University of Amsterdam

Institute for Logic, Language and Computation

Proportional and non-proportional quantifiers in ACT-R

CoSaQ Workshop, 28/9–29/9

Jakub Dotlačil

Plan

- A quick summary of ACT-R (Adaptive Control of Thought-Rational)
- A few observations on how ACT-R accounts for counting
- A few observations on how ACT-R can compute proportional and non-proportional quantifiers

All models and slides available at:
https://www.github.com/jakdot/conferences/

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, neural networks, Google DeepMind ...

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
- In the 00's and later – applied to (psycho)linguistics

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)
- In linguistics, it has been mainly used to model responses and reaction times

ACT-R

- abstract, symbolic structures to describe human knowledge
- subsymbolic part to describe human performance
- modular
- Strength – Interaction of modules; memory
- Weakness – Garden of forking paths; hand-coding and overfitting

ACT-R

2 main 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 behaviour
  - how to drive
  - how to walk

Declarative knowledge in ACT-R

- encapsulated in **chunks** (attribute-value matrices – HPSG-style, Pollard and Sag 1994)
- slot-value pairs

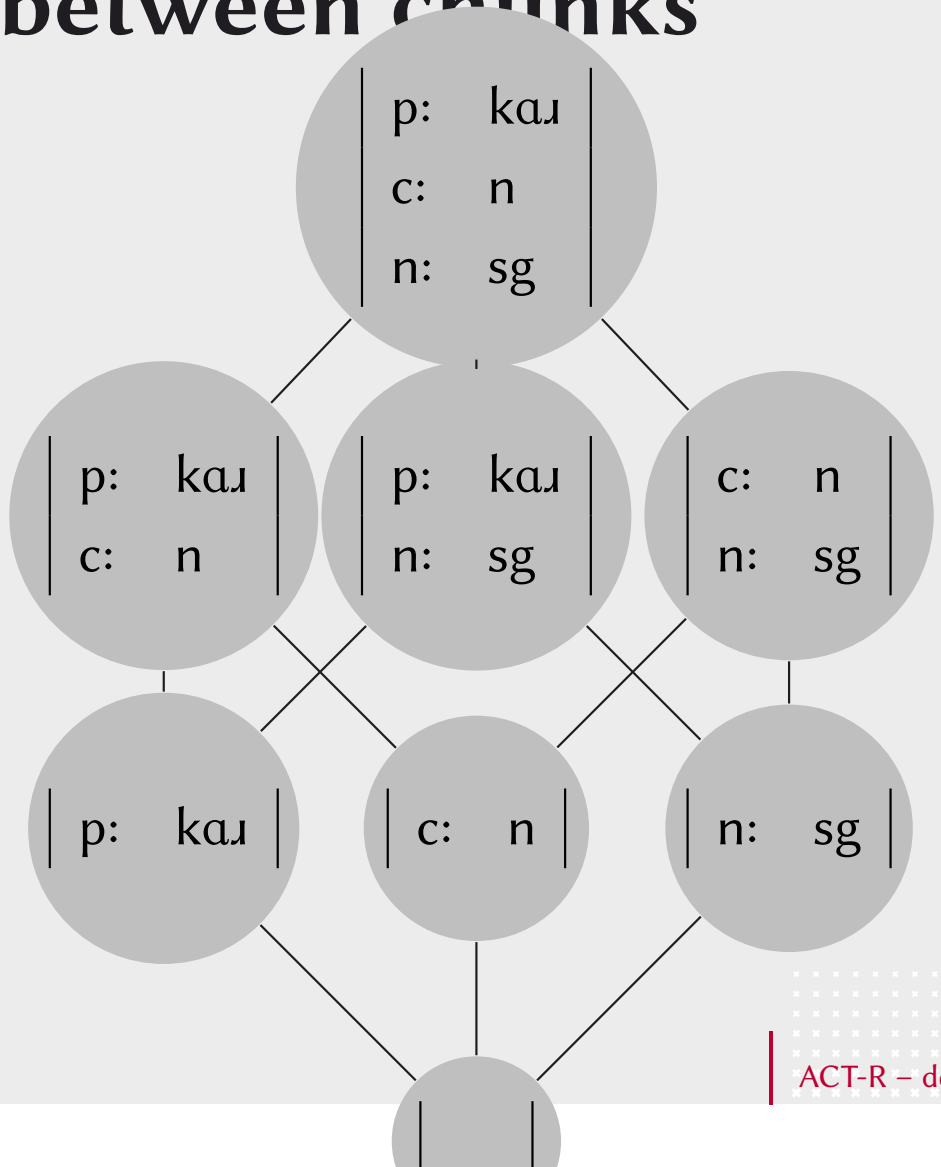
PHONOLOGY :	/kɑɹ/
MEANING :	⟦car⟧
CATEGORY :	*noun*
NUMBER :	*sg*

