EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Probabilistic program induction of symbols

Or: More complex models!

- *The Language of Thought: computational cognitive science approaches to category learning*
- Who: Fausto Carcassi
- When: Sommer semester 2022

Where are we?

- Last week, we have seen two applications of the idea of pLoT to new conceptual domains: handwritten digits and kinship systems.
- This week, we'll look at a few more applications.
- It'll be a bit of a whirlwind!
- We will look at:
  - Learning numerals in an LoT
  - Learning abstract visual concepts in an LoT
  - Learning sequences in an LoT

Learning Numerals with an LoT

- Piantadosi et al (2012), *Bootstrapping in a language of thought: A formal model of numerical concept learning*.
- Children exhibit very regular patterns in the way they learn number systems.
- The first learn to recognize small sets of size 1, then 2, then 3, etc.
    - In Carey's formulation, early number-word meanings are represented using mental models of small sets. For instance two-knowers might have a mental model of "one" as {X} and "two" as {X, X}. These representations rely on children's ability for enriched parallel individuation, a representational capacity that Le Corre and Carey (2007) argue can individuate objects, manipulate sets, and compare sets using one-to-one correspondence. Subset-knowers can, for instance, check if "two" applies to a set S by seeing if S can be put in one-to-one correspondence with their mental model of two, {X, X}
- Then at about 3;6 they learn the full recursive system (Cardinal Principal learners)
- This is a qualitative jump rather than continuous smooth progress.

Learning Numerals with an LoT

- Let's see if we can reproduce this qualitative conceptual jump with an LoT!
- We have at least three choices for how to set up the LoT. Each sentence in the LoT could be:
  - A function from a number word to a predicate of sets
  - A function from a set to a number word
  - A function that constructs a set from the objects in the situation
- Children can do all these three things, and there is no clear empirical evidence one way or the other.
- In the paper, they go the second way: from a set to a number word.

Learning Numerals with an LoT

- Rules in the LoT:

Functions mapping sets to truth values	
(*singleton? X*)	Returns true iff the set *X* has exactly one element
(*doubleton? X*)	Returns true iff the set *X* has exactly two elements
(*tripleton? X*)	Returns true iff the set *X* has exactly three elements
Functions on sets	
(*set-difference X Y*)	Returns the set that results from removing *Y* from *X*
(*union X Y*)	Returns the union of sets *X* and *Y*
(*intersection X Y*)	Returns the intersect of sets *X* and *Y*
(*select X*)	Returns a set containing a single element from *X*
Logical functions	
(*and P Q*)	Returns TRUE if *P and Q* are both true
(*or P Q*)	Returns TRUE if either *P or Q* is true
(*not P*)	Returns TRUE iff *P* is false
(*if P X Y*)	Returns *X* iff *P* is true, *Y* otherwise
Functions on the counting routine	
(*next W*)	Returns the word after *W* in the counting routine
(*prev W*)	Returns the word before *W* in the counting routine
(*equal-word? W V*)	Returns TRUE if *W* and *V* are the same word
Recursion	
(*L S*)	Returns the result of evaluating the entire current lambda expression on set *S*

Learning Numerals with an LoT

- Recursion is the only one that's a bit complicated. What do you think the following does?

```
λ S · (if (singleton? S)
        "one"
        (next (L (select S)))).
```

One-knower

```
λ S . (if (singleton? S)
          "one"
          undef)
```

Two-knower

```
λ S . (if (singleton? S)
          "one"
          (if (doubleton? S)
              "two"
              undef))
```

Three-knower

```
λ S . (if (singleton? S)
          "one"
          (if (doubleton? S)
              "two"
              (if (tripleton? S)
                  "three"
                  undef))
```

CP-knower

```
λ S . (if (singleton? S)
          "one"
          (next (L (set-difference S
                                   (select S)))))
```

Singular-Plural

```
λ S . (if (singleton? S)
          "one"
          "two")
```

Mod-5

```
λ S . (if (or (singleton? S)
              (equal-word? (L (set-difference S)
                                   (select S))
                           "five"))
          "one"
          (next (L (set-difference S
                                   (select S)))))
```

2-not-1-knower

```
λ S . (if (doubleton? S)
          "two"
          undef)
```

2N-knower

```
λ S . (if (singleton? S)
          "one"
          (next (next (L (set-difference S (select S))))))
```

