Induction and Inquiry via Probabilistic Reasoning over Language and Code

Anonymous Author(s)
Affiliation
Address
email

Abstract

1

2 1 Introduction

Inductive reasoning is a cornerstone of general intelligence: Learning new concepts from few examples, and revising those concepts in light of new evidence. Limited data is inherently ambiguous, motivating an inquiry process of asking questions or doing experiments to resolve uncertainty. This induction-inquiry cycle unfolds sequentially over time, with new data streaming in, because inquiry is an active process of asking questions and getting answers. Modeling human induction and inquiry is a longstanding challenge because such models must handle uncertainty, have a flexible hypothesis class covering much of what humans can think of, and support efficient online computation. These objectives interact: A flexible, open-ended hypothesis class yields more uncertainty, because there are now more competing explanations for the evidence. But this causes reasoning to be computationally expensive. Decades of research [1, 2, 3] suggest human inductive reasoning approximates probabilistic Bayesian belief updates, but we still cannot truly model what people seem to do: Efficient online induction and inquiry over flexible open-ended hypothesis spaces. This is the challenge we take on.

We start with the Bayesian cognitive modeling paradigm, which imposes probabilistic norms for calculating how credible a belief should be, but as a paradigm, does not say what people can believe in the first place—how they can efficiently reason about an endlessly open-ended range of concepts. Prior models of inductive reasoning [4, 5] further posit an inner *Language of Thought*, whether formal logic, symbolic schemas or Bayes net templates, or probabilistic programs, which formalize and delineate what hypotheses are representable, and therefore learnable. The literature on intuitive theories and cognitive development has also proposed natural language as a representation of hypotheses [6, 7, 8], but this has never been made formal.

Here we find that human behavior across a range of induction and inquiry setups is best explained by sequential probabilistic reasoning over *mental programs*, which we treat as a mix of natural language and computer source code (fig. 1). Although the idea of an inner Language of Thought is an old one, its past computational instantiations assumed rigid logical forms that are less malleable than natural language, and less practical than actual programming languages.

Why represent knowledge as a mix of natural language and source code? Language and code are generic representations for communicating and formalizing human knowledge, but only recently have they become tractable targets of inference, owing primarily to Large Language Models (LLMs). Our models equip LLMs with sequential probabilistic reasoning. The resulting models reproduce sequential phenomena such as garden-pathing and anchoring; capture gradations of uncertainty; and scale to more complex concepts, because of the powerful combination of the expressivity of language and the top-down feedback of code. Furthermore, we show how these models can perform human-like active inquiry, closing the sequential learning loop which alternates between induction and inquiry.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

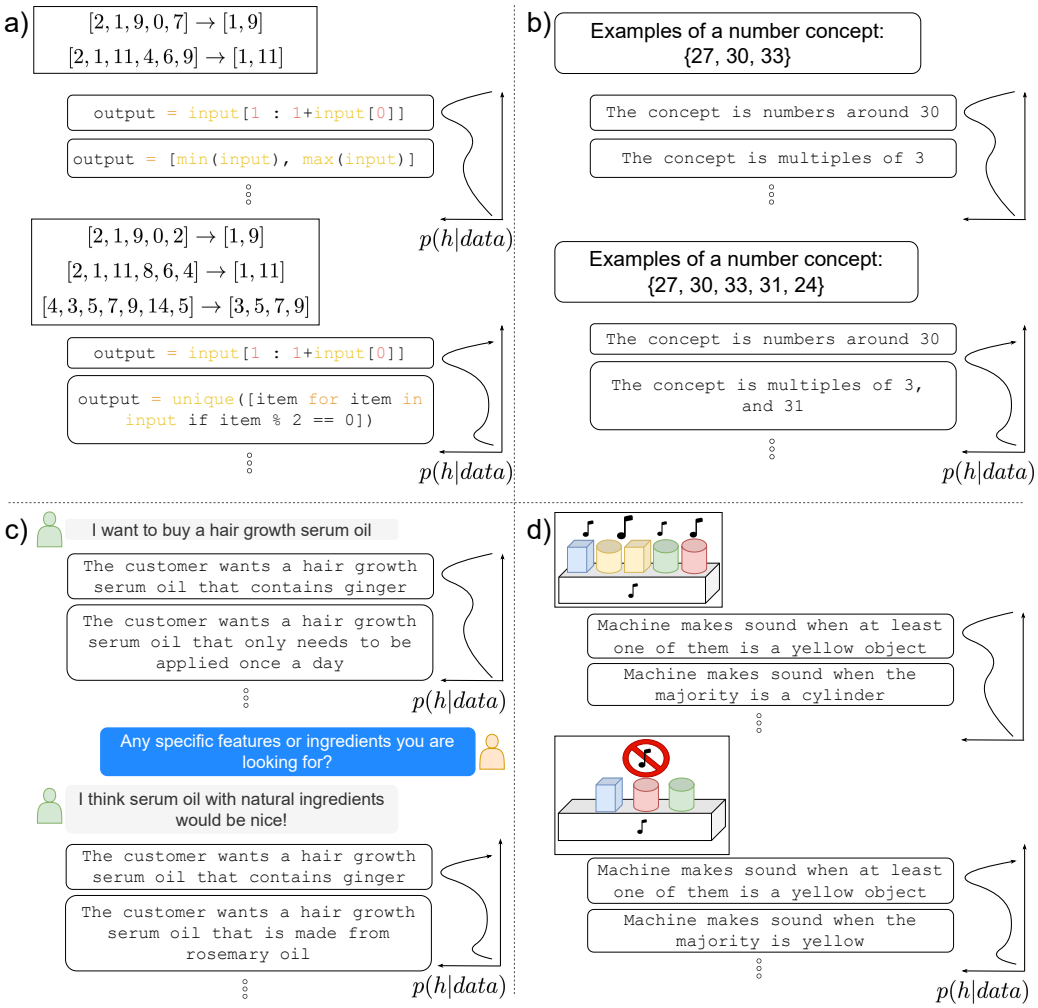


Figure 1: a) - d) show sequential inference problems that we study in this work and illustrate how, in each problem, Bayesian beliefs may change upon seeing new observations over time.