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

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

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 complemented 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 reverse operation, $\sqcup$, is not always defined (no contradicting knowledge allowed)

More on chunks

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

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

More on chunks

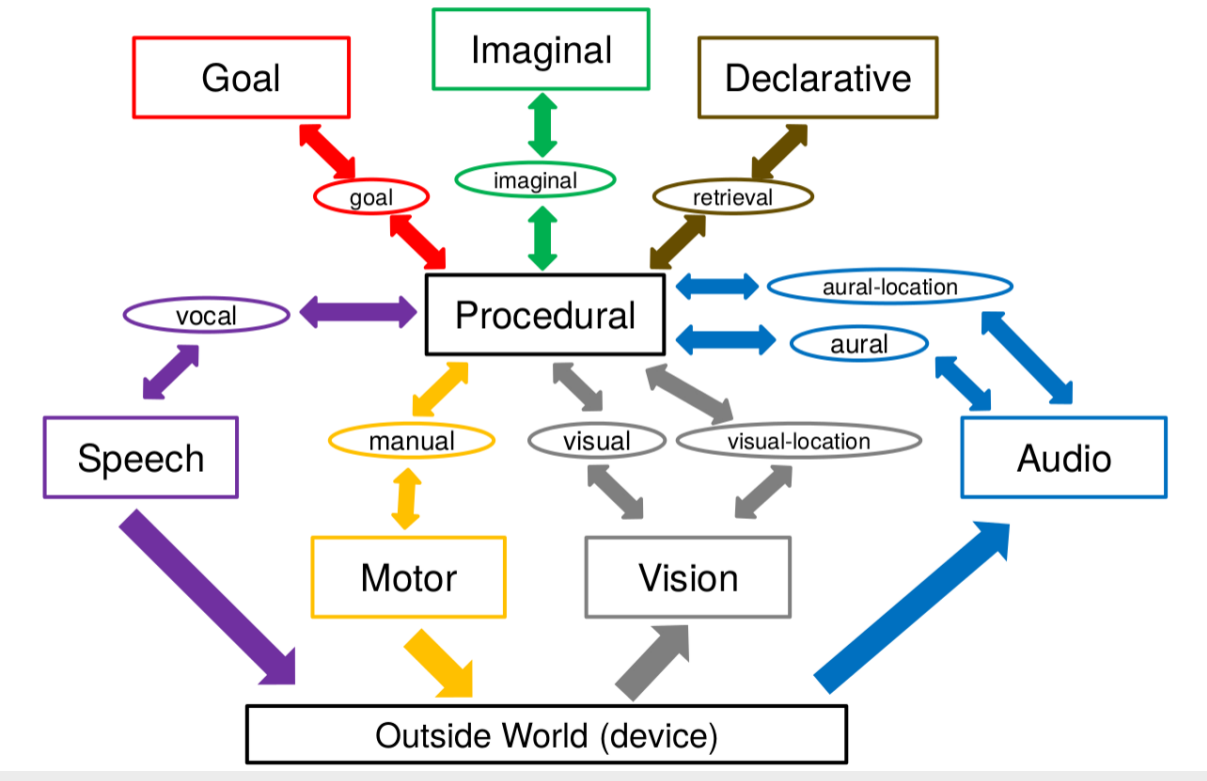
- Chunks can carry a negative value or a variable (such chunks are never part of the declarative memory)

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

Modules and buffers

- ACT-R is modular (declarative module, procedural module...)
- Modules are not directly accessible – they can only be accessed through buffers
- Buffers represent agent's current state; productions fire based on contents of buffers
- Buffers can normally hold only one chunk

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

Procedural knowledge in ACT-R

Left-hand side:

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

Right-hand side:

- Specify a buffer, specify how the current chunk must be modified
- Specify a buffer, specify what chunk must be created

Subsymbolic part – retrieval from declarative

- Power law: $P = At^{-d}$
  (P – performance; t – time; A, d – free params)

$$A = \log\left(\sum_{k=1}^{n} t_k^{-d}\right) + \text{“working memory” activation}$$

- e^A – odds that you will recall a chunk

$$T = Fe^{-B}$$

Numerical quantifiers

– computable by finite-state machines
– computable just by productions
■ There is more than 1 dot.

start: g – [counted: 0 end: 2]

```
=g>
counted     0
end         2
=visual>
value       dot
==>
=g>
counted     1
+visual>
cmd         move
```

```
=g>
counted     1
end         2
=visual>
value       dot
==>
=g>
counted     2
+visual>
cmd         move
```