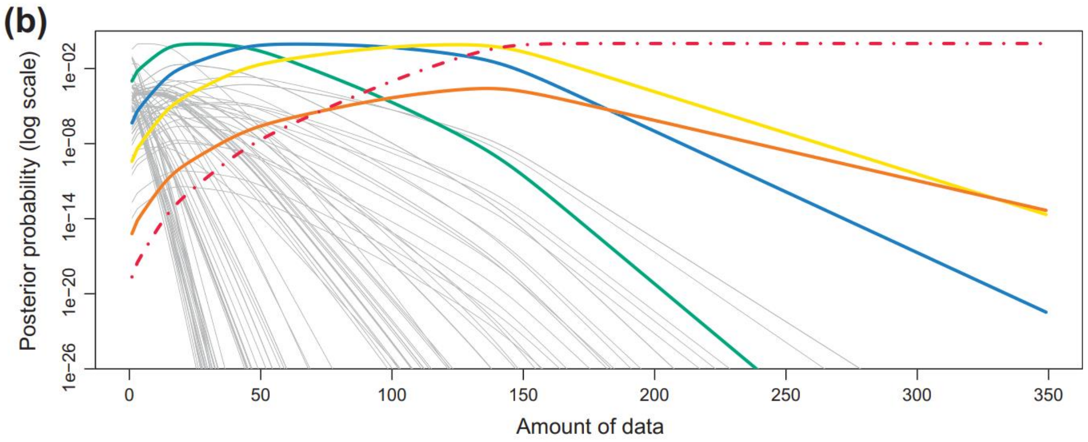
Learning Numerals with an LoT

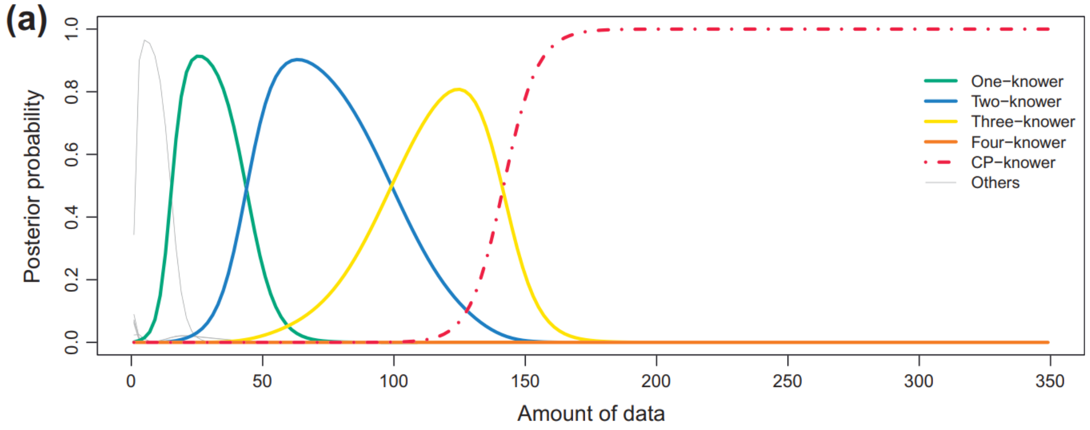
- The likelihood function is pretty typical:
  - First, a set of objects is chosen from the universe of object, e.g. 'cats'
  - Second, the hypothesis is evaluated on the set.
  - This can result either in a number word or 'undef'
  - If the result is 'undef', a random number word is produced
  - If the result is a number word, the word is produced with probability α and with $1 - \alpha$ a random word is picked.
- Resulting likelihood function:

$$P(w_i|t_i, c_i, L) = \begin{cases} \frac{1}{N} & \text{if } L \text{ yields } \textit{undef} \\ \alpha + (1-\alpha)\frac{1}{N} & \text{if } L \text{ yields } w_i \\ (1-\alpha)\frac{1}{N} & \text{if } L \text{ does not yield } w_i \end{cases}$$

Learning Numerals with an LoT

- The model also penalizes recursive functions with a parameter γ
- Results look strikingly like human learning patterns:

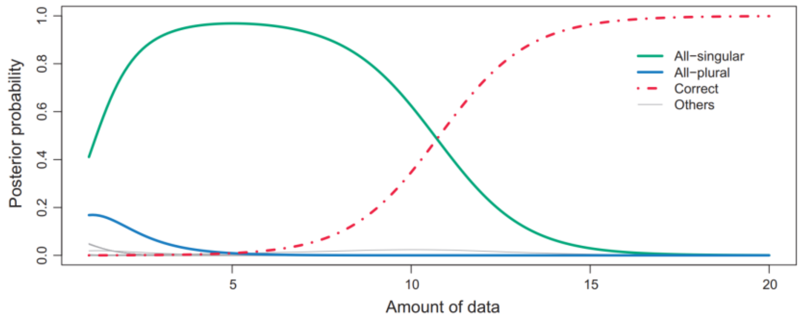




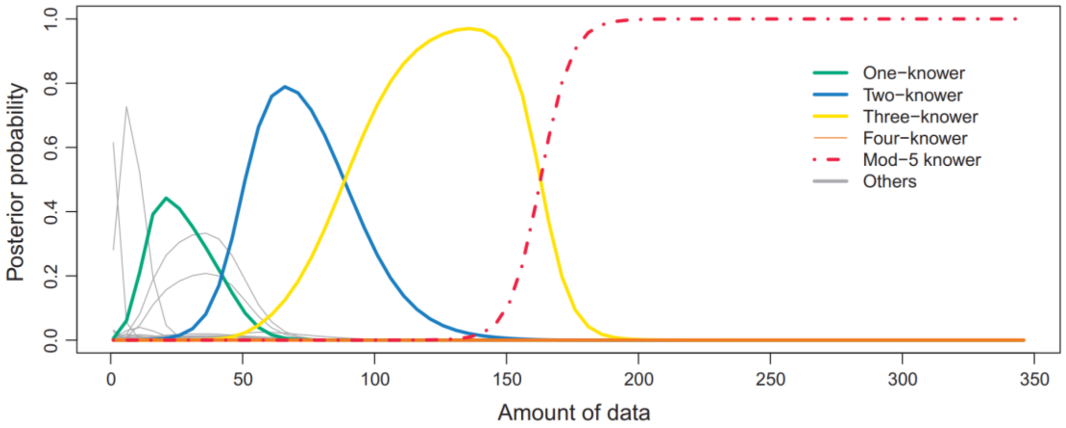
- In particular, note all the hypotheses that are considered and then disregarded!

Learning Numerals with an LoT

- The same LoT can also learn things like singular/plural morphology:

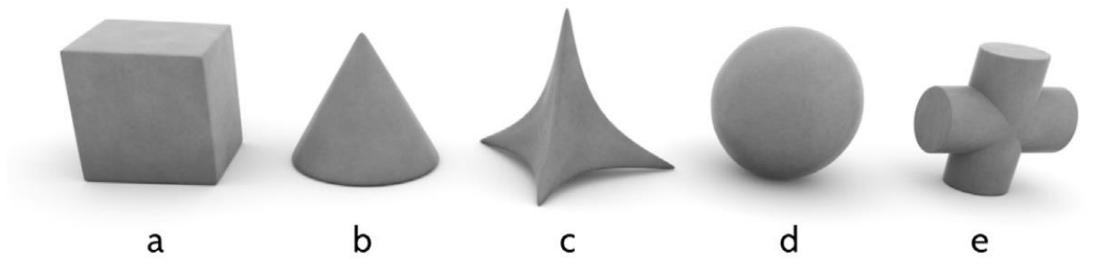


- And mod-n systems (e.g. days of the week)