Speculatively, our model suggests mental representations that lie on a continuum between logic and language, and shows how this representation is compatible with Bayesian reasoning.

2 Computational Model

Humans encounter evidence sequentially over time: One instance of a new category is seen first, another second, etc. Limited data is inherently ambiguous, so we model humans as mentally representing multiple competing hypotheses, maintaining those that both fit the data and admit simple natural-language description. Upon receiving new evidence, humans update their beliefs: They inductively reason about whether new data forces new conclusions, or eliminates old hypotheses. Therefore our model compares the latest hypotheses to the data, and stochastically revises them to better fit the data. Representing hypothesis in language and code, and then revising hypotheses using large language models, allows efficient open-ended reasoning. Modeling multiple competing hypotheses captures the intuition that people can think of several different explanations, which allows rational inquiry by asking questions that optimally split the competing hypotheses.

Formally, given a sequence of T examples $e_{1:T}$, our model hypothesizes mental programs h. Each mental program has two pieces: (1) a natural language description and (2) a Python implementation. Mixing language and code allows freely generating ideas in natural language, but forces formalizing hypothesis into executable form. We define priors $p(h)$ that favor short natural language descriptions,

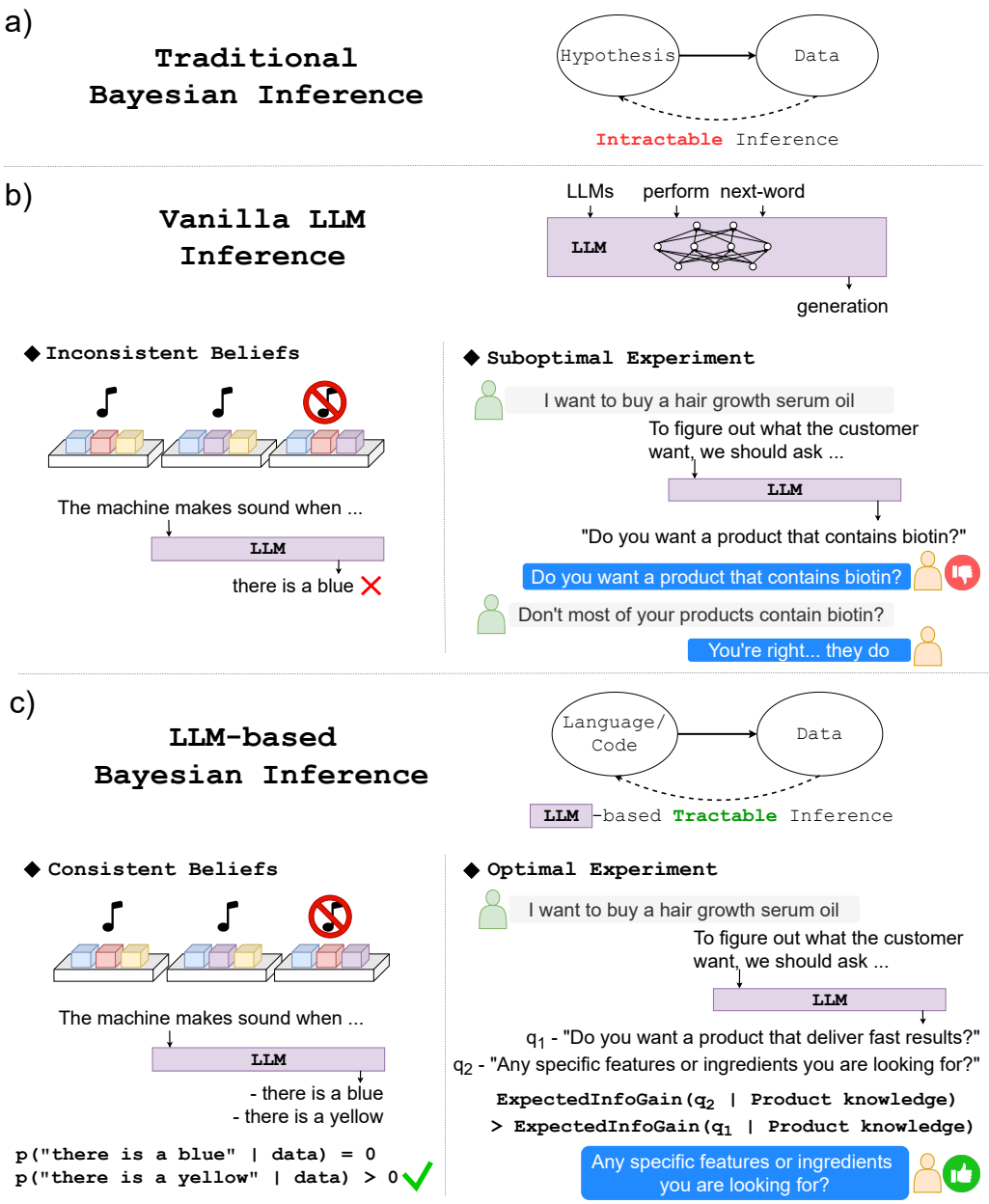


Figure 2: a) - c) show three types of inference methods: traditional Bayesian, vanilla LLM, and LLM-based Bayesian (ours). LLM-based Bayesian inference in language/code hypothesis space is the only method that is tractable while maintaining consistent beliefs and optimal experiments.

