Concept Learning as Coarse-to-Fine Probabilistic Program Induction

Maddy Bowers*,[1], Alexander Lew*,[1], Wenhao Qi[2], Vikash Mansinghka[1], Joshua Rule[3], Joshua Tenenbaum[1], and Armando Solar-Lezama[1]

[1]MIT
[2]UCSD
[3]UC Berkeley

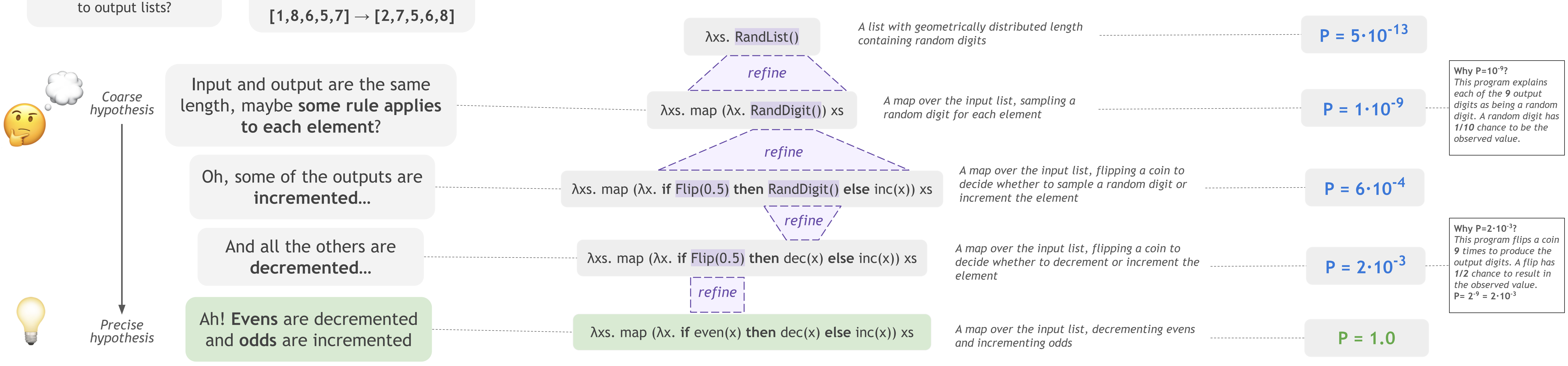
How do we refine our initial "coarse" hypotheses into increasingly precise ones?

We model "coarse" hypotheses as *probabilistic programs*, which are *refined* to gradually introduce deterministic structure

Task: What is the rule that transforms the input lists to output lists?

Input	Output
[3,2,7,6] →	[4,1,8,5]
[1,8,6,5,7] →	[2,7,5,6,8]

The likelihood of the examples under the program is a measure of hypothesis quality



Coarse hypothesis

Precise hypothesis

Input and output are the same length, maybe **some rule applies** to each element?

Oh, some of the outputs are **incremented…**

And all the others are **decremented…**

Ah! **Evens** are decremented and **odds** are incremented

λxs. RandList() — *A list with geometrically distributed length containing random digits* — P = $5\cdot10^{-13}$

refine

λxs. map (λx. RandDigit()) xs — *A map over the input list, sampling a random digit for each element* — P = $1\cdot10^{-9}$

refine

λxs. map (λx. **if** Flip(0.5) **then** RandDigit() **else** inc(x)) xs — *A map over the input list, flipping a coin to decide whether to sample a random digit or increment the element* — P = $6\cdot10^{-4}$

refine

λxs. map (λx. **if** Flip(0.5) **then** dec(x) **else** inc(x)) xs — *A map over the input list, flipping a coin to decide whether to decrement or increment the element* — P = $2\cdot10^{-3}$

