Unit 1 Review and Unit 2

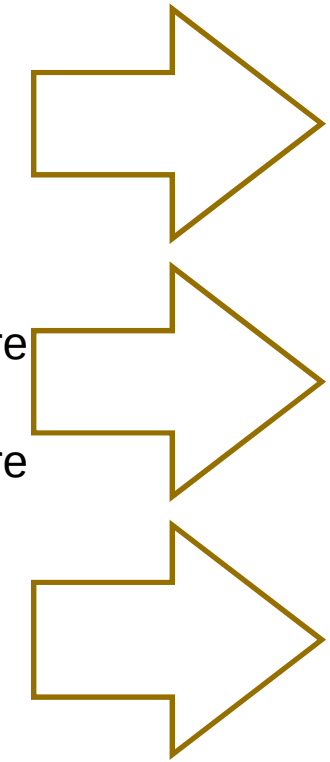
Dan Bothell
Department of Psychology
Carnegie Mellon University

Welcome

- The 17th Annual ACT-R Summer School

Schedule

- Monday (today)
  - Unit 2 – Dan Bothell
- Tuesday
  - Unit 3 – Dan Bothell
- Wednesday
  - Unit 4 – Christian Lebiere
- Thursday
  - Unit 5 – Christian Lebiere
- Friday
  - Unit 6 – John Anderson
- Saturday
  - Unit 7 – John Anderson



Symbolic: Attention & Motor actions

Sub-symbolic: Chunk Activation

Sub-symbolic: Production Utility

- Thursday August 5th
  - ACT-R workshop at ICCM

Daily

- About 8:30am
  - Baker Hall 340A Continental breakfast
- 9:00am
  - Baker Hall 340A Morning lecture
- About 10:30am
  - Lab in Baker Hall 332P (or free time if you want)
- 3:00pm
  - Baker Hall 340A Afternoon wrap-up
  - Instructor's research discussion
- After that
  - You're free, but remember the lab is still available