and likelihoods $p(e_t|h)$ that favor program executions that match the evidence. The language prior and code likelihood together define a posterior $p(h|e_{1:T})$, which evolves over time:

$$p(h|e_{1:T}) \propto p(e_T|h)p(h|e_{1:T-1}) \propto p(h)\prod_{t\leq T} p(e_t|h) \tag{1}$$

The above posterior is intractable because infinitely many hypotheses could explain the data. Instead, humans could only plausibly consider a small finite set of hypotheses.

How should we generate this small pool of possible hypotheses, given the vast hypothesis space of natural language and code? While we can compare competing hypotheses given the prior and likelihood (eq. (1)), we still need a heuristic proposal mechanism to know which hypotheses to

consider in the first place. LLMs are a natural choice. From a cognitive perspective, they are a fast bottom-up mechanism for suggesting different hypotheses, built through associative learning mechanisms that encode certain human priors by pretraining on human language. From an engineering standpoint, they serve as a data-driven proposal distribution over hypotheses h that *might* explain $e_{1:t}$, and where we can down-weight samples that do not fit the data by reweighing to target $p(h|e_{1:t})$, mitigating LLM hallucinations.

To evolve beliefs with each new piece of evidence, we use LLM-augmented Sequential Monte Carlo [9, 10], specifically LLM-SMC-S [11] (fig. 3). This maintains K particles $\left\{h_t^i\right\}_{i=1}^K$ representing candidate hypotheses after observing t examples, $e_{1:t}$. A prompt implements a bottom-up proposal distribution $q\left(h_{t+1}|e_{1:t+1}, \left\{h_t^i\right\}_{i=1}^K\right)$ which generates a new set of particles $\left\{h_{t+1}^i\right\}_{i=1}^K$, given the new evidence. Departing from standard SMC, we propose new particles given a global view of the previous posterior, which means all previous hypotheses are available in-context (Methods).

A bottom-up associative learner is not the only way of proposing hypotheses, but we think it is close to what happens in humans when drawing fast inferences from sparse data. Other related cognitive models either curtail the hypothesis space apriori—restricting what can be learned in principle—or demand exorbitant sampling budgets in an effort to cover the vast space of mental programs [12, 13]. But an LLM is not the whole story: Top-down probabilistic reasoning dampens the unpredictability of the language model; allows thinking longer by proposing more hypotheses; and supports a broader range of probabilistic queries, such as asking questions and doing experiments to resolve uncertainty by maximizing information gain.

3 Mental Algorithms from Sequential Observations

3.1 List functions

If humans can infer mental programs, then they should be able to learn new algorithms from examples. Many behavioral and modeling studies investigate this [14, 15, 16, 17, 18], but recently Rule et al. [19] substantially increased the behavioral and modeling challenge by testing human learners on 250 different algorithms, each learnable from a sequence of examples (fig. 1; algorithms 1-100 are easier to model, 101-250 are more challenging). This benchmark poses a modeling challenge because of the massive combinatorial search space of possible algorithms. To address this search problem, Rule et al. [19] design a custom programming language equipped with high-level search moves (termed

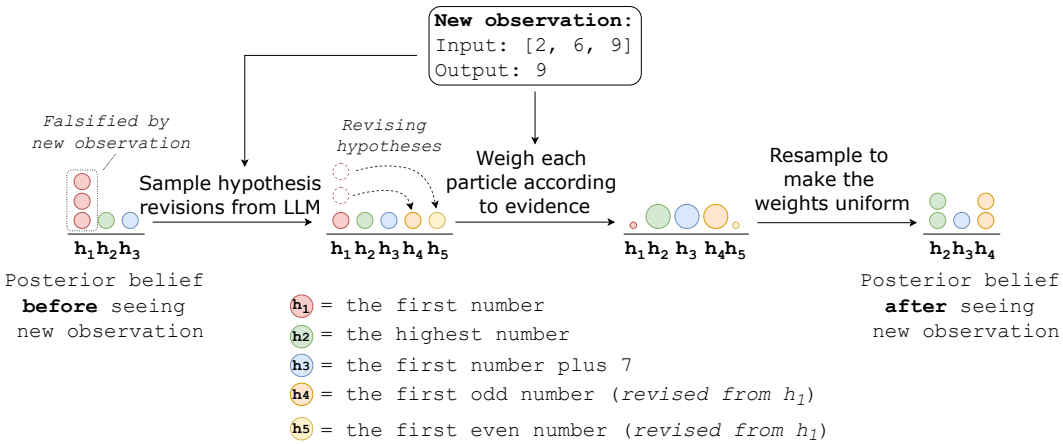


Figure 3: An illustration of how Sequential Monte Carlo methods change posterior belief upon receiving new observation. Sequential Monte Carlo method tracks a small number of hypotheses (called particles) represented above by circles. After each experiment, some particles are revised in light of the new observation, with the help of LLM. Then, the particles are reweighed according to how well each explains the observations we have seen so far. Resampling adjusts the weights of particles to be uniform by pruning low-probability hypotheses and multiplying high-probability ones.