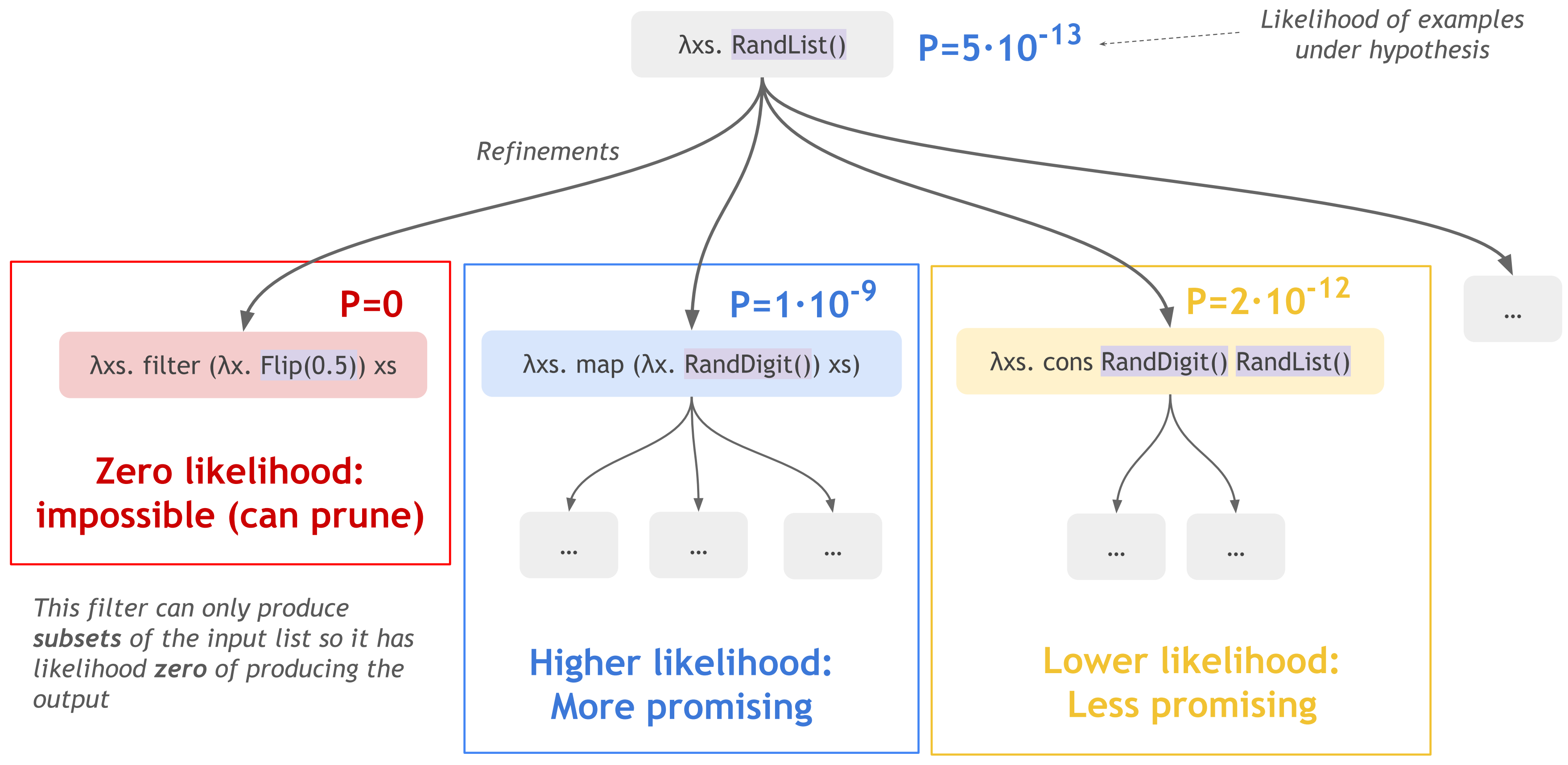
refine

λxs. map (λx. **if** even(x) **then** dec(x) **else** inc(x)) xs — *A map over the input list, decrementing evens and incrementing odds* — P = 1.0

Why P=10^{-9}?
This program explains each of the **9** *output digits as being a random digit. A random digit has* **1/10** *chance to be the observed value.*

Why P=$2\cdot10^{-3}$?
This program flips a coin **9** *times to produce the output digits. A flip has* **1/2** *chance to result in the observed value.*
P= $2^{-9} = 2\cdot10^{-3}$

SMC Search: Pruning & Guiding



λxs. RandList() P=$5\cdot10^{-13}$ *Likelihood of examples under hypothesis*

Refinements

P=0
λxs. filter (λx. Flip(0.5)) xs
Zero likelihood: impossible (can prune)

This filter can only produce ***subsets*** *of the input list so it has likelihood* ***zero*** *of producing the output*

P=$1\cdot10^{-9}$
λxs. map (λx. RandDigit()) xs)
Higher likelihood: More promising

P=$2\cdot10^{-12}$
λxs. cons RandDigit() RandList()
Lower likelihood: Less promising

Search details. We perform a sequential monte carlo (SMC) search over the space of probabilistic programs. At each step, we refine a hypothesis by replacing some stochastic leaf node with a depth 1 node sampled from PCFG grammar.

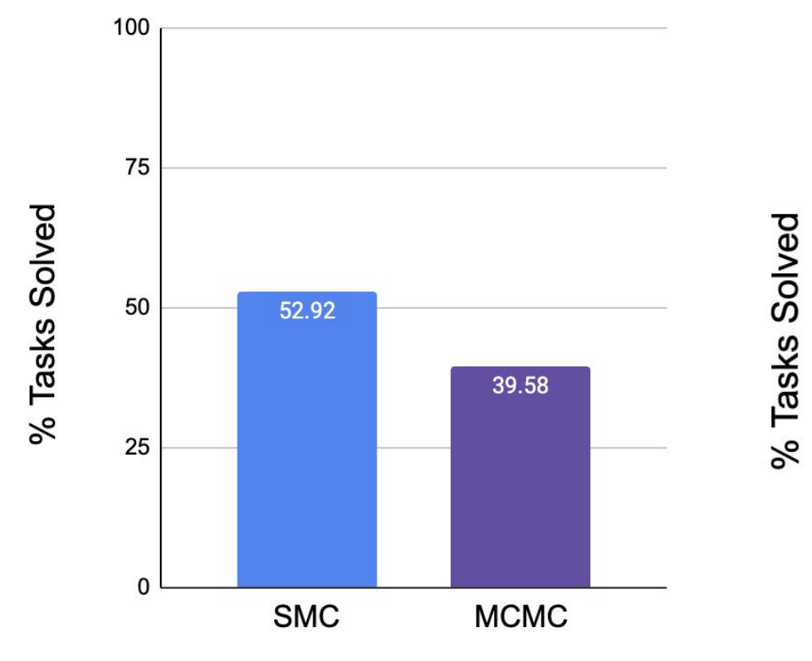
Why can we prune likelihood-zero programs? If a probabilistic program can *never* produce the observed outputs, refining randomness into determinism in it will never result in a program that can produce the outputs.

