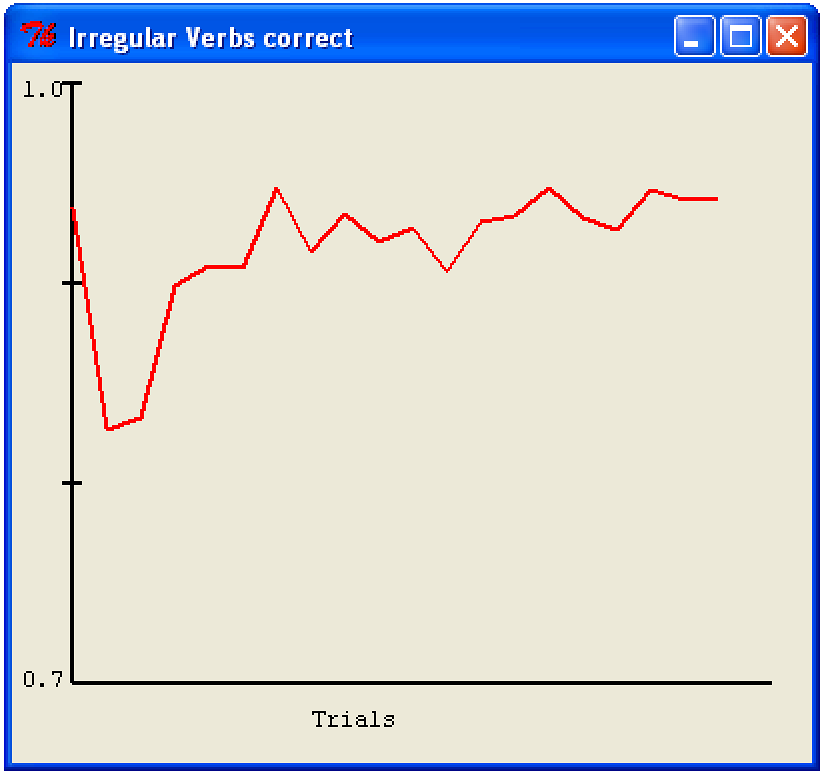


Unit 7 Homework



Unit 7 Homework





30,000 trials

10,000 trials 500 increment

Issues in
Cognitive Modeling

Issues in
Cognitive Modeling

- Philosophical issues
- Foundational issues
- Architectural issues
- Modeling issues
- Validation issues
- Scope issues

Philosophical Issues

Philosophical Issues

- Cognitive Science (& modeling) vs. Artificial Intelligence
- Sources of knowledge
- Mind-Body problem
- Computational Theory of Mind

Philosophical Issues

- Cognitive Science (& modeling) vs Artificial Intelligence
  - similar search for understanding:
    - what is intelligence
    - how the brain produces behavior
  - different purposes
    - understanding mind for science's sake
    - applying that understanding to problems

Philosophical Issues

- Source of knowledge
  - from experience (Locke, Hume, Mill)
  - innate, instinct, built-in (Decartes, Kant, & Pinker's "Language Instinct")

Philosophical Issues

- Mind-body problem
  - mind distinct from body (Dualism)
  - mind an extension of the body (Materialism)
  - reality is in the mind (Idealism)

Philosophical Issues

- Computational hypothesis
  - computation as a theory of mind
  - < 50 years old
  - John Searle's Chinese Room

Philosophical Issues

- Symbol hypothesis
  – symbols vs. Gestalt's holistic, parallel, analog perception of whole different as from parts
  – related to neural network approach to the Cognitive Science

Foundational Issues

Foundational Issues

- Symbol hypothesis (on boarder)
- Levels/bands – what are we studying?
- Where is foundation
- Cognitive plausibility

Newell's Levels & Bands

Architectural term	t (sec)	Units	System	Band
	10^{11-13}	10^4-10^6 years		Evolutionary
	10^{10}	Millennia		Historical
Lifetime	10^9	~50 years		Historical
	10^8	Years	(Expertise)	Historical
	10^7	Months	(Expertise)	Social
Development	10^6	Weeks		Social
	10^5	Days		Social
	10^4	Hours	Task	Rational
Knowledge acq	10^3	10 min	Task	Rational
	10^2	Minutes	Task	Rational
	10^1	10 sec	Unit task	Cognitive
Performance	10^0	1 sec	Operations	Cognitive
Temp storage	10^{-1}	100 ms	Deliberate act	Cognitive
Primitive act	10^{-2}	10 ms	Neural net	Biological
	10^{-3}	1 ms	Neuron	Biological
	10^{-4}	100 μs	Organelle	Biological

Foundational Issues

- Where is the foundation?
  - working from neuron up
  - theory of mind down, or
  - middle in both directions

Foundational Issues

- Cognitive plausibility
  - common usage
  - formal matching of clever experimental data
  - multi-level justification

Architectural Issues

Architectural Issues

- Goal of architecture
- Perception-cognition boundary
- Movement-movement boundary
- Symbolic memory representation Subsymbolic representation
- Strong assumptions

