Sentence Comprehension as a Cognitive Process
Day 2: Getting started with ACT-R modeling

Shravan Vasishth & Felix Engelmann

Source of these slides

- These slides are taken from Bill Kennedy’ s Sept 2011 slides entitled: **Notes on teaching ACT-R modeling**.
- I have just adapted them for the present course.

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 x.

```
(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

EXAMPLE:

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

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>                ← start a retrieval
      ISA    <chunk-type>
      <att> <value>
      ...
    -goal>                     ← explicit clearing of a buffer
    !output!     (text text =variable text =variable)
)
```

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 (from ACT-R tutorial)

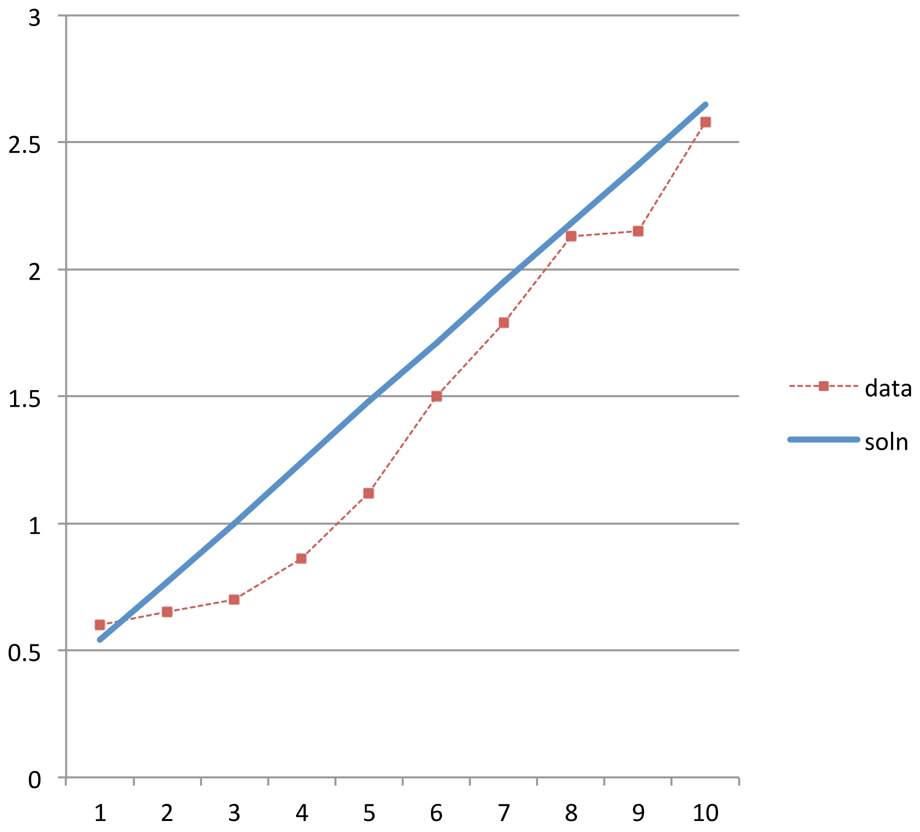
- Task: A bunch of objects appear on the display, report the number by speaking it
- Model starts with the counting facts from 0-11
- Will need to manage visual attention
  - Make sure the model gets to every item
  - Needs to know when its done
- Should not need to adjust parameters to get a reasonable 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

Memory’s Subsymbolic Representation

Memory’ s Subsymbolic Representation

```
(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
)
```

- At symbolic level
  - – chunks in DM
  - – retrieval process

- 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:

$$A_i = B_i + \varepsilon_i$$

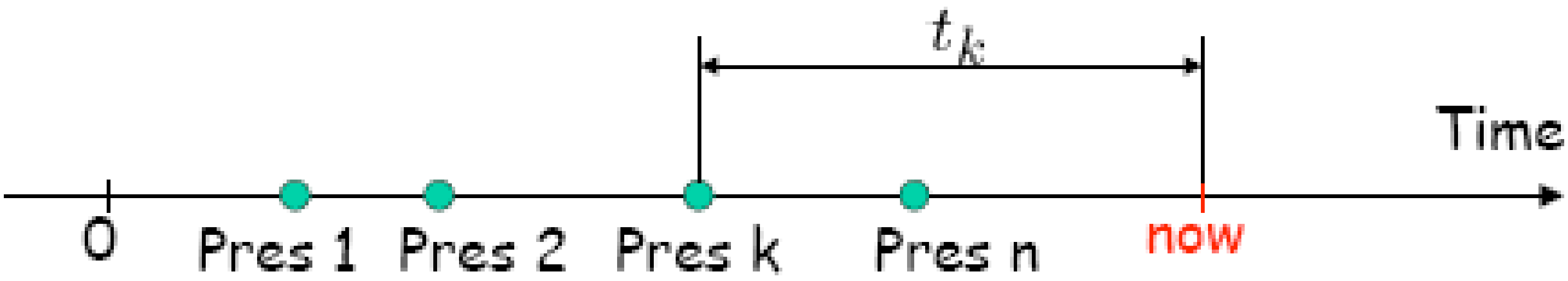
B_i = "Base-level activation"

ε_i = noise contribution

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

Memory's Subsymbolic Representation: Base-level Activation



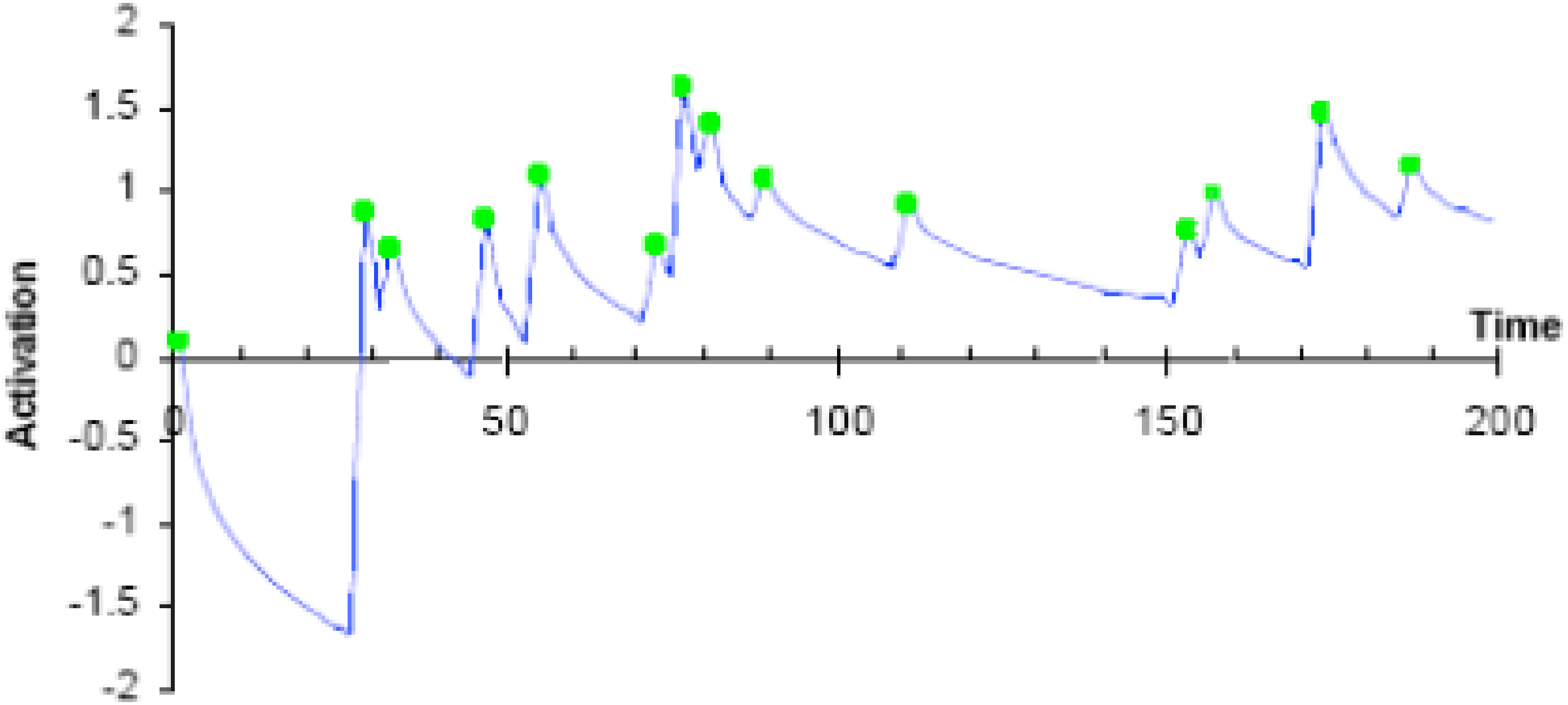
$$B_i = \ln\left(\sum_{k}^{n} t_k^{-d}\right)$$