Why is a high likelihood program promising? A probabilistic program is an *explanation* of how the data came to be. High likelihood programs have already explained much of the data through their *deterministic structure*, minimizing the amount that is explained as *random chance*.

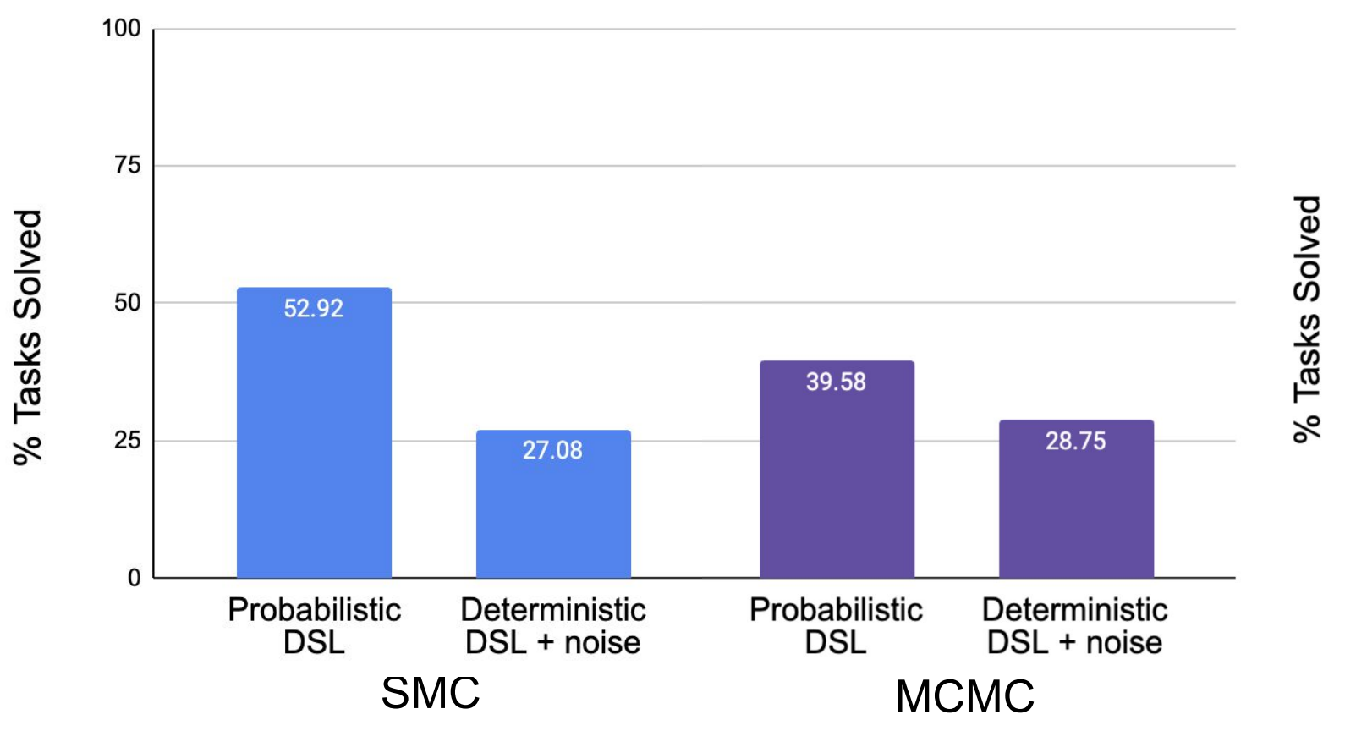
Preliminary Computational Results

- Dataset: First 80 tasks from list manipulation dataset of [Rule 2020; Rule et al 2024]
- SMC: 300 particles for 12 steps; MCMC: 7000 steps (similar amount of time)
- SMC moves: expand a random leaf to a depth one expression.
- MCMC moves: resample random subtree, similar to [Goodman et al. 2008]
- SMC outperforms MCMC, and in particular the probabilistic DSL benefits SMC greatly over using a deterministic DSL that just has add-remove-modify noise on the output list.
- Working in a high level DSL (with *map*, *filter*, etc) is important to coarse-to-fine SMC working well