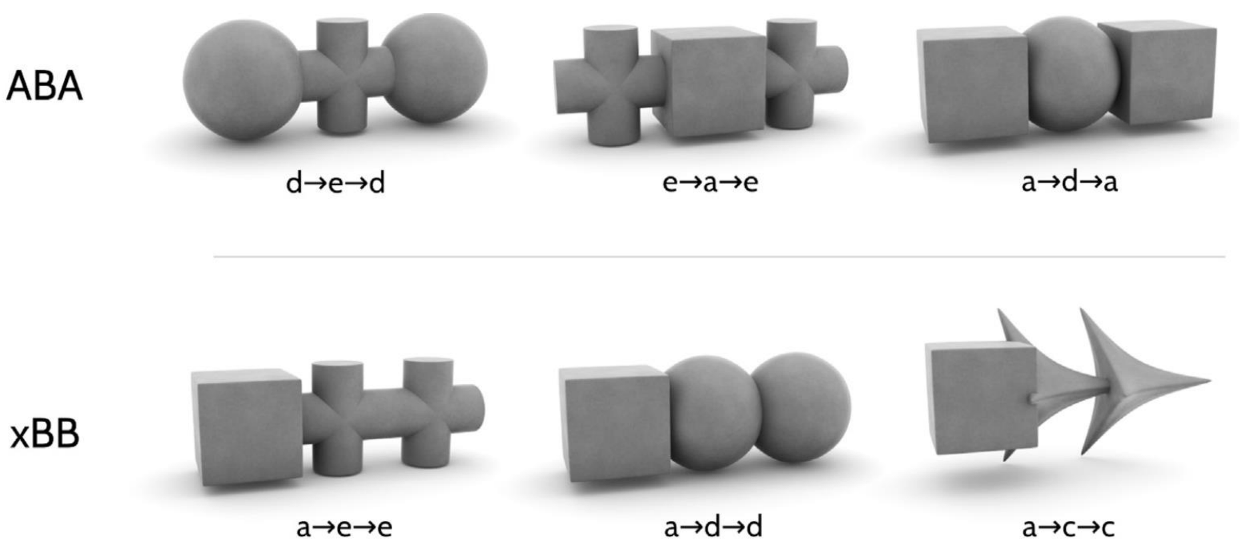
Visual Concept Learning in an LoT

- Overlan et al (2017), *Learning abstract visual concepts via probabilistic program induction in a Language of Thought*
- Last week, we have seen a model that can learn and do various other things with handwritten characters.
- Now let's see if we can work with something else in the visual modality, namely the structure of 3-d objects.
- Suppose we have the following primitive objects, and we can combine them in various ways:



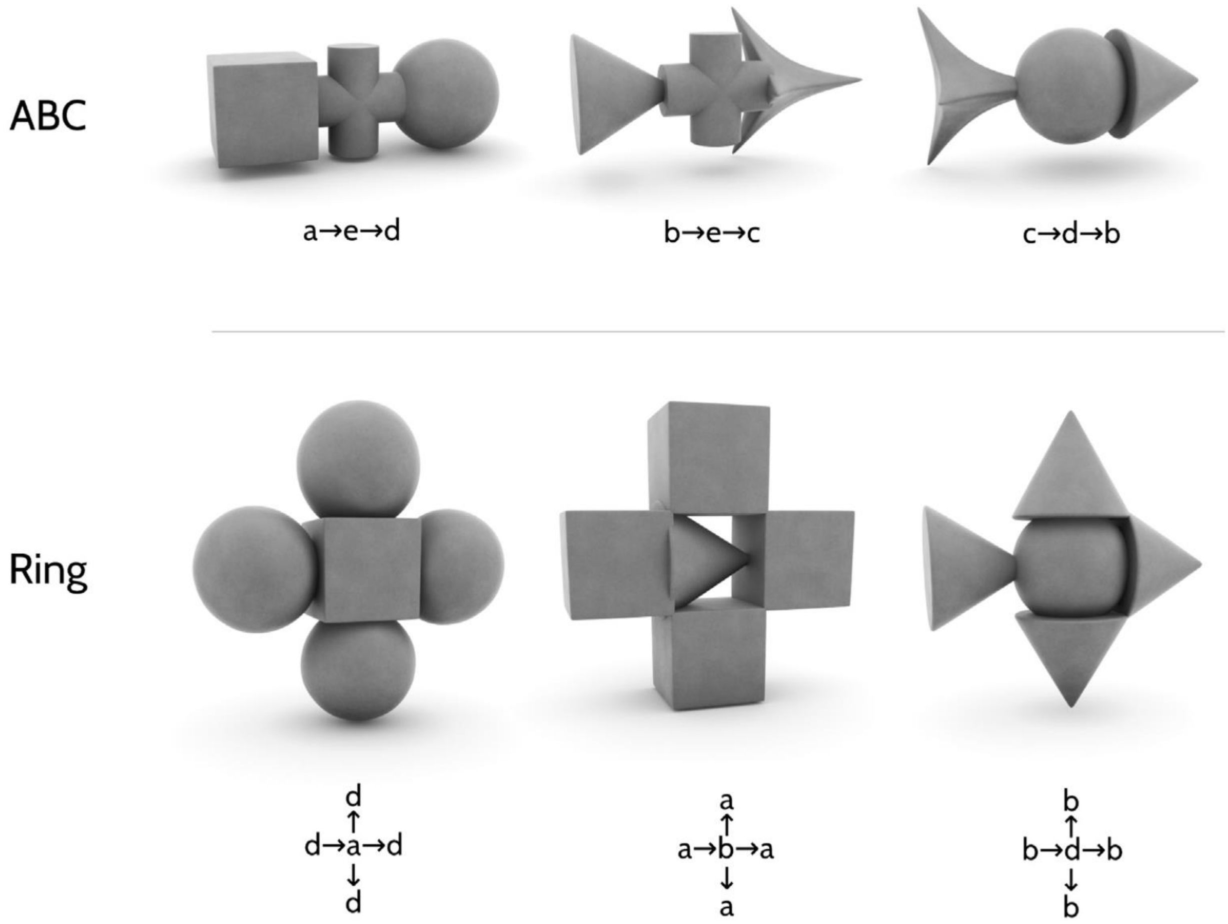
Visual Concept Learning in an LoT

- For instance, we can produce the following categories:



- Note that each category identifies a *class* of objects!