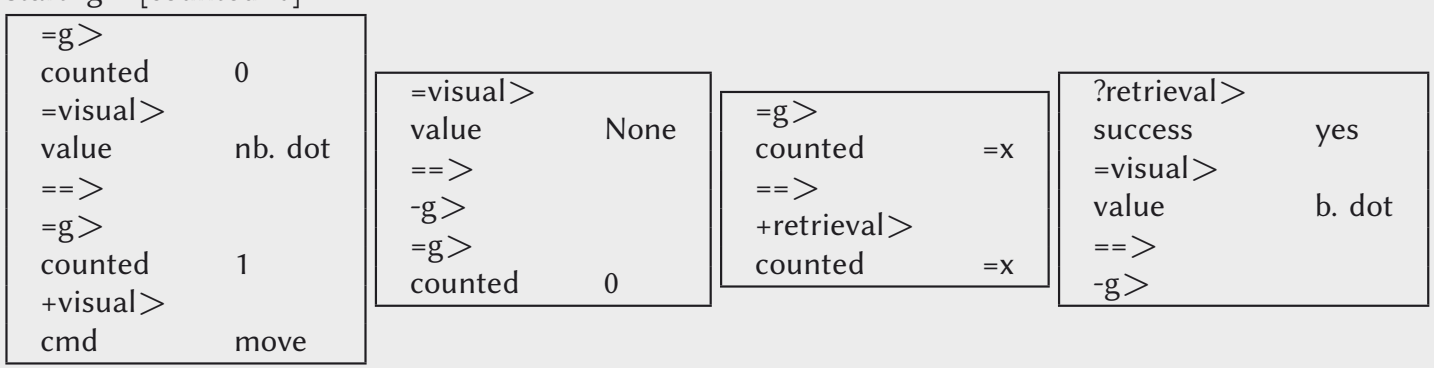
```
=g>
counted     2
end         2
=visual>
value       dot
==>
-g>
```

Proportional quantifiers

– computable by push-down automata
– can be computed by productions + declarative

- Most dots are blue.

start: g – [counted: 0]



Proportional quantifiers

– can be computed just by productions (using variable matching)

- Most dots are blue.

```
=g>
counted      0
=visual>
value        nb. dot
==>
=g>
counted      1
+visual>
cmd          move
```

```
=g>
counted      =x
=visual>
value        None
==>
=g>
counted      0
end          =x
```

```
=g>
counted      0
=visual>
value        b. dot
==>
=g>
counted      1
+visual>
cmd          move
```

```
=g>
counted      =x
end          =x
=visual>
value        b. dot
==>
-g>
```

We'll disregard this option for a while

Summary

- Productions faster and less prone to mistake
- Declarative slower (retrieval needed) and can misfire
- Productions (without variable matching) less powerful than productions + declarative

Hypothesis

- In cases solvable by finite-state machines, use only productions
- In cases solvable by push-down automata, use declarative + productions
- Compatible with implicit assumption in Jakub's work

  Szymanik and Zajenkowski, 2010, Zajenkowski et al. (2011)