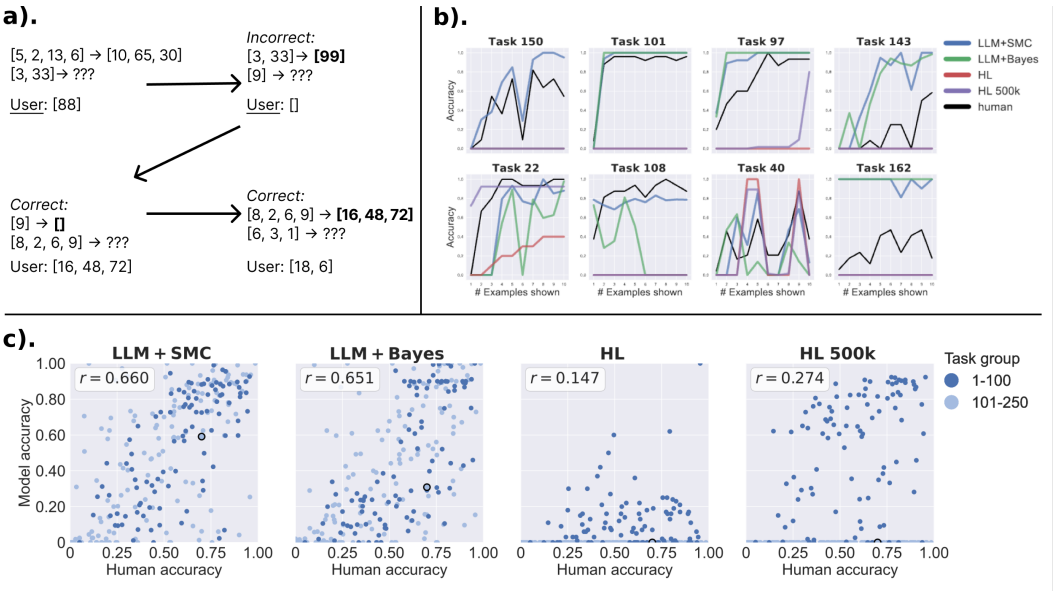


Figure 4: **a).** An example list function task: the participant is iteratively given examples and asked for predictions. This task – 150 – is visualized in panels b). (labeled) and c). (black outline). **b).** Posterior predictive curves for 4 models, and average accuracy for participants, on 8 randomly drawn tasks, across the 10 examples. Note that accuracy tends to increase as participants / models are shown more examples. **c).** Scatter plot of model accuracy and human accuracy for the 4 models in panel b). Model mean accuracy on the List Functions domain across examples versus mean human accuracy across participants and examples ($s = 5$ except for HL 500k, where $s = 500,000$).

HL), searching through up to 500k programs for each new input-output to find programs that explain the data. Plausibly, humans consider far fewer hypotheses— yet still learn these algorithms.

We test our model's ability to learn these algorithms while proposing (searching) far fewer hypotheses, and also test our model's ability to capture trial-by-trial dynamics of sequential inference. To study our ability to predict which algorithms are easier or harder to learn, Figure 4c plots human vs. model accuracy on 250 algorithms averaged across trials. At a search budget of just 5 proposals, our model fits the human data far better than HL given 500k proposals. This suggests a bottom-up proposal process could explain the search efficiency of human learners: With a neural proposal distribution, just a few samples suffice to predict average human accuracy. Modeling the sequence of examples proves important: Switching from Sequential Monte Carlo to Importance Sampling—which processes all examples at once—degrades model fit (fig. 4c, Importance Sampling). Figure 4b illustrates trial-by-trial accuracy for 8 randomly selected algorithms. Our model does not capture every detail of these learning curves, but for 70/100 algorithms, it matches these curves best (under MSE), with the remaining 30/100 roughly equally split between a best fit to HL, and a best fit to Importance Sampling.

The dataset of Rule et al. served as a significant challenge to both LLMs and conventional symbolic methods. Humans can learn these algorithms, but it took years of engineering to build a similarly performant model. Even then, prior work [19] confined itself to the 100 easier algorithms, and expended search effort far exceeding what humans plausibly perform. LLMs alone neither solve these problems nor fit human data. But LLM-guided sequential Bayesian inference suffices to solve this benchmark at human level, and reproduce basic human behavioral features.

3.2 Number concepts

Many human concepts are *categories*, rather than algorithmic functions, and are learned from only positive examples, such telling a child that an animal is 'cat', but not saying it isn't a giraffe. Here we study sequential learning of number categories, such as 'numbers ending in 3' or 'square numbers bigger than 20', following [20, 21]. When learning such concepts from small amounts of sequential

