From the holdings of the Department of Special Collections , Princeton University Library, Princeton, NJ

These documents are either in the public domain, or can only be used for educational and research purposes (Fair use) as per U.S. Copyright law (text below). By accessing this file, all users agree that their use falls in the public domain or within fair use as defined by the copyrightlaw. We do not charge any permission or usage fees for the publication of images of material inour collections, including those provided by Princeton University Library, via our website, catalog, or directly from staff.For more information regarding copyright, credit, and citation please review the information onthe Copyright, Credit and Citation Guidelines page of our website.
https://library.princeton.edu/special-collections/policies/copyright-credit-and-citation-guidelines. If you have any questions, including how to order higher-quality images from the collection, contact Special Collections Public Services via our Ask Us! Form. https://library.princeton.edu/special-collections/ask-us

---

Master's Theses, 2022-2026

Computer Science, 2023

2025

Understanding the Reverse Engineering Abilities of Large Language Models

Geng, Jiayi

Princeton, NJ : Princeton University

---

http://arks.princeton.edu/ark:/88435/dsp0102871029d

U.S. Copyright law: The copyright law of the United States (Title 17, §108, United States Code) governs the making of photocopies or other reproductions of copyrighted material. Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of fair use that user may be liable for copyright infringement. This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve a violation of copyright law. https://www.copyright.gov/title17/title17.pdf

Understanding the Reverse Engineering Abilities of Large Language Models

Jiayi Geng

A Master's Thesis
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Master of Science in Engineering

Recommended for Acceptance by
the Department of
Computer Science

Advisors: Danqi Chen; Thomas L. Griffiths

May 2025

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Jiayi Geng

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Jiayi Geng

Abstract

With the rapid development of large language models (LLMs), these models have demonstrated impressive capabilities not only in formal reasoning tasks but also certain desirable behaviors similar to human thinking. The emergence of these cognitive-like patterns motivated me to leverage insights from cognitive science to better understand the remaining challenges and explore the boundaries of the capabilities of LLMs. In this thesis, I mainly focus on the challenges that LLMs face when engaged in autonomous scientific processes. Using AI to create autonomous researchers has the potential to accelerate scientific discovery. A prerequisite for this vision to become reality is evaluating how well an AI model can identify the underlying structure of a system from its behavior. In this thesis, we explore the question of whether an LLM can learn from passive observations and actively collect informative data to refine its own hypotheses. To answer this question, we investigate the ability of LLMs to reverse-engineer three types of black-box systems chosen to represent problems that might appear in different domains of research: list mapping programs, formal languages, and mathematical equations. We use Bayesian models as a normative reference to quantify the gap between LLMs and optimal inference under a given observation space. Through extensive experiments, we show that while LLMs have difficulty reverse-engineering these systems from observations alone, data generated by LLM-driven interventions can effectively improve the models' own performance. By testing edge cases, the LLM is able to refine its own hypotheses and avoid failure modes such as *overcomplication*, where the LLM falsely assumes prior knowledge about the black box, and *overlooking*, where the LLM fails to incorporate observations. These insights provide practical guidance for helping LLMs more effectively reverse-engineer black-box systems, supporting their use in making new discoveries.

Acknowledgments

The past two years at Princeton have profoundly transformed my academic perspective and strengthened my confidence to step beyond my comfort zone. I would like to express my sincere gratitude to my advisor, Professor Danqi Chen, for providing me with an environment that continually sparked my curiosity. Working closely with Danqi on multiple papers significantly improved my engineering and communication skills. I am grateful to my other advisor, Professor Tom Griffiths, who introduced me to cognitive psychology, broadening my perspective and teaching me creative thinking, effective paper framing, and the value of interdisciplinary research. Beyond Princeton, I want to thank Graham Neubig from CMU and Ilia Sucholutsky from NYU for their support throughout my Master's journey.