ACT-R learning – More nuanced

ACT-R intro

ACT-R – declarative & procedural

Learning counting

Quantification

Conclusion

ACT-R intro

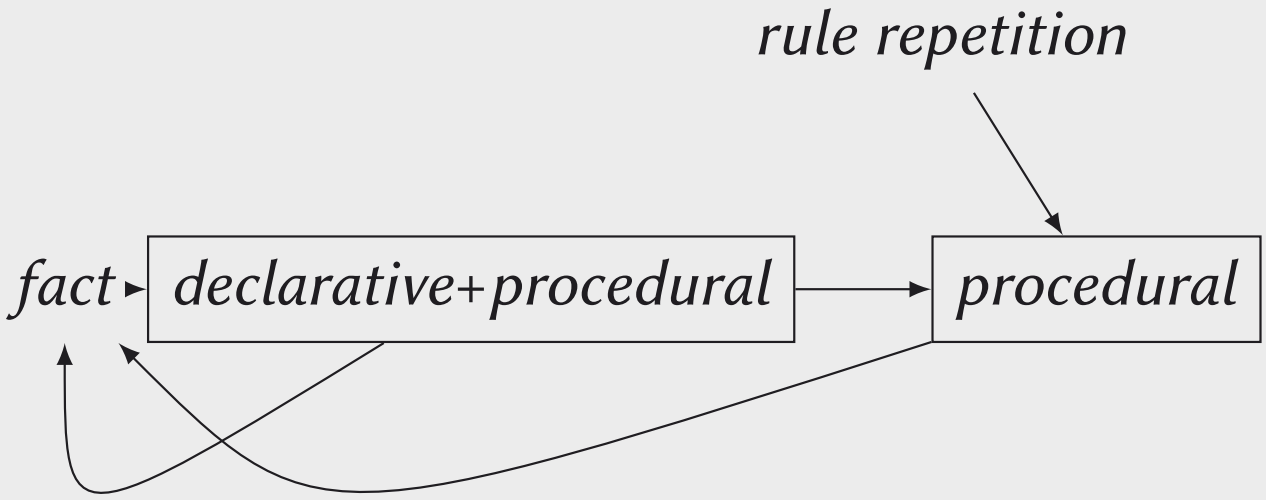
ACT-R – declarative & procedural

Learning counting

Quantification

Conclusion

Learning in ACT-R



Learning counting

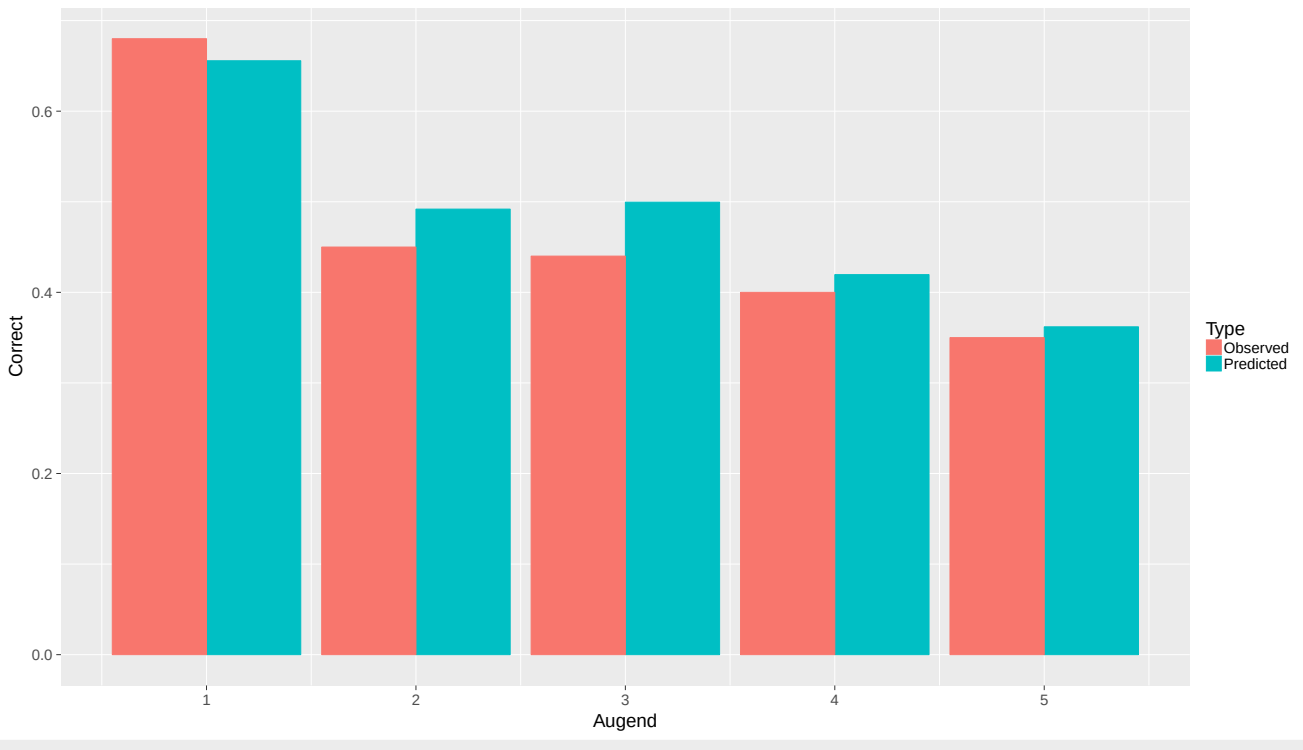
Lebiere (1999), modified

2+3=?