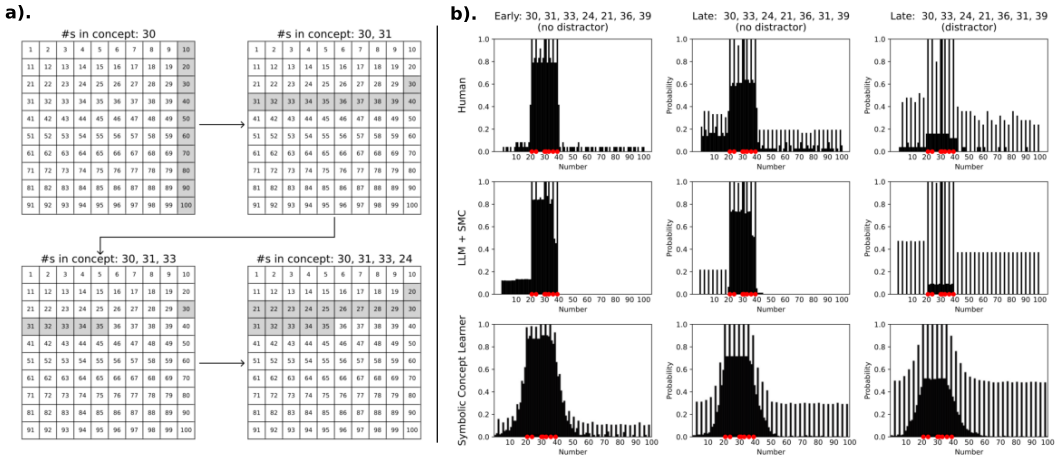


Figure 5: **a).** Demonstration of the number game task. Participants are sequentially shown a new example of an unknown number concept and asked to predict what other numbers 1-100 are in the concept (gray). **b).** Human data, our model predictions, and predictions from Thaker et al. [20] in 3 conditions. The early, distractor condition is omitted for clarity (results are nearly indistinguishable from the early, no distractor case).

data, humans show ordering effects such as *anchoring* or *garden-pathing*: A category that seems likely in earlier examples will dominate later inferences, even if invalidated by later data. For example, given the positive examples *30, 31, 33*, humans reliably infer a category such as *numbers around 30*, even when later data suggests multiples of 3, such as *24, 21, 36, 39*. Ordering the distractor *31* late in the sequence, such as *30, 33, 24, 21, 36, 31, 39*, has the opposite effect: Humans anchor to *multiples of three*, and discount the distractor.

Thaker et al. [20] study human number-category anchoring, which we computationally model (Figure 5). Our model's sequential inference successfuly reproduces the ordering effects seem in humans, and surprisingly, fits the human data better at smaller compute budget than the custom model in Thaker et al. Our model also reproduces attentional effects: When placed under greater cognitive load (a secondary distractor task), humans anchor more strongly. Modeling cognitive load by reducing our sampling budget replicates this effect (Supplement).

4 Resolving Uncertainty by Doing Experiments and Asking Questions

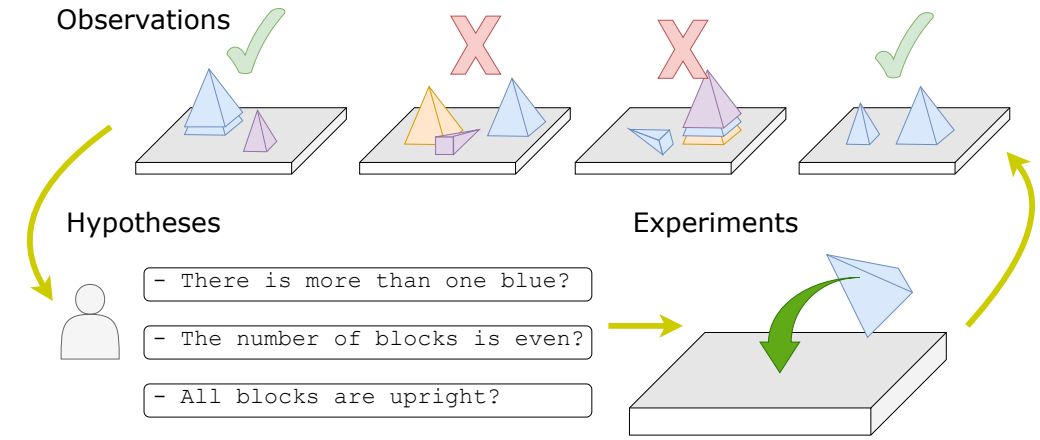
Humans can procure new data to aid learning by asking questions or trying out experiments in the real world, such as eating a new kind of berry to tell if it makes us sick. But it costs something to acquire new learning data, so humans need to decide whether to incur the cost of doing an experiment or asking a question to resolve uncertainty. Building on our sequential inference setup, we treat humans as considering different experiments ξ, and pick the experiment which maximizes expected information gain, under their particle-based approximate probabilistic beliefs:

$$\xi^* = \arg\max_{\xi} \mathop{\mathbb{E}}_{p(e|\xi,e_{1:t})} [D_{\mathrm{KL}}(p(h|e_{1:t},e)\|p(h|e_{1:t}))] \tag{2}$$

Exactly computing expected information gain is intractable. We make a particle-based approximation by treating the hypothesis space as the set of unique current particles, enabling tractable approximation of the requisite distributions.

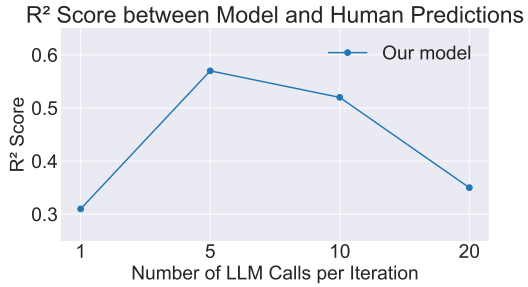
We first investigate this model by comparing its behavior to humans playing the game Zendo, which resembles classic 'blicket' studies in developmental psychology [22, 23, 24], but adds active experimentation. In Zendo, players infer a hidden binary category by building constructions from colored shapes, and then receiving feedback on if their construction belongs to the hidden category (Figure 6a). Each construction is an experiment ξ. We take human data from [25], where after 7 rounds of experimentation, participants make 8 predictions on holdout test constructions. Our model mimics a human participant by alternating between experimentation and inference, and finally testing on the the same holdout constructions. The resulting model captures fine-grained structure in

(a)



(b)

(c)



(d)

Method	LogL↑
Bramley et al. 2018 [25]	−1539
Important Sampling + Refinement	−3499.76
Importance Sampling	−1660.90
LLM-SMC-S	**−1478.82**