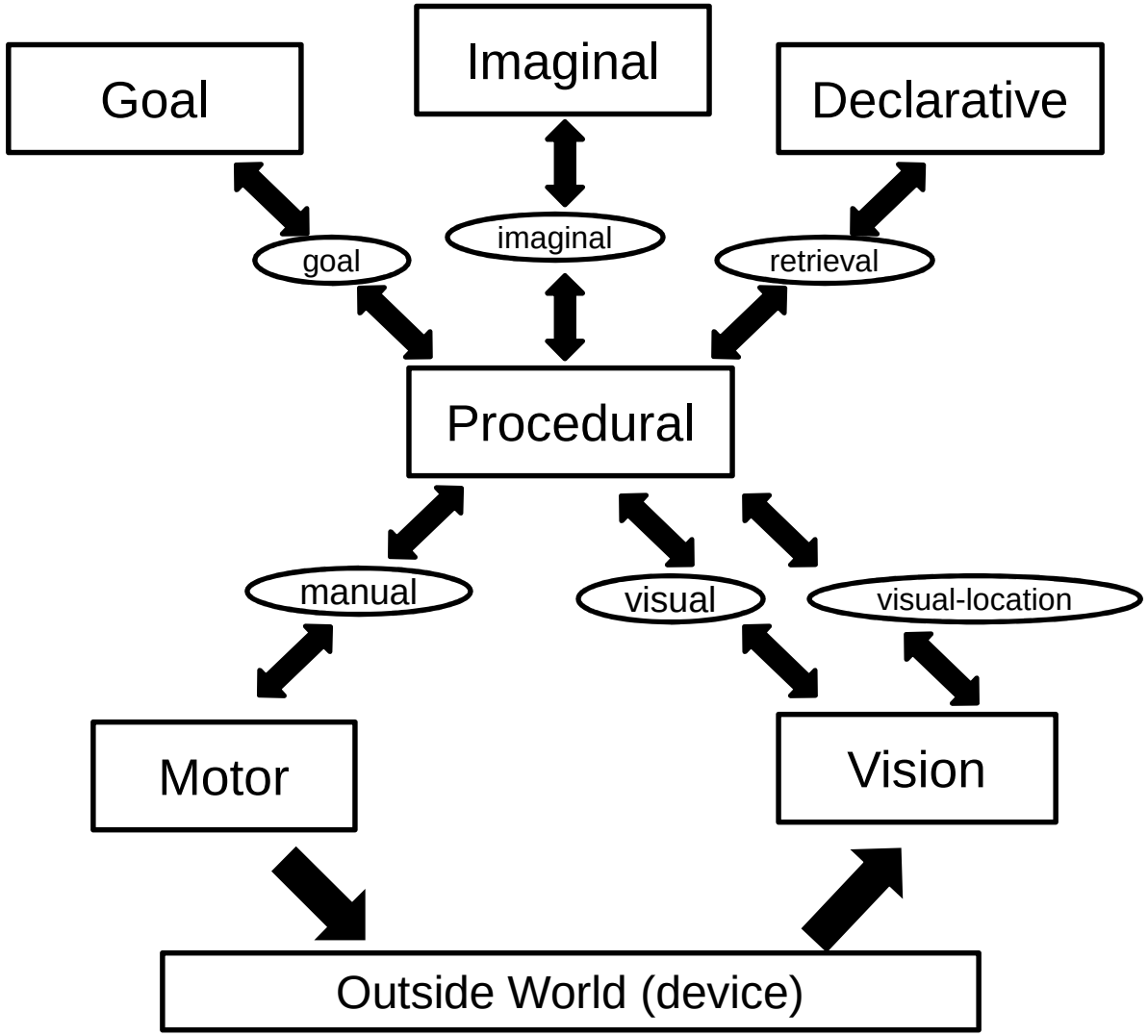
Pre-Summer School Assignment

- Everybody read and complete unit 1 exercises?
- Comments, questions or difficulties?

ACT-R overview

- A cognitive architecture
  - A set of special purpose modules
    - Declarative memory, procedural memory, motor control, vision, goals, etc
  - A description of how they are integrated
  - An association of modules to cortical regions

ACT-R 6 Overview



Unit 1 overview

- Introduction to knowledge structures
- Two types of knowledge in ACT-R
  - Declarative
  - Procedural

Declarative knowledge

- Knowledge we are aware of
  - 3 + 4 = 7
  - Cardinals are red birds
- Represented it ACT-R by chunks

Chunks

- Defined by their type and slots
- Type represents a category
  - Addition facts
  - Birds
- Slots represent attributes
  - Addends and sum
  - Color and size

Procedural knowledge

- Knowledge displayed in behavior
- Generally not consciously aware of it
  - Speaking a language
  - Driving a car with a manual transmission
    - learning productions from declarative in unit 7
- Represented in ACT-R by productions