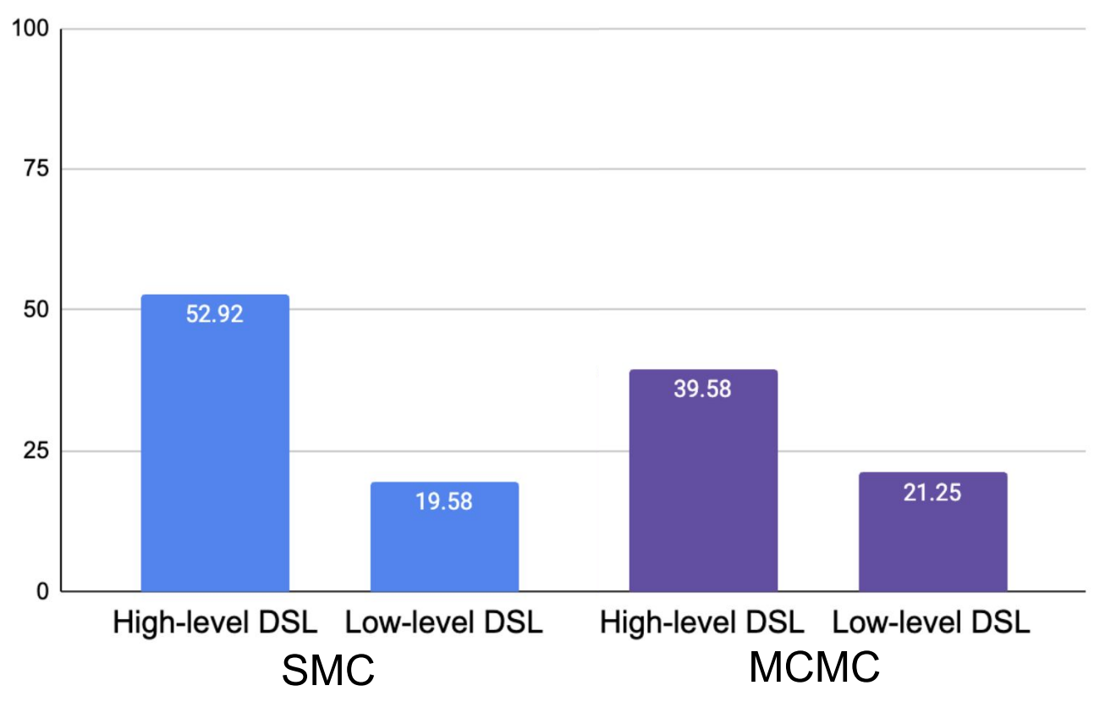
1. SMC outperforms MCMC

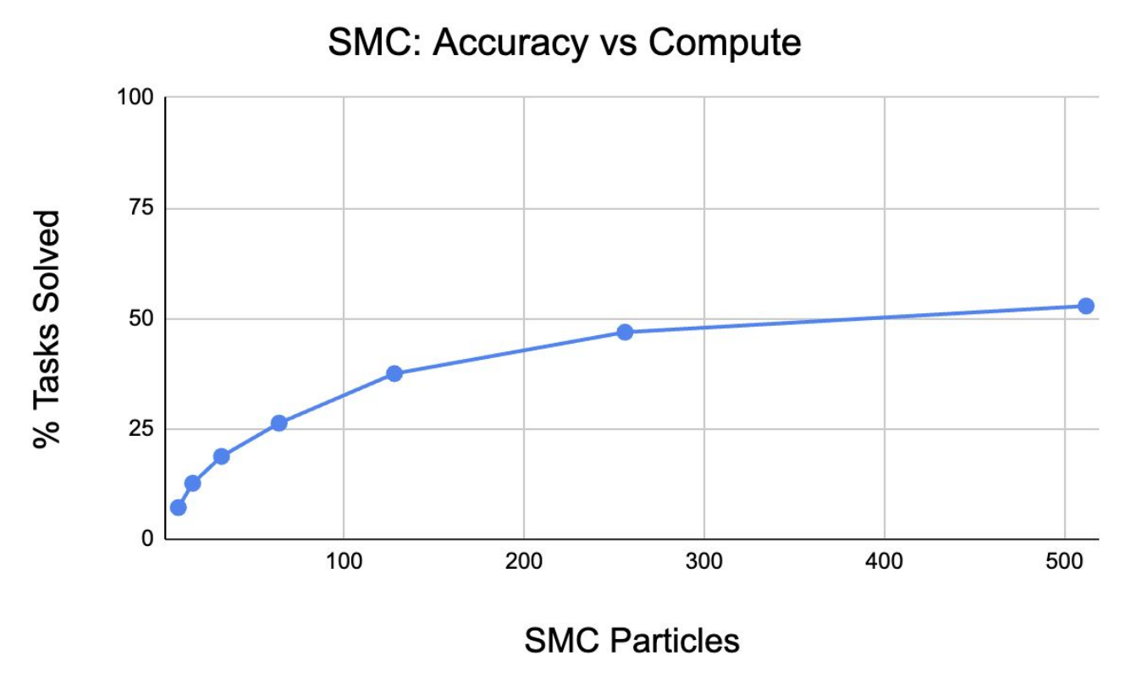


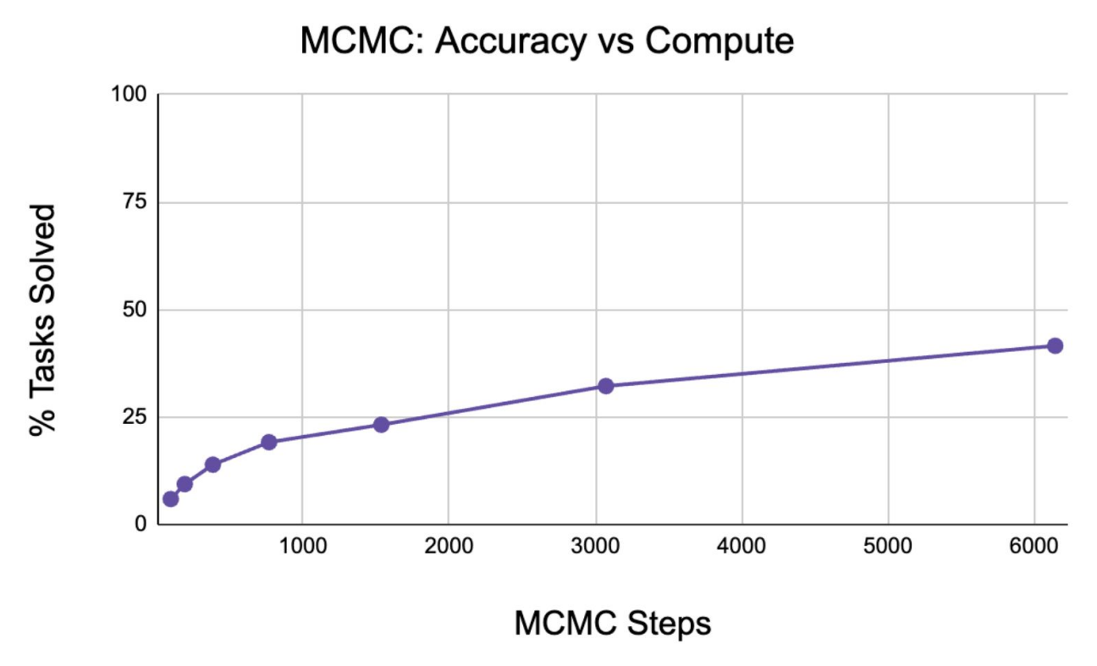
2. Probabilistic DSL (full model) outperforms deterministic DSL with noise on the output