Visual Concept Learning in an LoT



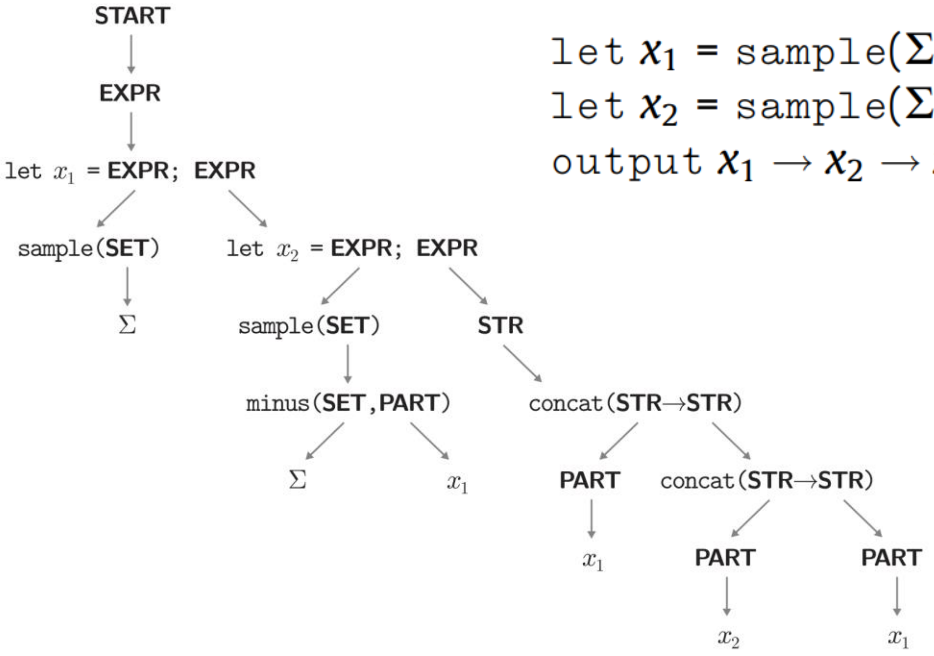
Visual Concept Learning in an LoT

- What could a grammar for this look like?

$$
\begin{array}{rll}
\text{START} & \rightarrow \texttt{let}\ \langle\text{BV\_PART}\rangle{:}x_1 = \text{FIRST\_PART}; \text{EXPR} & \\
\text{EXPR} & \rightarrow \texttt{let}\ \langle\text{BV\_PART}\rangle{:}x_n = \text{PART}; \text{EXPR} & \\
 & \rightarrow \text{STRING} & \\
\text{STRING} & \rightarrow \text{BV\_PART} & \\
 & \rightarrow \text{STRING CONNECT STRING} & \\
 & \rightarrow \{\text{STRING}\} & \\
\text{FIRST\_PART} & \rightarrow \texttt{sample(}\text{FIRST\_SET}\texttt{)} & 1 - p_{single} \\
 & \rightarrow \text{SINGLE} & p_{single} \\
\text{PART} & \rightarrow \text{BV\_PART} & (1 - p_{single})/2 \\
 & \rightarrow \texttt{sample(}\text{SET}\texttt{)} & (1 - p_{single})/2 \\
 & \rightarrow \text{SINGLE} & p_{single} \\
\text{FIRST\_SET} & \rightarrow \Sigma & 1 - p_{minus} \\
 & \rightarrow \texttt{minus(}\text{FIRST\_SET}\texttt{,}\ \text{FIRST\_PART}\texttt{)} & p_{minus} \\
\text{SET} & \rightarrow \Sigma & 1 - p_{minus} \\
 & \rightarrow \texttt{minus(}\text{SET}\texttt{,}\ \text{BV\_PART}\texttt{)} & p_{minus} \\
\text{CONNECT} & \rightarrow \text{`}\uparrow\text{'} \mid \text{`}\downarrow\text{'} \mid \text{`}\leftarrow\text{'} \mid \text{`}\rightarrow\text{'} & \\
\text{SINGLE} & \rightarrow \text{`a'} \mid \ldots \mid \text{`e'} &
\end{array}
$$