Productions

- A condition and action pair
  - If my goal is to answer 3 + 4 and

    I have a chunk that says 3 + 4 =7

    then set the answer to 7
- When the condition (left-hand side) is met

  perform the actions (right-hand side)
- Only one production at a time can fire

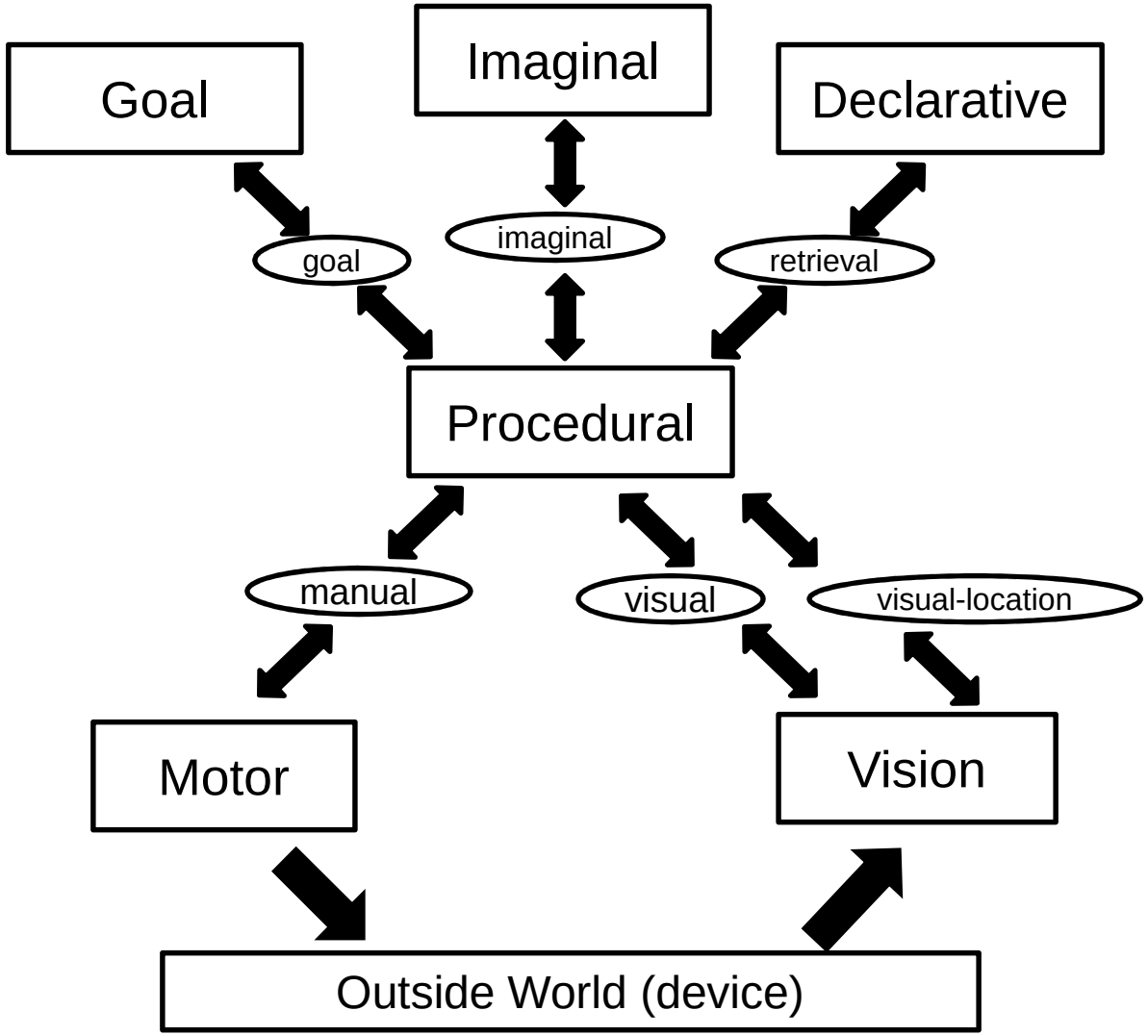
Productions (cont.)

- The conditions are a conjunction of tests
  - the contents of buffers
  - the state of the buffers
  - The state of the modules
- The actions perform
  - changes to the buffers
  - requests of the modules
- What is a buffer?

Buffers

- A buffer is the interface to a module
- Responds to queries about state of the module or buffer
- Accepts requests for the module
- Can hold one chunk
  - Typically placed there in response to a request
  - Scratch pad for chunk creation

ACT-R 6 Overview



Conditions (LHS)

- Buffer tests specify a buffer and pattern

```
=goal>
      isa find-sum
      addend1 =a1
      addend2 =a2
```

- Buffer queries specify a buffer and state-value pairs to check

```
?retrieval>
     state    free
     buffer  empty
```

- If all patterns match and all queries are true the production may be selected and fired

Actions (RHS)

- Three types of actions:
  - Clear the chunk from the buffer
    - Specified with a – before the buffer name

    -goal>
  - Modify the chunk in the buffer
    - Specified with an = before the buffer name and followed by slot-value pairs indicating the changes to make

    =goal>

      answer =sum
  - Request the module do something
    - Specified with a + before the buffer name followed by a description of the action to perform
    - Implicitly clears any chunk in the buffer

    +retrieval>

      isa add-fact

The Count model

- The trace shows what happens
- Questions before moving on?