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

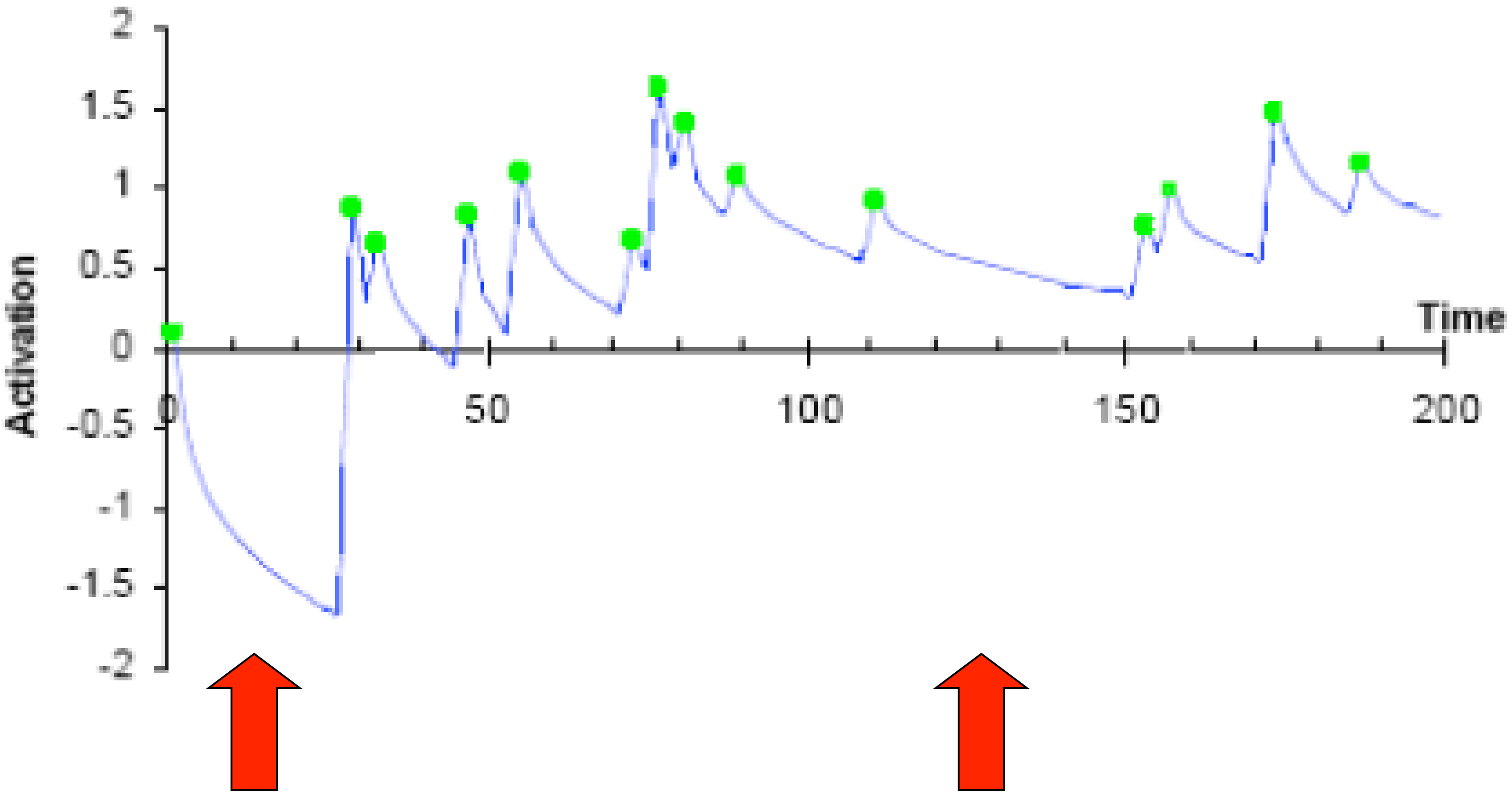
d decay parameter (sgp :bll 0.5)