3. High-level DSL is particularly important to SMC performance

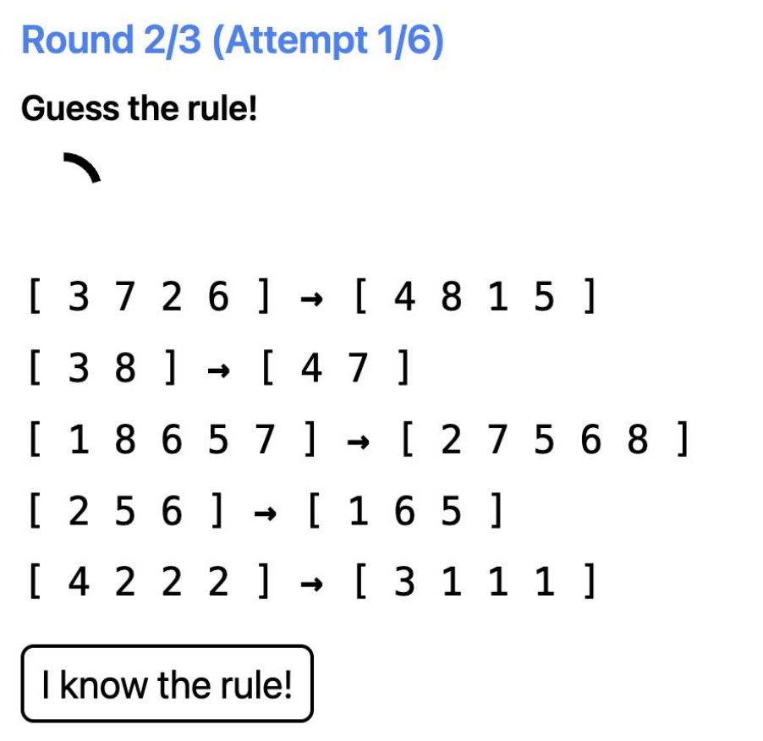






Pilot Design

Step 1: Participant has a limited amount of time to observe input-output examples.

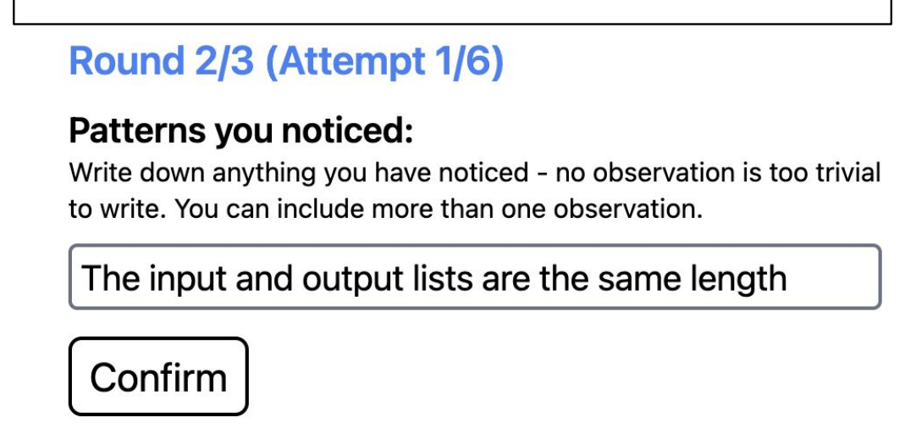


Round 2/3 (Attempt 1/6)

Guess the rule!