Visual Concept Learning in an LoT

- Example of a derivation and some categories:



let x_1 = sample(Σ)
let x_2 = sample($\Sigma - x_1$)
output $x_1 \to x_2 \to x_1$

ABA

let x_1 = sample(Σ_R)
let x_2 = sample($\Sigma_R - x_1$)
output $x_1 \to x_2 \to x_1$

xBB

let x_1 = sample(Σ)
let x_2 = 'a'
output $x_2 \to x_1 \to x_1$

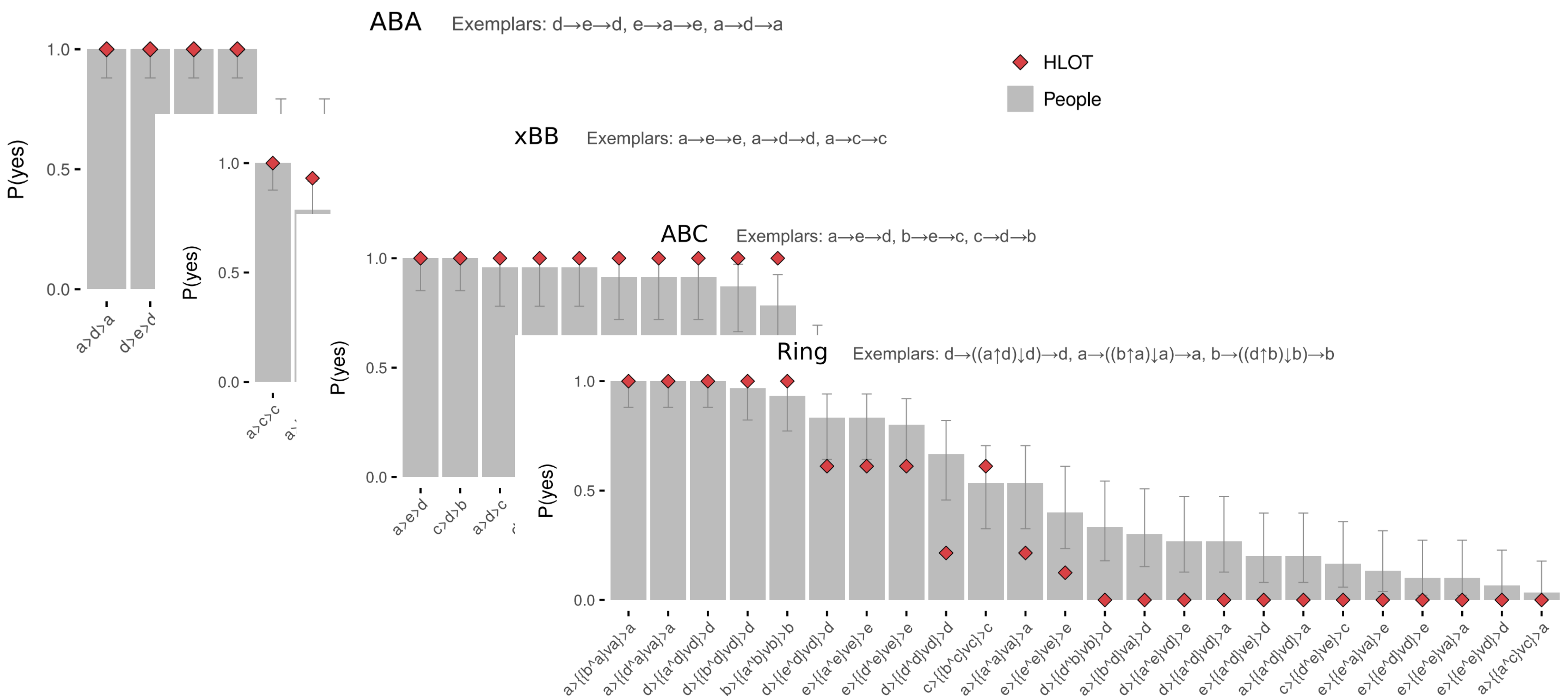
ABC

let x_1 = sample(Σ)
let x_2 = sample($\Sigma - x_1$)
let x_3 = sample($\Sigma - x_2 - x_1$)
output $x_2 \to x_3 \to x_1$