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

Memory's Subsymbolic Representation: Base-level Activation



Memory’s Subsymbolic Representation: Base-level Activation



With use, decays less

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)

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: Activation

Activation for chunk i:

$$A_i = B_i + \varepsilon_i$$

B_i = "Base-level activation"

ε_i = noise contribution

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 necessary to match noise of human subjects...

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

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 context 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 to decide which to retrieve?
- 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):

$$A_i = \text{Base Level Activation} + \text{noise}$$
$$= B_i + \varepsilon_i$$

- 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

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

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))

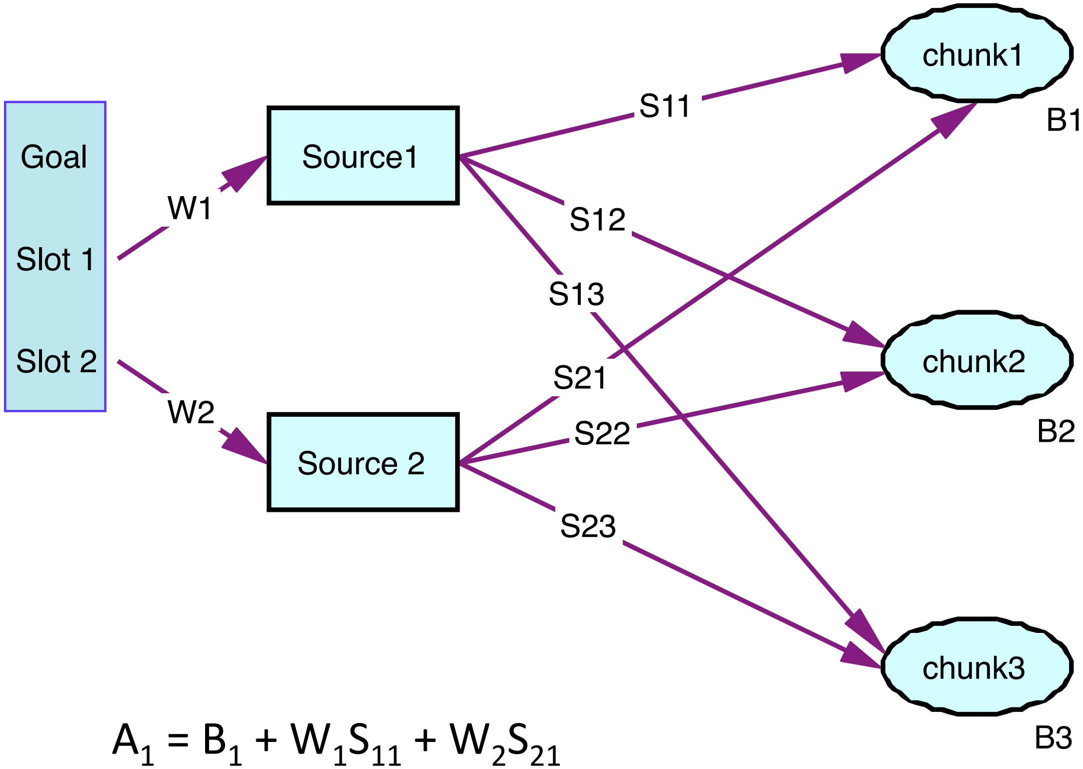
Spreading Activation

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



$A_1 = B_1 + W_1S_{11} + W_2S_{21}$
$A_2 = B_2 + W_1S_{12} + W_2S_{22}$
$A_3 = B_3 + W_1S_{13} + W_2S_{23}$

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>

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

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 <u>similar chunks</u>

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

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

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).

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.

ACT-R Model Development

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

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 ...

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
- ...

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)

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

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

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

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?