- Step 1: fact: a sequence of numbers
- Step 2: increment 2 by recalling what number follows; repeat three times
- Step 3: increment 2 three times without recalling
- Step 4: new fact: given 2 and 3, the sum is 5
- Step 5: when seeing 2 and 3 recall sum
- Step 6: 2+3=5 (without recalling)

Children's ability of addition

Children's accuracy in recalling, addition up to 10



Learning and testing data from Ashcraft (1992)

ACT-R model based on Lebiere (1998)

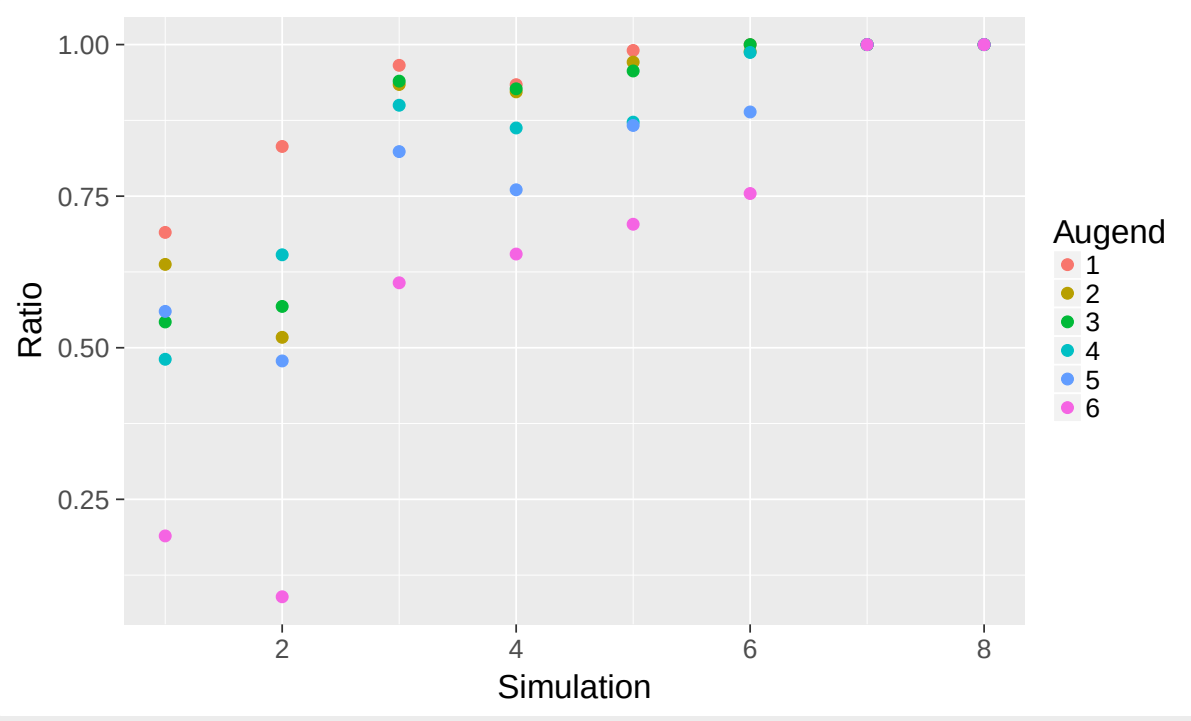
Children's ability of addition

Why match?

- Smaller sums practiced more often and ACT-R sensitive to practice
- Higher numbers matched in activation more closely than smaller numbers (ANS-like effect)