Ring

let x_1 = sample(Σ_R)
let x_2 = sample($\Sigma_R - x_1$)
output $x_1 \to ((x_2 \uparrow x_1) \downarrow x_1) \to x_1$

Visual Concept Learning in an LoT

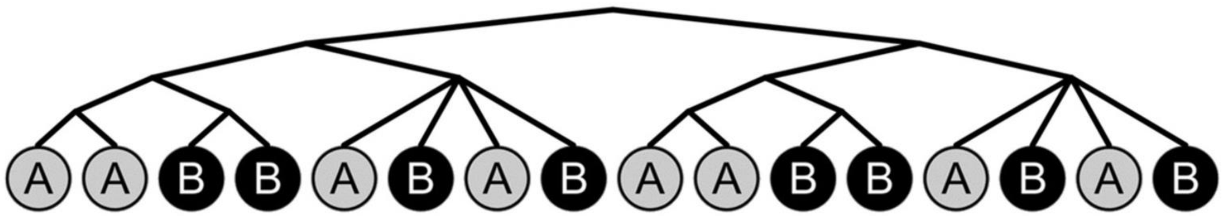


Sequence Learning in an LoT

- Planton et al (2021), *A theory of memory for binary sequences: Evidence for a mental compression algorithm in humans*
- The scope here is to understand how humans deal with *binary sequences*, i.e. sequences composed of only two elements.
    - For instance, {0,1}*
    - But not that it doesn't need to be symbols. E.g. it can be two pitches.
- Much literature has been devoted to understand how humans learn these.
- Usually, the general strategy is to find a mechanism where strings can be encoded, which explains how long strings can be learned, which would be impossible with pure memorization.

Sequence Learning in an LoT

- In this case, we'll use an LoT where each sentence can encode a binary series. For instance:



LoT program expression :