Perceptual and Motor Modules

- Originally ACT-R/PM by Mike Byrne
  - Ran as an add-on for ACT-R 4.0
- Now an integrated component of ACT-R 6.0
- Based on Visual Interface and EPIC
- Situates ACT-R models in a “real” world
- Provides precise timing information

Demo2

- ACT-R interacts with experiments that people can interact with
  - Mechanisms provided that work with existing Lisp based windows/experiments
  - Some tools provided to make simple experiments easy to generate (hopefully)
  - Not tools one would want to use for real (human) data collection however if timing is important

Imaginal Module

- Internal imagery
  - Task representation
- Similar to the goal module
- Has one buffer
  - Imaginal

Imaginal module requests

- Every request creates a new chunk
  - Same as the goal buffer requests
- Unlike the goal module it takes time to do so
  - Default of 200ms

Imaginal buffer request

```
(P encode-letter
   =goal>
      ISA         read-letters
      state       attend
   =visual>
      ISA         text
      value       =letter
==>
   =goal>
      state       respond
  +imaginal>
      isa         array
      letter      =letter
)
```

```
0.235   PROCEDURAL      PRODUCTION-FIRED ENCODE-LETTER
…
0.235   PROCEDURAL      MODULE-REQUEST IMAGINAL
…
0.235   PROCEDURAL      CLEAR-BUFFER IMAGINAL
…
0.435   IMAGINAL        CREATE-NEW-BUFFER-CHUNK IMAGINAL ISA ARRAY
0.435   IMAGINAL        SET-BUFFER-CHUNK IMAGINAL ARRAY0
```

```
> (buffer-chunk imaginal)
IMAGINAL: ARRAY0-0
ARRAY0-0
   ISA ARRAY
    LETTER  "v"
```

Vision Module

- Models visual attention – not eye positions
  - EMMA module (an extra available for ACT-R 6)
- Contains a Where and What distinction
- Interaction through two buffers
  - Visual-location
  - Visual

Vision module's representation of the world

- Set of objects with specific features
- Vision module does not construct representations
  - Object features are predefined by the modeler
  - Some default mechanisms built in
    - Text
    - Buttons
- The visicon (visual icon)
  - The currently available objects
  - Print-visicon command will show the objects and standard features

```
>(print-visicon)
Loc         Att   Kind          Value             Color           ID
---------   ---   ------------  ----------------  --------------  ------------
( 29  20)   NEW   TEXT          "hello"           BLACK           VISUAL-LOCATION0
( 65  60)   NEW   OVAL          OVAL              GRAY            VISUAL-LOCATION2
( 65  60)   NEW   TEXT          "ok"              BLACK           VISUAL-LOCATION1
( 94  20)   NEW   TEXT          "there"           RED             VISUAL-LOCATION3
```

Visual-location buffer ("where")

- Request the location of some object
- Visual / Spatial Properties
  - Screen-x
  - Screen-y
  - Color
- Attentional status
  - :Attended

Visual-location request

```
(P find-unattended-letter
   =goal>
      ISA         read-letters
      state       start
 ==>
   +visual-location>
      ISA         visual-location
      :attended    nil
   =goal>
      state       find-location
)
```

```
0.050   PROCEDURAL     PRODUCTION-FIRED FIND-UNATTENDED-LETTER
0.050   PROCEDURAL     MOD-BUFFER-CHUNK GOAL
0.050   PROCEDURAL     MODULE-REQUEST VISUAL-LOCATION
0.050   PROCEDURAL     CLEAR-BUFFER VISUAL-LOCATION
0.050   VISION         Find-location
0.050   VISION         SET-BUFFER-CHUNK VISUAL-LOCATION VISUAL-LOCATION0-0
```

Visual-location buffer

- Returns a result immediately
  - Pre-attentive processing
- Either returns a chunk of type visual-location

```
VISUAL-LOCATION0-0
 ISA VISUAL-LOCATION
  SCREEN-X  130
  SCREEN-Y  160
  DISTANCE  15.0
  KIND  TEXT
  COLOR  BLACK
  VALUE  TEXT
  HEIGHT  10
  WIDTH  7
  SIZE  0.20
```

- Or Fails
  - Buffer remains empty
  - State error query is then true

    ?visual-location>

    state error
  - May also be queried as

    ?visual-location>

    error t

Visual buffer ("what")

- Shift attention to a visual-location
- Create a representation of the object there
  - An episodic not semantic representation
  - A retrieval needed to make that mapping
- Mark the object as having been attended