Learning addition

Simulation of learning, addition up to 12

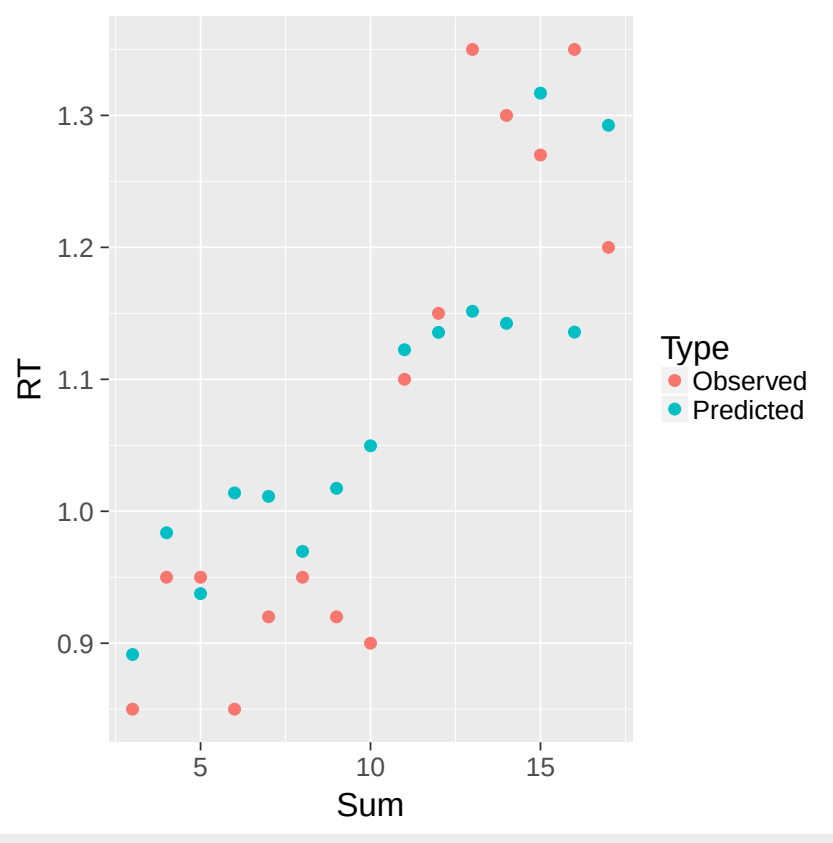


Recalling is strengthened

New rule is learned, independent of activation of chunk

Addition, adults

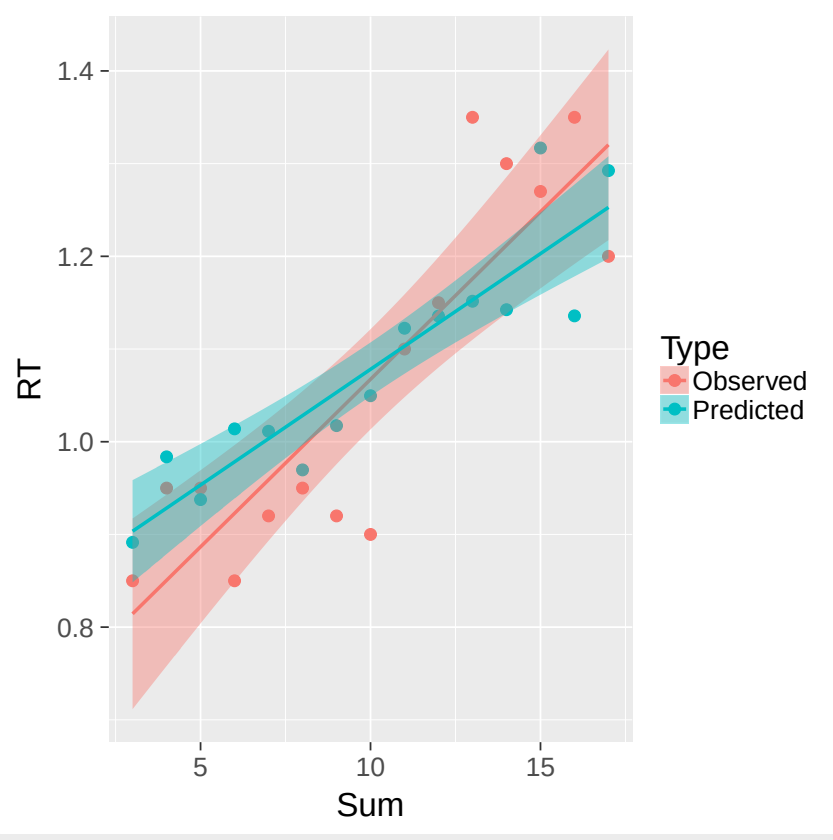
Reaction times, adults (Lebiere, 1998)



Note that RTs increase with higher sums

Addition, adults

Reaction times, adults (Lebiere, 1998)



Addition, adults

- Higher numbers are less practiced
- New rules are less likely created, addition is harder to recall

Learning counting

- The final model has 4 sub-systems:
- increment by recalling a sequence of numbers
- increment by using just productions
- recall the sum
- find the sum by using just productions

ACT-R intro

ACT-R – declarative & procedural

Learning counting

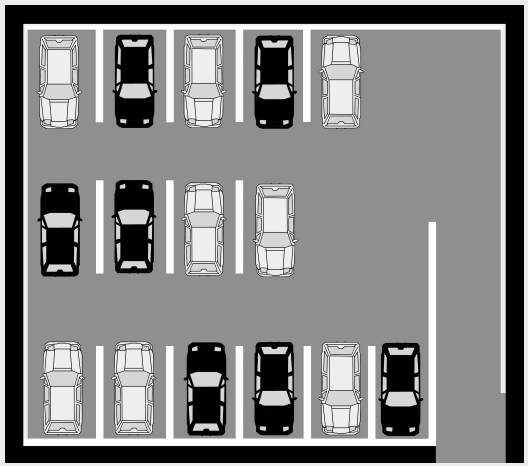
Quantification

Conclusion

Zajenkowski, Styła, Szymanik

- Polish sentences with quantifiers:

  Więcej niż polowa samochodów jest niebieska
  More than half cars is blue



- Verification task

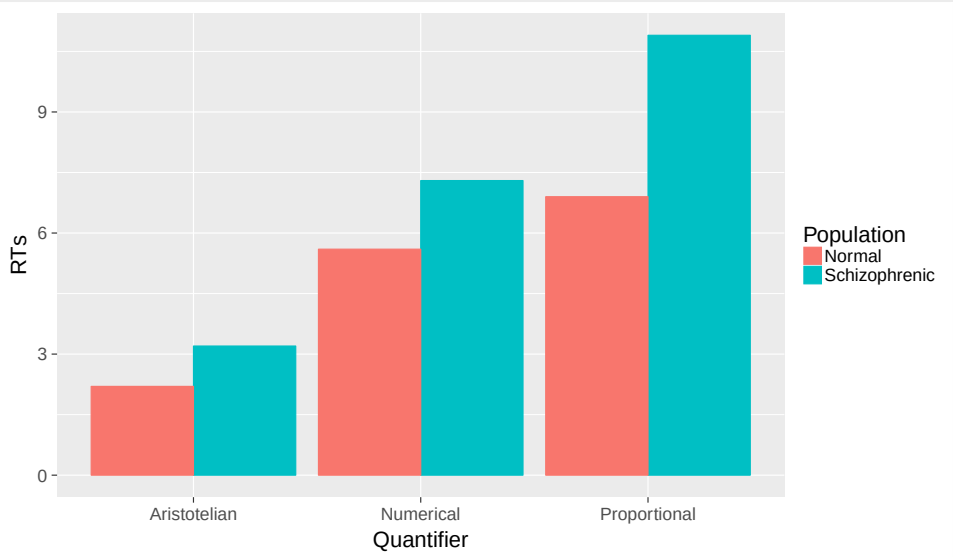
Four quantifier groups tested in normal and schizophrenic population:

1. **PQ:** proportional quantifiers (*less than half, more than half*)
2. **NQ:** numerical quantifiers (*less than 8, more than 7*)
3. **A:** Aristotelian quantifiers (*all/some*)

Zajenkowski, Styła, Szymanik

- Schizophrenic population – decreased working memory

  Lee and Park (2005)
- Hence, (greater) problems with proportional quantifiers



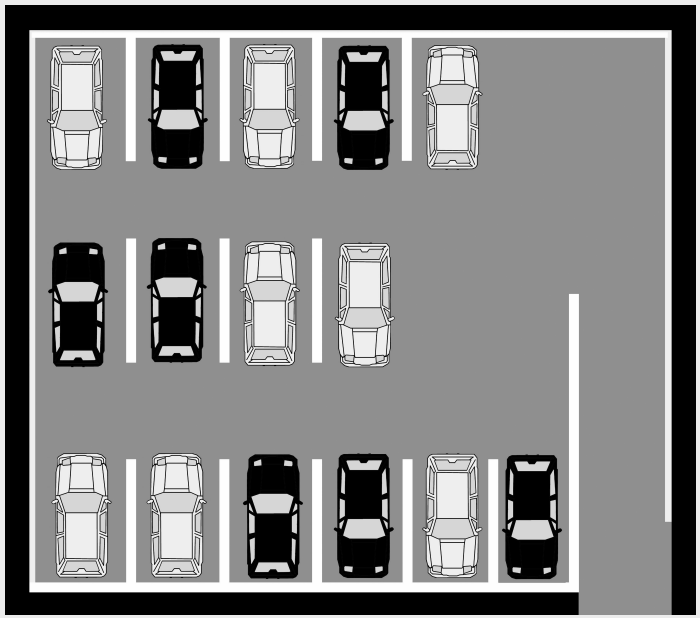
Zajenkowski, Styła, Szymanik

ACT-R model

- Recall that both NQs and PQs would involve procedural *and* declarative knowledge
- However, PQs rely on it more often, and higher sums take more time to retrieve
- This will suffice to account for the difference
- NB. no declarative memory needed for PQs beyond counting

A note on modeling

- We'll take over the model from learning counting (adult version)
- The model is expanded with vision, key pressing



