Computing Dynamic Meanings: Building Integrated Competence-Performance Theories for Semantics

Day 1, part 1: Introduction to ACT-R

Jakub Dotlačil & Adrian Brasoveanu

ESSLLI 2018, August 6 2018

Course plan

- Providing a framework to connect theoretical linguistics to performance behavioral measures (on-line data) in a formally and computationally explicit way
- Applying the framework to examples from syntax & semantics, and on several experimental types (self-paced reading, eye tracking...)
- Hands-on (Python3 code supplied and discussed)
- Upcoming book – Brasoveanu and Dotlačil (in prep.)

Course plan by day [subject to change]

- **Monday:** Intro to the ACT-R cognitive architecture (Adaptive Control of Thought-Rational) and the *pyactr* Python3 library
- **Tuesday:** Syntactic parsing and Bayesian methods of model fitting
- **Wednesday:** Embedding ACT-R models of linguistic phenomena into Bayesian models $\rightarrow$ first examples of modeling experimental data
- **Thursday:** DRT (Discourse representation Theory) and ACT-R, modeling memory recall and self-paced reading data
- **Friday:** extensions – more memory recall, psycholinguistic corpora and their modeling

Practicalities

- Advanced course – combination of several topics not often combined
- Knowledge of Python useful, but not required
- Slides & code available at: https://people.ucsc.edu/~abrsvn/esslli-2018-course.html

Today's plan

- Intro into ACT-R (Adaptive Control of Thought-Rational) & *pyactr*
- Toy examples of models in *pyactr*

Introduction to ACT-R

- Cognitive architecture
  - A theory about the structure of the human mind
  - Summary of various cognitive sub-disciplines into one model
  - ACT-R, Soar, [EPIC, Connectionist / Neural network models]

ACT-R – a bit of history

- Developed in the 70's and 80's as ACT (Adaptive Control of Thought)
- John R. Anderson, inspired by Allen Newell
- In the 90's – ACT-R (Adaptive Control of Thought-Rational)
- In the 00's and later – focus on neural implementation

Anderson and Lebiere (1998); Anderson et al. (2004); Anderson (2007)

ACT-R – what can it do?

- It models cognitive components (memory, reasoning...) and interfaces (visual, motor modules...)
- It models (simulates) human performance (reaction times, accuracies) and neurobehavioral data (EEG, brain images)
- Traditionally, mainly used to model responses and reaction times (but cf. Anderson 2007, 2012)

ACT-R

- Symbolic and subsymbolic systems meet (hybrid architecture)
- abstract, symbolic structures to describe human knowledge
- subsymbolic part to describe human performance
- modular
- Strengths: hybrid (theoretical linguistics friendly); interaction of modules; memory
- Weaknesses: garden of forking paths; hand-coding; overfitting (but this is a problem for all complex statistical models)

ACT-R

2 main types of modules:

- interacting with environment (perceptual and motor actions...)
- representing internal cognitive capabilities

ACT-R

2 types of knowledge

- declarative knowledge
- procedural knowledge

ACT-R

2 types of knowledge

- declarative knowledge
  - knowledge of facts
  - the current king of the Netherlands
  - 2 + 5 = 7
  - lexical knowledge
- procedural knowledge
  - knowledge displayed in behavior
  - how to drive / walk / swim / ride a bicycle

Declarative knowledge in ACT-R

- encapsulated in **chunks**
- attribute-value matrices / feature structures / sets of slot-value pairs

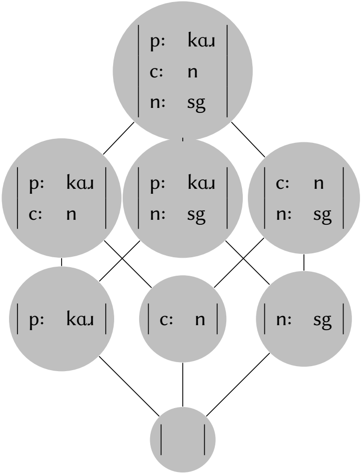
$$\left|\begin{array}{ll} \textsc{phonology}: & /\text{kɑɹ}/ \\ \textsc{meaning}: & [\![\text{car}]\!] \\ \textsc{category}: & noun \\ \textsc{number}: & sg \end{array}\right|$$