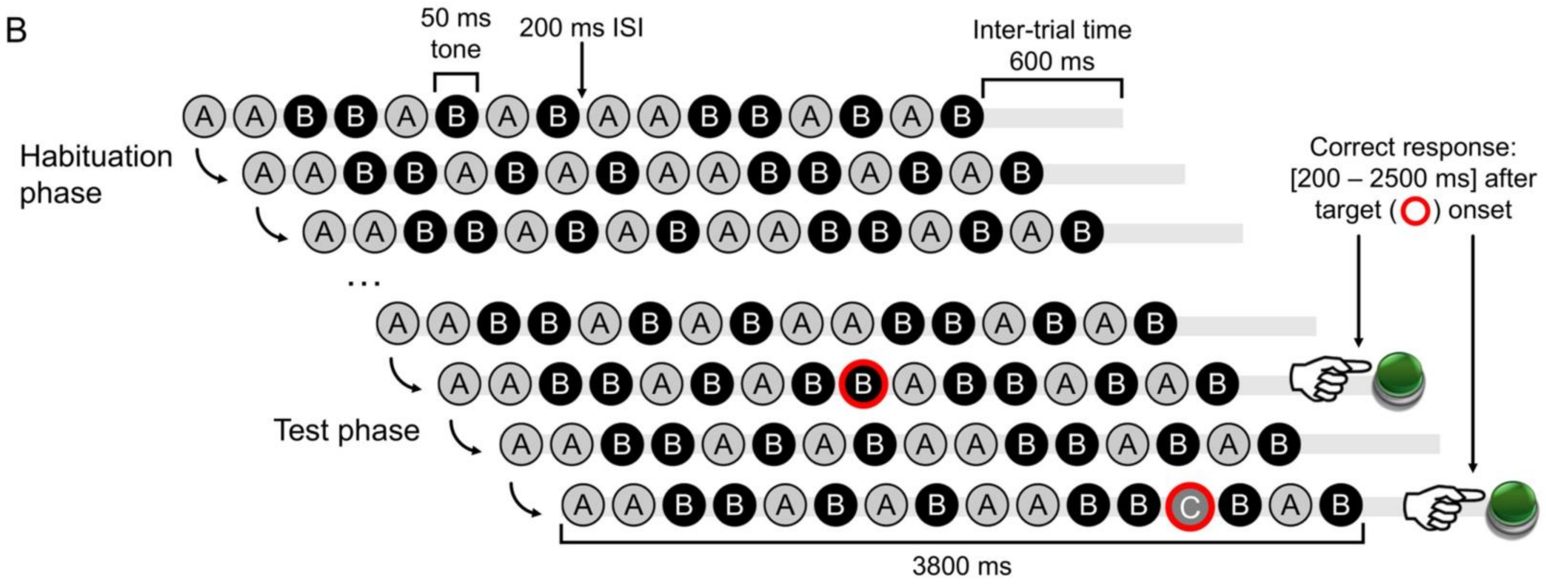
```
[[[+0]^2]^2<b>,[b]^4]^2<+0>
```

LoT complexity = 12

- LoT:
  - Staying ("+0")
  - Moving to the other item (here denoted 'b')
  - Repetition ("^n", where n is any number),
    - Possibly with a variation in the starting point
    - Denoted by <x> where x is an elementary instruction, either +0 or b
  - Embedding of expressions is represented by brackets ("[. . ..]") and concatenation by commas (",").

Sequence Learning in an LoT

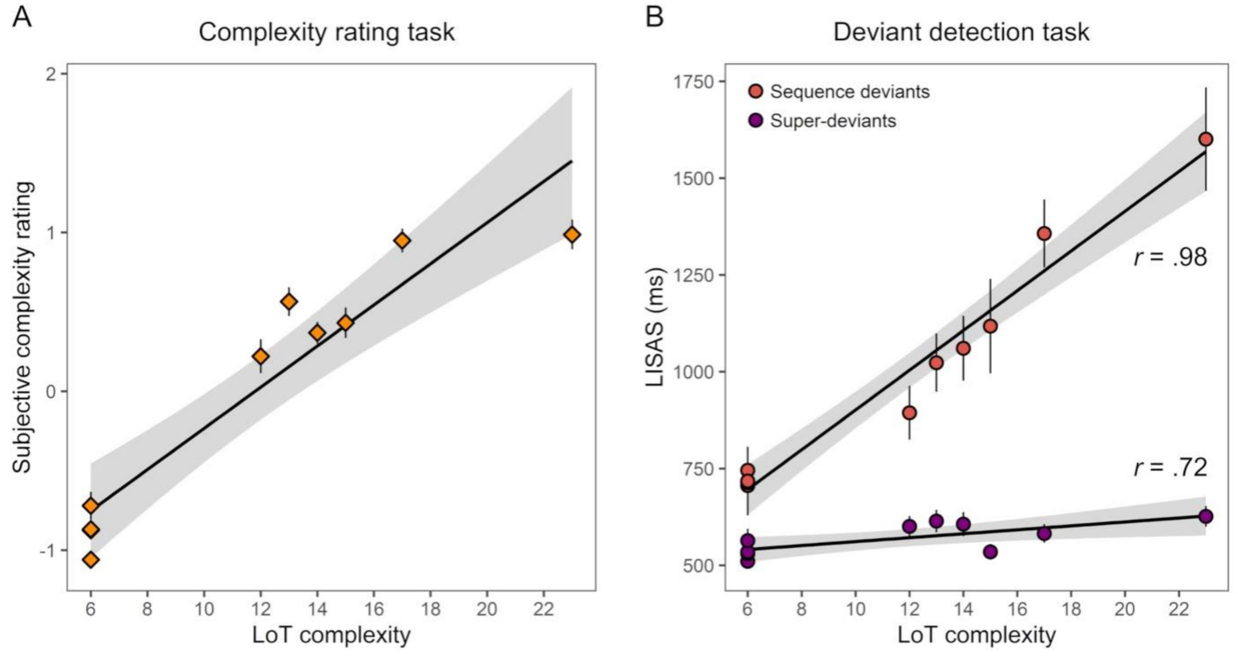
- ‘Sequence violation’ experimental paradigm:



- The basic hypothesis was that, for equal sequence length, error rate and response time in violation detection would increase with sequence complexity.

Sequence Learning in an LoT

- This is the results of the first experiment
- Participants were asked subjective complexity (left)
- And to identify deviations in:
- Sequence deviants
  - A note is replaces with the other note
- Superdeviant
  - A new note is introduced
- LISAS: Linear Integrated Speed-Accuracy Score



Summary

- This week, we saw some other applications of the pLoT idea
- First, we saw a model of numeral acquisition
- Then, a model of learning visual concepts (categories of 3-d objects)
  - This could easily be extended, if you're into 3-d rendering
  - Possible final project?
- Finally, we saw a model of learning binary sequences of tones
- In the lab this week, we'll try to implement a model from scratch
  - I am not going to prepare beforehand so we can all write it together
- Next week is the last week, so the lecture is going to be a review of what we've done, with some concluding remarks on the general project.
- In the lab next week we'll implement another model.