Figure 6: (a) Zendo gameplay (b) Human accuracy on examples that do not belong to the category (not rule following, 'Not RF') is higher than accuracy on in-category examples (rule following, 'RF'). Our model reproduces this phenomenon, while an LLM on its own does not. (c) Human-model R^2 has a U-shaped relationship to compute budget: Humans are boundedly rational, and increasing compute eventually degrades model fit. (d) The log likelihood of the human data is highest under our model, and surprisingly surpasses the fit of a custom model designed for this dataset [25], despite the fact that we do not hand-engineer a dataset-specific hypothesis space.

human error patterns (fig. 6b). As before, we get insight into boundedly-rational human behavior by modulating the inference-time budget (fig. 6c), and despite minimal domain-specific engineering, our model fits the human data better than a custom Bayesian learner designed specifically for this dataset (fig. 6d). We find therefore that the LLM-guided Bayesian learner is surprisingly versatile: Rational probabilistic reasoning tied to LLM backends support both concept learning and active experimentation, giving an induction-inquiry cycle that predicts human behavior better than LLMs or Bayes on their own.

Beyond Code: Asking Questions. In social contexts, the analog of an experiment is question-asking. Because our model operates over natural language, we can also use it to generate informative questions. We study this in a web shopping task where the model serves as a shopping assistant, and

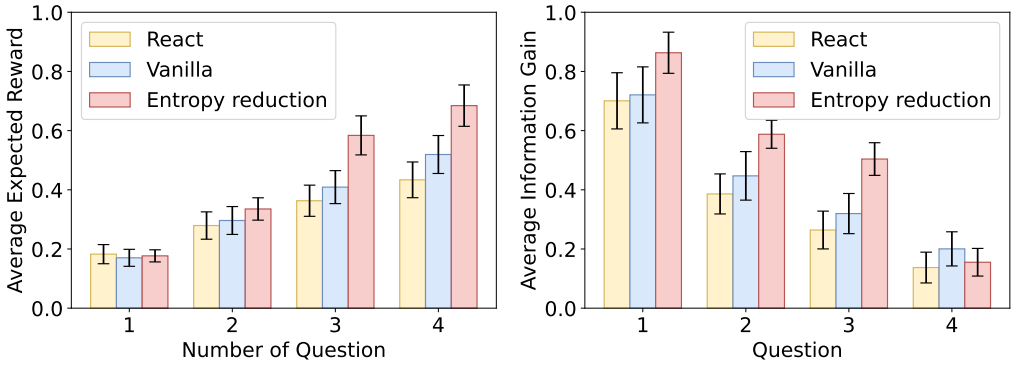


Figure 7: (Left) Average expected binary reward at increasing number of number of questions. (Right) Average information gain at each question.

asks natural-language questions that optimize information gain (eq. (2)). As a proof-of-concept, we assume a finite hypothesis space of products a customer can purchase, and perform exact Bayesian belief updates. An experiment ξ is a natural language question, an example e is a question-answer pair, and hypotheses h are different products (such as different brands of shampoo).

The agreement of a product h and a question-answer pair e is not generally expressible as a Python program. Therefore the likelihood $p(e|h)$ simply queries a language model, rather than generate code. Conceptually our argument is that *neither* natural language nor high-level programming languages truly capture the biases and expressiveness of human mental programs, so we should not expect that every problem is best solved by a hybrid of language and code.

Figure 7 compares our model with basic LLM prompts and also ReAct, which prompts an LLM to think before it acts. Our model chooses more informative questions, as defined by information gain, giving a higher chance of discovering the customer's preferred product. Although we mainly focus on modeling human data, we believe that these methods can also impact how AI systems are built.

5 Discussion

How people learn new categories, laws, and abstract concepts from the sparse streaming data of experience is a difficult open question. Candidate computational models must be simultaneously flexible and efficient—which are generally in tension. To make progress on that question, our models make mechanistic commitments about the underlying mental representation—code and language—and the underlying mental algorithms, which use neurally-guided sequential inference for tractable reasoning over open-ended hypothesis spaces. Across a range of inductive reasoning problems, the resulting model fits human data better than specialized models designed for each individual behavioral experiment, and further allow induction to alternate with inquiry, explaining how online learning and acting can cooperate together. Our results suggest a language of thought that lies between logic and language, and also suggest that we are not far from a unified computational account of human induction and inquiry that could explain how humans think flexibly across the endless range of situations in which these cognitive faculties can be brought to bear.

Cognitive Implications. Our models assume a solution space whose hypotheses are at least definable in clear English language. Yet many human concepts are famously tricky to formalize, such as the meaning of 'chair' [**?**] or even 'dog walking' [26]. It remains open whether symbolic hypotheses are the right representation for such categories, but our view is that a symbolic language of thought remains the best account of inductive reasoning from small data. We use a specific language of thought termed *mental programs*, which combines natural language and code. Code can represent what would be hard to precisely express language, such as detailed physical properties, the shape of a hand-drawn character [27], or the precise rules of a board game. Language can represent fuzzier, higher-level propositions. Current AI heavily uses language as a knowledge representation, and achieves unprecedented coverage as a result: LLMs can converse about virtually every human topic, even if they do not always make sense. Classic Bayesian models using symbolic programs have

succeeded at modeling humans within narrow domains by tailoring the representation to the problem domain [28, 29, 30, 25, 31, 32, 33, 34]. Our work suggests that Bayesian priors should operate over in language-like representations, but that the grounding between hypothesis and data—the likelihood function—is a better fit for programs. Potentially, a future unified representation could serve both roles, which we view as an important open direction.