Relation between chunks

- $c_1 = c_2$ iff c_1, c_2 have the same slot-value pairs
- $c_1 \leq c_2$ iff c_1 carries less information than/is more general than/subsumes c_2
- $c_1 \leq c_2$ iff the slots in c_1 are in c_2 and for each slot in c_1 the value of slot is identical to the value of the same slot in c_2

$$\left|\begin{array}{ll} \text{PHONOLOGY}: & /\text{kɑɹ}/ \\ \text{MEANING}: & [\![\text{car}]\!] \\ \text{NUMBER}: & sg \end{array}\right| \leq \left|\begin{array}{ll} \text{PHONOLOGY}: & /\text{kɑɹ}/ \\ \text{MEANING}: & [\![\text{car}]\!] \\ \text{CATEGORY}: & noun \\ \text{NUMBER}: & sg \end{array}\right|$$

Relation between chunks



Relation between chunks

- $c_1 \sqcap c_2$ – meet of c_1 and c_2
- $c_1 \leq c_2 \Leftrightarrow c_1 \sqcap c_2 = c_1$
- chunks in general form a pseudocomplemented semi-lattice, $\langle C, \sqcap \rangle$
  cf. unification-based grammars (LFG, HPSG, Shieber (2003))
- the empty chunk is the bottom element (no slot-value specified)
- the unification (join) operation $\sqcup$ is not always defined (no contradicting knowledge allowed)

More on chunks

- Chunks can carry a negative value or a variable (such chunks are never part of the declarative memory)

$$\begin{vmatrix} \textsc{phonology}: & /\text{kaɹ}/ \\ \textsc{meaning}: & \sim [\![\text{boy}]\!] \\ \textsc{number}: & sg \end{vmatrix} \leq \begin{vmatrix} \textsc{phonology}: & /\text{kaɹ}/ \\ \textsc{meaning}: & [\![\text{car}]\!] \\ \textsc{category}: & noun \\ \textsc{number}: & sg \end{vmatrix}$$

More on chunks

- Chunks can carry a negative value or a variable (such chunks are never part of the declarative memory)

$$\begin{vmatrix} \text{PHONOLOGY}: & /\text{kaɹ}/ \\ \text{MEANING}: & =x \\ \text{NUMBER}: & sg \end{vmatrix} \leq \begin{vmatrix} \text{PHONOLOGY}: & /\text{kaɹ}/ \\ \text{MEANING}: & [\![\text{car}]\!] \\ \text{CATEGORY}: & noun \\ \text{NUMBER}: & sg \end{vmatrix}$$

More on chunks

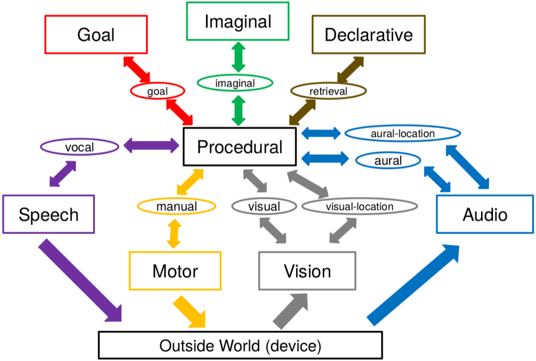
- Chunks are recursive (values of chunks can be chunks)

$$\begin{vmatrix} \text{PHONOLOGY}: & /\text{kɑɹ}/ \\ \text{MEANING}: & \begin{vmatrix} \text{CONSTANT\_NAME}: & car' \\ \text{ARITY}: & 1 \end{vmatrix} \\ \text{NUMBER}: & sg \end{vmatrix}$$

Modules and buffers

- ACT-R is modular (declarative module, procedural module...)
- Modules are not directly accessible – they can only be accessed through buffers
- Buffers serve a dual function:
  - individually, they provide the interface to modules
  - as a whole, they represent agent's current state; productions fire based on contents of buffers
- Buffers can hold only one chunk (cognitive 'bottleneck')

ACT-R in one picture



Bothell: slides, Introduction to ACT-R

Procedural knowledge in ACT-R

A condition and an action:

- When the condition (left-hand side) is met, perform the action (right-hand side)
- Many productions, but only one can fire at a time (another cognitive ‘bottleneck’)

Procedural knowledge in ACT-R

Left-hand side:

- Specify a buffer – a chunk in condition must subsume it

Right-hand side:

- Specify a buffer (use *=buffer> in pyactr*), specify how the current chunk must be modified
- Specify a buffer (use *+buffer> in pyactr*), specify what chunk must be created
- Flush a buffer (use *~buffer> in pyactr*); the chunk is automatically harvested and stored in declarative memory

Example: numerical quantifiers

– Evaluating numerical quantifiers relative to visual display
– Computable by finite-state machines

- There is more than 1 dot.

start: goal buffer – [counted: 0 end: 2]

Rule1	
=goal>	
counted	0
end	2
=visual>	
value	dot
==>	
=goal>	
counted	1
+visual>	
cmd	move

Rule2	
=goal>	
counted	1
end	2
=visual>	
value	dot
==>	
=goal>	
counted	2
+visual>	
cmd	move

Rule3	
=goal>	
counted	2
end	2
=visual>	
value	dot
==>	
-goal>	

Declarative memory: basic subsymbolic components

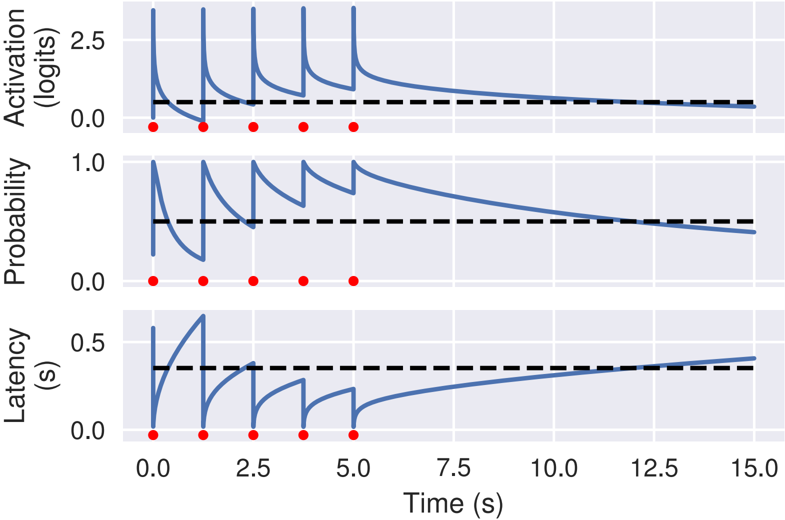
- ACT-R: retrieval from declarative memory is a power function of time elapsed since item presentation
- the power function is used to compute (base) activation and is based on the number of practice trials / 'rehearsals' of a word (1) (free parameters enumerated in parentheses)
- activation of an item is in turn used to compute accuracy (2) and latency (3) for retrieval processes

(1) $A_i = \log\left(\sum_{k=1}^{n} t_k^{-\mathbf{d}}\right)$ ($\mathbf{d}$: decay)

(2) $P_i = \frac{1}{1+e^{-\frac{A_i-\tau}{\mathbf{s}}}}$ ($\mathbf{s}$: noise, τ: threshold)

(3) $T_i = \mathbf{F}e^{-\mathbf{f}A_i}$ ($\mathbf{F}$:factor, $\mathbf{f}$: exponent)

Figure: Activation, retrieval probability and retrieval latency as a function of time (threshold – dotted black line; 5 presentations – red)



Example: frequency effects in lexical decision

- for any word, every time a speaker is exposed, the presentation contributes to its activation
- the ‘schedule of presentations’ is determined by a word’s frequency (we ignore other factors in this model)
- we predict shorter times of retrieval and higher accuracy for high frequency words
- predictions confirmed: we come back to this

On to some basic *pyactr* models ...

Anderson, John R. 2007. *How can the human mind occur in the physical universe?*. Oxford University Press.

Anderson, John R. 2012. Tracking problem solving by multivariate pattern analysis and hidden markov model algorithms. *Neuropsychologia* 50:487–498.

Anderson, John R., Daniel Bothell, and Michael D. Byrne. 2004. An integrated theory of the mind. *Psychological Review* 111:1036–1060.

Anderson, John R., and Christian Lebiere. 1998. *The atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Brasoveanu, Adrian, and Jakub Dotlačil. in prep. *Formal Linguistics and Cognitive Architecture*. Language, Cognition, and Mind (LCAM) Series. Dordrecht: Springer. The *pyactr* library (Python3 ACT-R) is available here: `https://github.com/jakdot/pyactr`.

Shieber, Stuart M. 2003. *An introduction to unification-based approaches to grammar*. Microtome Publishing.