[3 7 2 6] → [4 8 1 5]
[3 8] → [4 7]
[1 8 6 5 7] → [2 7 5 6 8]
[2 5 6] → [1 6 5]
[4 2 2 2] → [3 1 1 1]

Step 2: Participant records observations they've noticed, regardless of whether they figured out the rule.



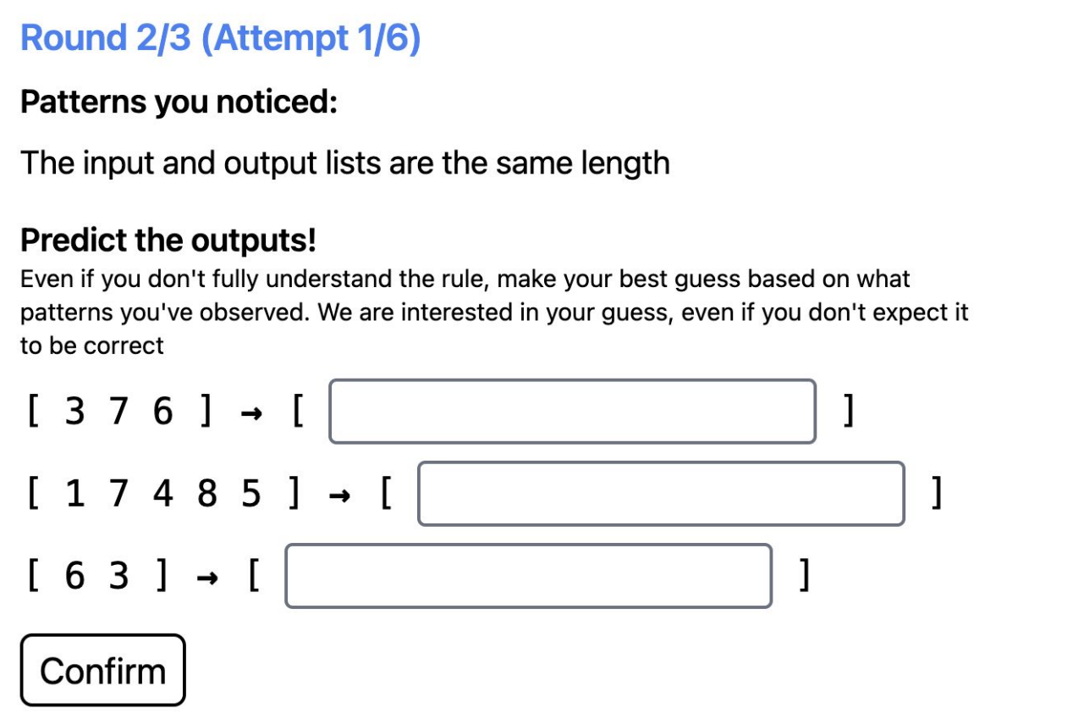
Round 2/3 (Attempt 1/6)

Patterns you noticed:

Write down anything you have noticed - no observation is too trivial to write. You can include more than one observation.

The input and output lists are the same length

Step 3: Participant tries to guess the outputs. If they are correct, they move to the next round (with a new rule). Otherwise they return to Step 1 with more time and the same rule. Time limits: 4s, 8s, 16s, 32s, 64s, ∞.



Round 2/3 (Attempt 1/6)

Patterns you noticed:

The input and output lists are the same length

Predict the outputs!

Even if you don't fully understand the rule, make your best guess based on what patterns you've observed. We are interested in your guess, even if you don't expect it to be correct

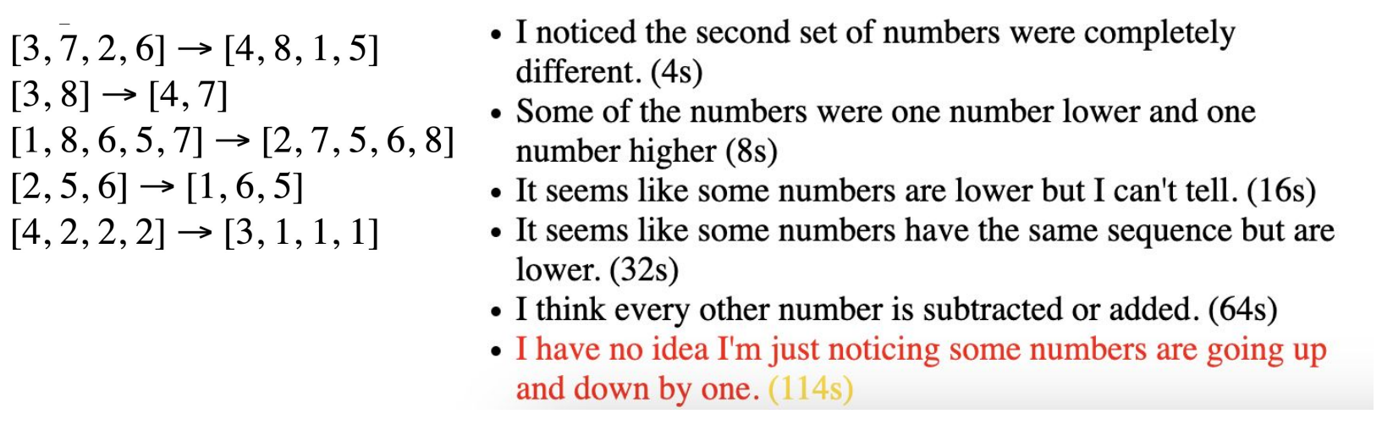
[3 7 6] → [ ]
[1 7 4 8 5] → [ ]
[6 3] → [ ]