Bayesian cognitive models are known both for their predictive power and their computational intractability, and have been criticized as lacking an account of their biological implementation. Our work is not alone in trying to address these issues with Bayesian models: Metalearning offers another tractable neural instantiation of Bayes, where a neural network approximates the Bayesian predictive distribution [35, 36, 37] in a single forward pass, either via in-context learning or via MAML-style [38] weight updates. Unlike our work, such models do not reason over latent discrete representations. This is complementary to our models: We construct explicit verbalizable hypotheses, do not require training new neural networks, and can trade more inference-time compute for better predictions. At the same time, relative to metalearning, our approach has important limitations: It cannot learn what pretrained models do not already understand, and is unlikely to be a good account of fast, non-verbalizable inference. Roughly, we think of our models as a System 2 way of using neural networks for approximate reasoning, while metalearning is best thought of as a fast System 1 process. Humans likely use both strategies when thinking probabilistically.

Neural networks and Bayes. LLMs are probabilistic models trained on masses of human data—yet, in isolation, they do not reproduce human behavior in the tasks considered here. Why is that? Fundamentally, probabilistic inference requires reasoning about uncertainty. Reasoning requires expending variable compute, to think longer on harder problems. Handling uncertainty further constrains the model to the laws of probability. The newest LLMs are increasingly trained to reason via reinforcement learning (using chain-of-thought [39]), but to date such training focuses on problem-solving, not probabilistic inference. Chain-of-thought may also not be sufficiently constrained to ensure sound convergence, unlike the classic Monte Carlo algorithms that we and others build on [10]. It remains open however whether future LLMs could implicitly learn to mimic the reasoning patterns of sound inference algorithms.

cut? Engineering Implications. To get to human-level AI, the dominant paradigm today trains larger models on more data, effectively scaling learning. We consider scaling thinking, but very differently from large reasoning model such as OpenAI's O1 [40] and Deepseek's R1 [41]: Using principled probabilistic inference, critically in a way that leverages and complements the dominant scaling route by using LLMs as foundational components. We hope that this work can serve as a blueprint for combining LLMs with classic probabilistic reasoning methods, whose range of application is wide-ranging, including program analysis, social dialogue (i.e. pragmatic inference), forecasting, planning, and physical reasoning, to name a few. Our pilot experiments in question-asking are one example of this.

AMBITIOUS FORWARD LOOKING CONCLUSION

References

[1] John R. Anderson. The adaptive character of thought. 1990.

[2] Thomas L. Griffiths, Falk Lieder, and Noah D. Goodman. Rational use of cognitive resources: Levels of analysis between the computational and the algorithmic. *Topics in Cognitive Science*, 7(2):217–229, 2015.

[3] T.L. Griffiths, N. Chater, and J. Tenenbaum. *Bayesian Models of Cognition: Reverse Engineering the Mind*. MIT Press, 2024.

[4] Jake Quilty-Dunn, Nicolas Porot, and Eric Mandelbaum. The best game in town: The reemergence of the language-of-thought hypothesis across the cognitive sciences. *Behavioral and Brain Sciences*, 46:e261, 2023.

[5] Steven T. Piantadosi. *Learning and the language of thought*. PhD thesis, MIT, 2011.

[6] Susan Carey. The origin of concepts: A précis. *The Behavioral and brain sciences*, 34(3):113, 2011.

[7] Elizabeth Spelke. *What Makes Us Smart? Core Knowledge and Natural Language*, pages 277–312. 03 2003.

[8] Elizabeth Spelke. *What babies know: Core knowledge and composition volume 1*, volume 1. Oxford University Press, 2022.

[9] Alexander K Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*.

[10] Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential monte carlo. *arXiv preprint arXiv:2404.17546*, 2024.

[11] Wasu Top Piriyakulkij, Cassidy Langenfeld, Tuan Anh Le, and Kevin Ellis. Doing experiments and revising rules with natural language and probabilistic reasoning. In *Neural Information Processing Systems*, 2024.

[12] Steven T Piantadosi, Joshua B Tenenbaum, and Noah D Goodman. The logical primitives of thought: Empirical foundations for compositional cognitive models. *Psychological review*, 123(4):392, 2016.

[13] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *PLDI*, 2021.

[14] Steven T Piantadosi. The computational origin of representation and conceptual change. 2016.

[15] Brenden M Lake and Steven T Piantadosi. People infer recursive visual concepts from just a few examples. *Computational Brain & Behavior*, 3(1):54–65, 2020.

[16] Bonan Zhao, Christopher G Lucas, and Neil R Bramley. A model of conceptual bootstrapping in human cognition. *Nature Human Behaviour*, 8(1):125–136, 2024.

[17] Céline Hocquette, Johannes Langer, Andrew Cropper, and Ute Schmid. Can humans teach machines to code?, 2025.

[18] Douglas R Hofstadter and Melanie Mitchell. The copycat project: A model of mental fluidity and analogy-making. 1994.

[19] Joshua S Rule, Steven T Piantadosi, Andrew Cropper, Kevin Ellis, Maxwell Nye, and Joshua B Tenenbaum. Symbolic metaprogram search improves learning efficiency and explains rule learning in humans. *Nature Communications*, 15(1):6847, 2024.

[20] Pratiksha Thaker, Joshua B. Tenenbaum, and Samuel J. Gershman. Online learning of symbolic concepts. *Journal of Mathematical Psychology*, 77:10–20, 2017.