A note on modeling

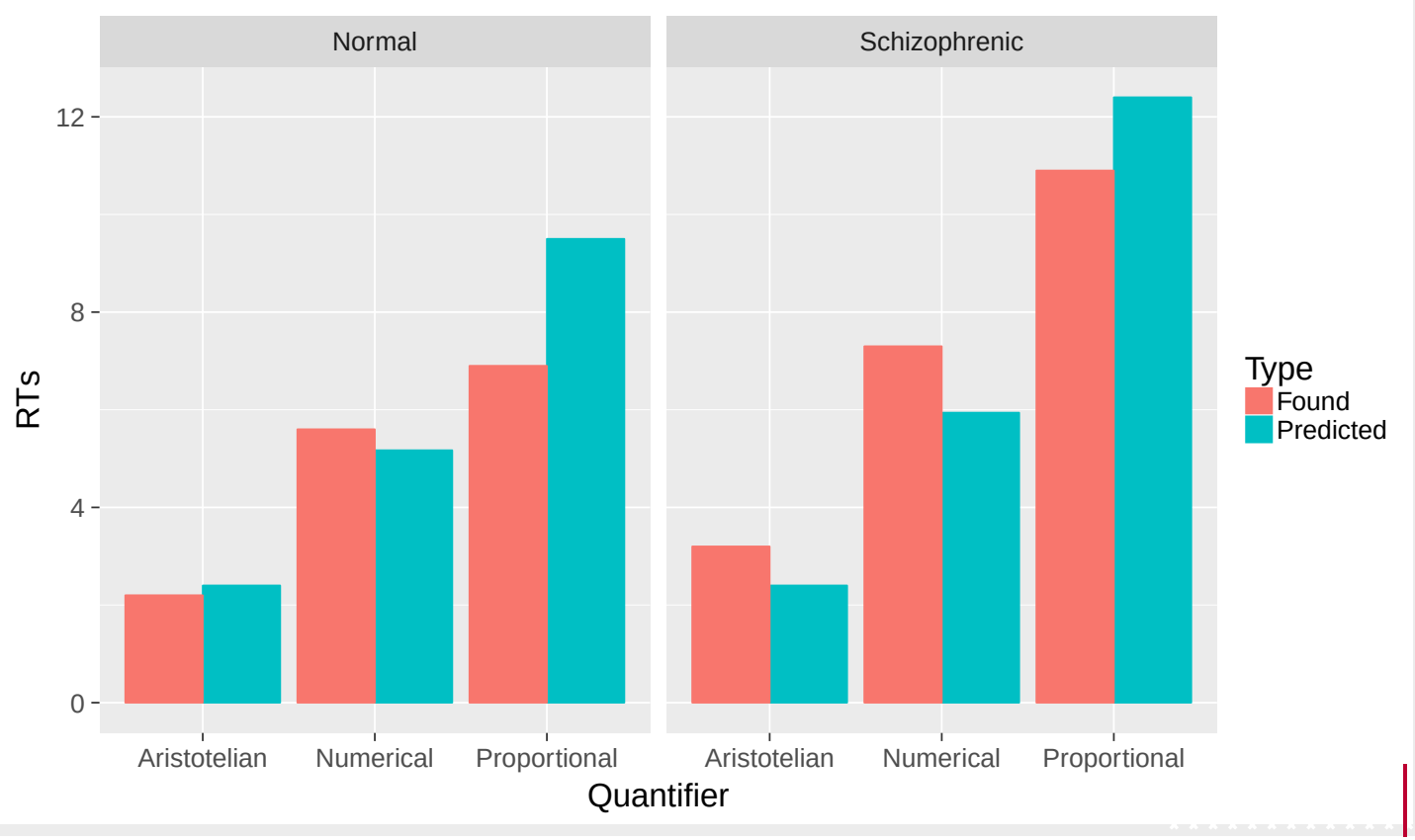
- We'll take over the model from learning counting (adult version)
- The model is expanded with vision, key pressing
- Schizophrenic population – decreased working memory

  Lee and Park (2005)
- We'll model this by assuming that schizophrenics do not have spreading activation from "working memory"

  van Rij et al. (2012)

ACT-R model for quantification

Data and simulation



Discussion

- We capture the interaction of population $\times$ quantifiers
- However, we overpredict time needed to deal with proportional quantifiers
- Possible reasons:
  - more incentive to speed up counting when many objects
  - people do not count all the cars, they count blue and non-blue cars

Conclusion

- Adding model on counting strengthens predictions for verification of quantifiers
- Difference between finite-state and push-down automata meaningful in ACT-R
- At least some behavioral data accounted for even though both numerical and proportional quantifiers are verified using productions + declarative

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

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.

Ashcraft, Mark H. 1992. Cognitive arithmetic: A review of data and theory. *Cognition* 44:75–106.

Lebiere, Christian. 1999. The dynamics of cognition: An act-r model of cognitive arithmetic. *Kognitionswissenschaft* 8:5–19.

Pollard, Carl, and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

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

Zajenkowski, Marcin, Rafał Styła, and Jakub Szymanik. 2011. A computational approach to quantifiers as an explanation for some language impairments in schizophrenia. *Journal of communication disorders* 44:595–600.