Pilot Results

N=15 pilot to explore how people refine their guesses over time; some tasks adapted from [Rule 2020]. Example tasks and responses are given below.

Task: *Increment odds and decrement evens*

2/5 people got this correct, and 2 others ended with noticing that numbers were getting incremented/decremented but were unsure why, as in the example below.

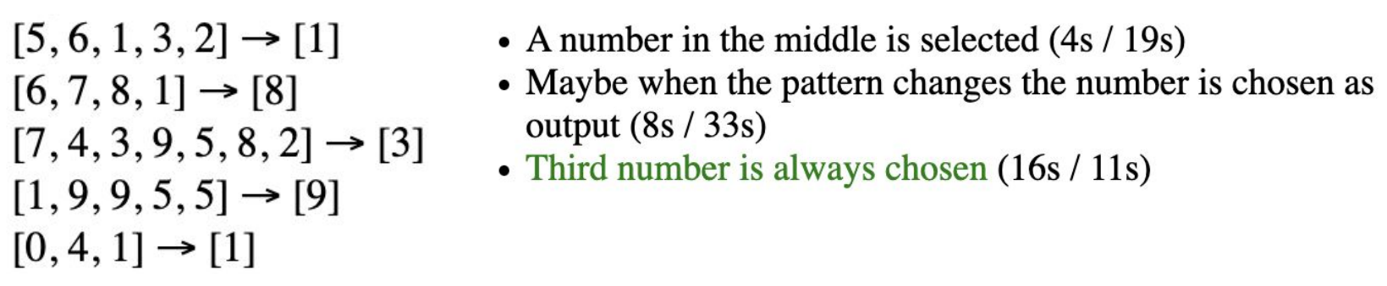


[3, 7, 2, 6] → [4, 8, 1, 5]
[3, 8] → [4, 7]
[1, 8, 6, 5, 7] → [2, 7, 5, 6, 8]
[2, 5, 6] → [1, 6, 5]
[4, 2, 2, 2] → [3, 1, 1, 1]

- I noticed the second set of numbers were completely different. (4s)
- Some of the numbers were one number lower and one number higher (8s)
- It seems like some numbers are lower but I can't tell. (16s)
- It seems like some numbers have the same sequence but are lower. (32s)
- I think every other number is subtracted or added. (64s)
- I have no idea I'm just noticing some numbers are going up and down by one. (114s)

Task: *Return the 3rd element of the list*

3/3 people got this correct, all within 16 seconds; if there is coarse to fine reasoning it's happening very fast



[5, 6, 1, 3, 2] → [1]
[6, 7, 8, 1] → [8]
[7, 4, 3, 9, 5, 8, 2] → [3]
[1, 9, 9, 5, 5] → [9]
[0, 4, 1] → [1]

- A number in the middle is selected (4s / 19s)
- Maybe when the pattern changes the number is chosen as output (8s / 33s)
- Third number is always chosen (16s / 11s)

Our model's trajectory:

```
1. RandList()                          # a random list
2. (cons RandDigit() [])               # a random 1-element list
3. (cons (index RandDigit() xs) [])    # index randomly into the list
4. (cons (index 3 xs) [])              # 3rd element of the list
```