Architectural Issues

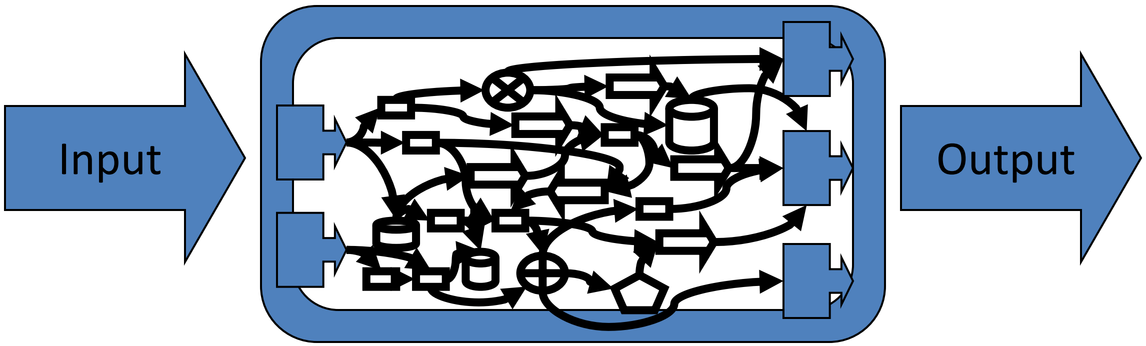
- Goal of architecture
  - AI or Cognitive Science
  - Narrow behavioral focus or broad

Architectural Issues

- Perception-cognition boundary

Architectural Issues

- Perception-cognition boundary



Perception Cognition Action

Architectural Issues

- Perception-cognition boundary
  - where's the edge?
  - how much does one affect the other?
  - being studied

Architectural Issues

- Movement-movement boundary
  - presumed well defined: thought vs. action
  - “Mirror neurons” challenging definition
  - not focus of study
  - (robotics not the same subject)

Architectural Issues

- Symbolic memory representation
  - LTM & STM accepted as different
  - Procedural generally accepted as different
  - Episodic different?
  - Images different?
  - Spatial different?
  - (Feelings different?)

Architectural Issues

- Subsymbolic representation
  – Generally accepted as necessary
  – Architectural design impact not settled
  – Neural representation?
  – Functional mathematical representation?
  – Stochastic representation?

Architectural Issues

- Strong assumptions
  - disprovable theoretic claims to advance science
  - Newell's UTC stake in the ground
  - Icarus' variation
  - Clarion (& ACT-R) neural representation dependency
  - Modularity and physiological comparisons

Modeling Issues

Modeling Issues

- Standard methodologies
- Bottom-up vs. top-down
- Chunk size
- Production size
- Parameter standardization
- Other: modeling or architecture issues

Modeling Issues

- Standard methodologies
  - Some:
    - rule based behavior
    - some patterns of rules
      - problem solving techniques
      - find, attend, harvest
      - prepare, execute
  - Lack of standard methodologies an indicator of the youth of the field or something else...

Modeling Issues

- Bottom-up vs. top-down
  - Goal driven or environmentally driven?
  - Goal: procedural process retrieval
  - Env: perception, reactive behavior
  - Mixed? How?

Modeling Issues

- Chunk size
  - How many slots?
    - Smaller is better
    - When is larger, too large, 7+/- 2? (!)
  - How complicated is a slot
    - indirect references?
    - variable slot names?
    - un-named slots?

Modeling Issues

- Production size
  - How many conditions?
  - How complicated is the logic?
    - and
    - not
    - or?
    - evaluation function?
  - How many actions on RHS?
  - How complicated are the actions?

Modeling Issues

- Parameter standardization
  (ACT-R)
    - Production firing: 50ms
    - Retrieval threshold: varies
    - Move attention: 85ms
    - Imaginal vs. goal buffer: ?
    - others...
  – Other architectures?

Modeling Issues

- Other: modeling or architecture issues
  - Memory for goals (are goals different?)
  - Multi-tasking (interleaved by model or architecture)
  - Emotion affecting cognition
    - different rules
    - different parameters (eg. RT)
  - Motivation(?)

Validation Issues

Validation Issues

- Validation criteria
- Validity vs. reliability
- Acceptable evidence

Validation Issues

- Validation criteria
  - Statistical hypothesis testing
  - Goodness of fit
  - Verbal protocols

Validation Issues

- Validity vs. reliability
  - Well defined phenomenon
  - Clearly demonstrated
  - Explanation rational
  - Explanation/model cognitively plausible

Validation Issues

- How demonstrated?
  - match single human on single experiment
  - match multiple subjects on single experiment
  - match data on a wide range of behaviors
  - replicated

Scope Issues

- Natural Language

- "Numerosity" (sense of quantity, size, etc.)
- Judgment
- Realistic vs. rational behavior
- Social behavior
- Abnormal behavior
- Creativity, art, music
- ...