Visual request

```
(P attend-letter
   =goal>
      ISA         read-letters
      state       find-location
   =visual-location>
      ISA         visual-location
   ?visual>
      state        free
==>
   +visual>
      ISA         move-attention
      screen-pos  =visual-location
   =goal>
      state       attend)
```

```
0.100   PROCEDURAL      PRODUCTION-FIRED ATTEND-LETTER
0.100   PROCEDURAL      CLEAR-BUFFER VISUAL-LOCATION
0.100   PROCEDURAL      CLEAR-BUFFER VISUAL
0.100   PROCEDURAL      CONFLICT-RESOLUTION
0.100   VISION          Move-attention VISUAL-LOCATION0-0-1 NIL
0.100   PROCEDURAL      CONFLICT-RESOLUTION
0.185   VISION          Encoding-complete VISUAL-LOCATION0-0-1 NIL
0.185   VISION          SET-BUFFER-CHUNK VISUAL TEXT0
```

Visual buffer

- Requires time to shift and encode
  - 85ms by default
- Either returns a chunk which is a subtype of visual-object

```
TEXT0
 ISA TEXT
  SCREEN-POS  LOC10
  VALUE  "v"
  STATUS  NIL
  COLOR  BLACK
  HEIGHT  10
  WIDTH  7
```

- Or Fails
  - Buffer remains empty
  - State error query is true

```
?visual>
      state     error
```

Harvesting a Visual Object

```
(P encode-letter
   =goal>
      ISA         read-letters
      state       attend
   =visual>
      ISA         text
      value       =letter
==>
   =goal>
      state       respond
  +imaginal>
      isa         array
      letter      =letter
)
```

- Find/Attend/Harvest typical usage

Visual State Query

```
(P attend-letter
   =goal>
      ISA         read-letters
      state       find-location
   =visual-location>
      ISA         visual-location
   ?visual>
      state        free
==>
   +visual>
      ISA         move-attention
      screen-pos  =visual-location
   =goal>
      state       attend)
```

- Represents internal state of vision module
  - Free or busy
- Need to check before making a request
  - Because request takes time
- Do not want to make overlapping requests
  - Important for many modules/buffers
  - Avoid “jamming” the module

Automatic Visual Re-encoding

- If an object is attended and the screen changes there is an automatic re-encoding of that object
- The module is busy during that time
- Be careful when working with changing displays

```
0.885   MOTOR            OUTPUT-KEY #(4 5)
0.885   PROCEDURAL       CONFLICT-RESOLUTION
0.970   VISION           Encoding-complete VISUAL-LOCATION0-0-1 NIL
0.970   VISION           No visual-object found
```

Motor Module

- ACT-R’s hands
- Can manipulate a keyboard or mouse
  - Other “devices” can be implemented
- Based on EPIC's Manual Motor Processor
- Contains detailed timing information

The manual buffer

- Commands are sent to the hands through the manual buffer
- Only takes requests (RHS)
  - Does not respond with a chunk in return
- Type of action requested by the isa value
  - Called the “style” of the action
- Details provided in the slots

Manual request

```
(p respond
   =goal>
      isa read-letters
      state respond
   =imaginal>
      isa array
      letter =letter
   ?manual>
      state free
 ==>
   =goal>
      state done
   +manual>
      isa press-key
      key =letter
)
```

```
0.285   PROCEDURAL   PRODUCTION-FIRED RESPOND
0.285   PROCEDURAL   MODULE-REQUEST MANUAL
0.285   PROCEDURAL   CLEAR-BUFFER MANUAL
0.285   MOTOR        PRESS-KEY v
0.285   PROCEDURAL   CONFLICT-RESOLUTION
0.535   MOTOR        PREPARATION-COMPLETE
0.535   PROCEDURAL   CONFLICT-RESOLUTION
0.585   MOTOR        INITIATION-COMPLETE
0.585   PROCEDURAL   CONFLICT-RESOLUTION
0.685   MOTOR        OUTPUT-KEY #(4 5)
0.685   PROCEDURAL   CONFLICT-RESOLUTION
0.835   MOTOR        FINISH-MOVEMENT
```

Keyboard presses

- The press-key style
- Assumes hands start at the home row
- Return there when complete
- Sufficient for simple models
  - Competent but not an expert typist

Unit2 exercise

- One of these things is not like the others
- 3 letters presented
- 2 are the same
- Press the other one

Questions

- Any questions?

Baker Hall 332P Lab

- Open from 9am until 5pm
- Locked but accessible 7am-9am and 5pm-11:30pm
  - Can enter with the code xxxxx
- Alarmed from midnight until 7am
  - If you're inside you will likely get to meet campus security