Task: *Use the first number in the list to index into the list*

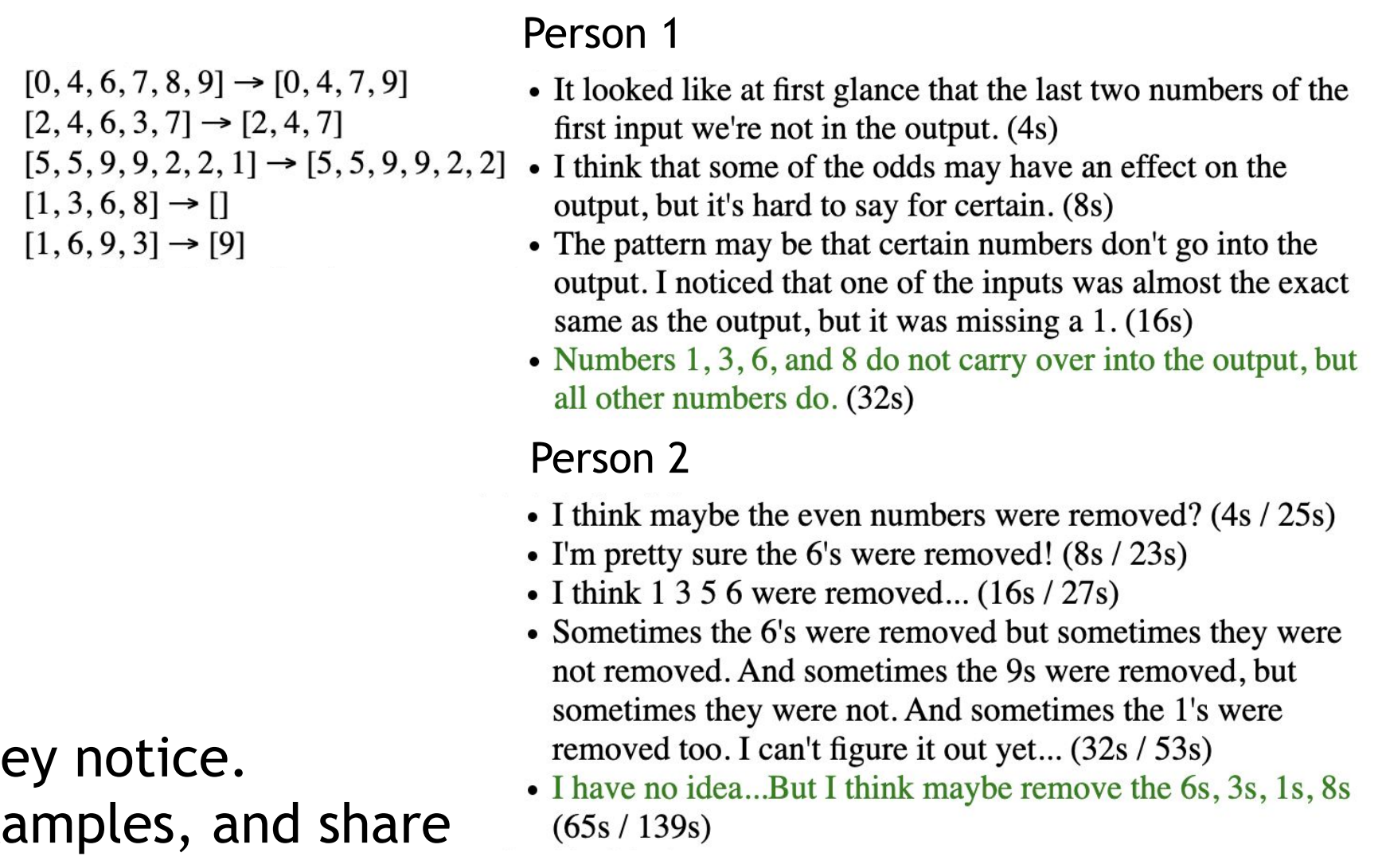
1/4 people got this correct, making it much harder than *Return 3rd element of list*, though our model takes a similar trajectory to the other task.

Task: *Append length of list on to end of list*

3/4 people got this correct, and 3/4 started by hypothesizing that *something* is appended to the list, as does our model.

Task: *Remove 1, 3, 6, and 8 from the list*

5/7 people got this correct, and 1 other person noticed "some numbers are being removed". People write many incorrect observations along the way.



[0, 4, 6, 7, 8, 9] → [0, 4, 7, 9]
[2, 4, 6, 3, 7] → [2, 4, 7]
[5, 5, 9, 9, 2, 2, 1] → [5, 5, 9, 9, 2, 2]
[1, 3, 6, 8] → []
[1, 6, 9, 3] → [9]

Person 1

- It looked like at first glance that the last two numbers of the first input we're not in the output. (4s)
- I think that some of the odds may have an effect on the output, but it's hard to say for certain. (8s)
- The pattern may be that certain numbers don't go into the output. I noticed that one of the inputs was almost the exact same as the output, but it was missing a 1. (16s)
- Numbers 1, 3, 6, and 8 do not carry over into the output, but all other numbers do. (32s)

Person 2

- I think maybe the even numbers were removed? (4s / 25s)
- I'm pretty sure the 6's were removed! (8s / 23s)
- I think 1 3 5 6 were removed... (16s / 27s)
- Sometimes the 6's were removed but sometimes they were not removed. And sometimes the 9s were removed, but sometimes they were not. And sometimes the 1's were removed too. I can't figure it out yet... (32s / 53s)
- I have no idea...But I think maybe remove the 6s, 3s, 1s, 8s (65s / 139s)

Some challenges and observations:

- It's hard to get people to write down everything they notice.
- People often hypothesize based on one or a few examples, and share these incorrect hypotheses rather than writing down the features they're sure of.

Future Work

Human Study

- There's a lot that people don't write down – what can we do to better probe people's intermediate hypotheses?
- Can we model the effect of people attending to only a few examples when proposing?
- More participants, and longer time limits
- Quantitative analysis of relationship between our model and the human data

Computational Modelling

- Richer MCMC & SMC moves, including the MCMC moves from [Saad et al. 2023]
- Library learning to bootstrap from a low-level DSL as in [Ellis et al 2021, Bowers et al 2023]
- Add domain: formal grammars from [Yang & Piantadosi 2022] where they similarly explore probabilistic program hypotheses through MCMC, but also have edit-noise on the list outputs.
- Add domain: world modelling domain like Autumn [Das et al 2023] or VGDL