[21] Joshua Brett Tenenbaum. *A Bayesian framework for concept learning*. PhD thesis, Massachusetts Institute of Technology, 1999.

[22] Alison Gopnik and David M Sobel. Detecting blickets: How young children use information about novel causal powers in categorization and induction. *Child development*, 71(5):1205–1222, 2000.

[23] Claire Cook, Noah D. Goodman, and Laura E. Schulz. Where science starts: Spontaneous experiments in preschoolers' exploratory play. *Cognition*, 120(3):341–349, 2011. Probabilistic models of cognitive development.

[24] Chi Zhang, Baoxiong Jia, Mark Edmonds, Song-Chun Zhu, and Yixin Zhu. Acre: Abstract causal reasoning beyond covariation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 10643–10653, 2021.

[25] Neil Bramley, Anselm Rothe, Josh Tenenbaum, Fei Xu, and Todd Gureckis. Grounding compositional hypothesis generation in specific instances. In *Proceedings of the 40th annual conference of the cognitive science society*, 2018.

[26] Max H. Quinn, Anthony D. Rhodes, and Melanie Mitchell. Active object localization in visual situations, 2016.

[27] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[28] Noah D Goodman, Joshua B Tenenbaum, Jacob Feldman, and Thomas L Griffiths. A rational analysis of rule-based concept learning. *Cognitive science*, 32(1):108–154, 2008.

[29] Marie Amalric, Liping Wang, Pierre Pica, Santiago Figueira, Mariano Sigman, and Stanislas Dehaene. The language of geometry: Fast comprehension of geometrical primitives and rules in human adults and preschoolers. *PLoS computational biology*, 13(1):e1005273, 2017.

[30] Kevin Ellis, Adam Albright, Armando Solar-Lezama, Joshua B Tenenbaum, and Timothy J O'Donnell. Synthesizing theories of human language with bayesian program induction. *Nature communications*, 13(1):5024, 2022.

[31] Goker Erdogan, Ilker Yildirim, and Robert A Jacobs. From sensory signals to modality-independent conceptual representations: A probabilistic language of thought approach. *PLoS computational biology*, 11(11):e1004610, 2015.

[32] Mathias Sablé-Meyer and Stanislas Dehaene. Visual sequence primitives in humans. Master's thesis, ENS, 2017. DSL proposal in appendix: A proposal for a Language of Shape.

[33] Lucas Tian, Kevin Ellis, Marta Kryven, and Josh Tenenbaum. Learning abstract structure for drawing by efficient motor program induction. *Advances in Neural Information Processing Systems*, 33:2686–2697, 2020.

[34] Feras A Saad, Marco F Cusumano-Towner, Ulrich Schaechtle, Martin C Rinard, and Vikash K Mansinghka. Bayesian synthesis of probabilistic programs for automatic data modeling. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–32, 2019.

[35] Brenden M Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121, 2023.

[36] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

[37] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*, 2018.

[38] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[39] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[40] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[41] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

A Appendix

A.1 Redundant/Move to methods

sample-matched model accuracy versus human accuracy for the **Full model**, **HL**, and **Sample+Weight** for a random subset of 40 tasks. Each model uses 5 samples / search steps per example. Across the 100 tasks, the **Full model** has the closest fit to human data (measured by mean squared error (MSE) in the 10 examples), for 70 of 100 tasks, **Sample+Weight** for 17 of 100 tasks, and **HL** for the remaining 13.

the model average accuracy of the model against the average human accuracy across all 100 algorithms,

ofhas a comprehensive algorithm-learning

We first study algorithmic and numerical concepts (Figure 1a-b): Learning, from input-outputs, algorithms on lists of numbers (), and learning, from positive-only examples, a novel number category, such as numbers from 30-40.

Algorithmic concepts. [19]

In the List Functions domain, the task is to, given a set of input-output examples, induce the underlying program which maps each input list to its corresponding output list. The learner then must apply this induced program to a new input for which the output is hidden. One such task from the domain is:

We model the human data using the LLM+SMC model outlined above. We use `gpt-4-0613` [**?**] as our proposal distribution q over natural language language hypotheses H. Preliminary experiments showed that other publicly-accessible LLMs perform substantially worse (**TODO: these experiments were 1 year ago. I should probably re-run w/ more modern (and much cheaper) models**). For our prior, we find that a simple length-prior $p(H) \propto \frac{1}{|H|}$ works just as well as a linear prior model fitted to human data, in contrast to previous work in concept learning.

For our likelihood function, we use a simple likelihood function

$$P(X_{1:K} \mid H) = \prod_{1 \leq k \leq K} (1-\theta)\frac{\mathbb{1}[X_k^o = H(X_k^i)]}{K} + \theta\frac{\mathbb{1}[X_k^o \neq H(X_k^i)]}{K} \tag{3}$$

where θ is the probability that an example is mislabeled. Note that the likelihood increases monotonically with the number of correct examples; a hypothesis will always have a higher likelihood if it can directly explain more of the data than another hypothesis. Initially, we set $\theta = \frac{1}{100}$, and after proposing all hypotheses, we fit θ to human data using k-fold cross-validation, with $k = 10$. Fitting the likelihood parameters, in this case only θ, while running the model is interesting future work that may improve performance.

We compare our model – **Full model** – with two baselines. The **Hacker-like (HL)** model [**?**] is a search algorithm over *meta-programs* that uses *term-rewriting systems* to drastically improve search efficiency over alternative symbolic search algorithms such as Fleet [**?**] and Metagol [**?**]. The **Sample+Weight** baseline removes the sequential aspect of our model, treating each example k as a separate task.