I am infinitely fortunate to have had Howard Chen as my mentor during my Master's program. His insightful guidance, clear direction, kindness, and countless meetings have been instrumental in my decision to pursue further research and a PhD. Over the past two years, beginning with very minimal exposure to AI, Howard patiently introduced me to the field: teaching me how to write elegant Python code, independently manage projects, clearly structure and present my work, and thoughtfully reflect on my future vision, career, and life. His meticulousness, patience, generosity, and support have exceeded my greatest expectations.

Looking back, I am especially grateful for the invaluable time spent with talented and supportive colleagues. I deeply appreciate my primary co-authors, Ryan Liu and Mengzhou Xia, for our close collaborations, and the other collaborators who work with me on at least one paper during my Master's study: Sanjeev Arora, Mengdi Wang, Shuyan Zhou, Manoel Horta Ribeiro, Joshua C Peterson, Dilip Arumugam, Jiahao Qiu, Kaixuan Huang, Alexis Chevalier, Alexander Wettig, Addison Wu, Adithya Bhaskar, Dan Friedman, Xingjian Zhang, Shuyao Zhou, Yuhan Liu. Also, I want to thanks my labmates, Zexuan Zhong, John Yang, Carlos E. Jimenez, Ofir Press,

Tianyu Gao, Lucy He, Howard Yen, Shunyu Yao and Austin Wang for their consistent support.

At Princeton, I have also been fortunate to find like-minded friends, especially Ruyi Ke and Yanjin Chen, whose unwavering support, guidance, and encouragement have sustained me through numerous challenges. Lastly, my sincere appreciation goes to Princeton NLP, PLI, the CocoSci Lab, and the AI Lab for their generous financial support of my research.

Contents

Abstract 1

Acknowledgements 2

List of Tables 6

List of Figures 6

1 Introduction 8

2 Related Work 11

3 Reverse Engineering 14

3.1 Problem Formulation 14

3.2 Evaluation Protocol 15

3.3 Black Box Types 15

4 Experiments: How Well LLM Do Reverse-Engineering? 17

4.1 Experiments Setup 17

4.2 Inferring Hypotheses from Observations 17

4.3 How Well Can LLM Use Observations? 18

4.4 Effective Intervention Is Crucial for Solving Reverse Engineering Problems 19

4.5 Two Common Failure Modes: Overcomplication & Overlooking . . . 20

4.6 Effective Intervention via Iterative Hypothesis Refinement 21

5 Analysis 24

5.1 Reverse Engineering Abilities Across Different Categories of LLMs . . 24

5.2 Transferability of intervention data 25

5.3 How to Evaluate Reverse-Engineering? 27

6 Future Directions 29

6.1 Training LLMs 29

6.2 Controlled Experiments vs Realistic Simulations 29

6.3 Multi-Agent Collaboration 30

7 Conclusion 31

8 Appendix 42

8.1 Black Box Designs 42

8.2 Bayesian models as the 'Optimal' Reference 44

8.3 Statistical Significant Tests 47

8.3.1 Repeated-measures ANOVA 47

8.4 Prompts 48

8.4.1 Intervention prompt 48

8.4.2 Evaluation Prompts 50

List of Tables

4.1 Different reasoning approaches during the LLM intervention. Intv refers to the intervention 23

5.1 Different trends between descriptive evaluation and functional evaluation for Observation-only. 27

List of Figures

1.1 Left: the reverse engineering problem. The AI reverse engineer needs to collect observations from the black box and construct a hypothesis. Right (top): with only passive observations, the LLM cannot make effective use of the data and lags behind the optimal by large margin; allowing the LM to intervene bridges the gap. Right (bottom): effective intervention can mitigate two common failure modes: overcomplication and overlooking. 9

4.1 Observation-only results for GPT-4o across three black box types. We compare the LLM performance to Bayesian reference. For Program and Formal Language, we use LLM to judge whether the reverse rules can describe each black box instance. For math equation, to better compare with Bayesian model, we directly calculate the flipped RMSE. 19

4.2 Observation-intervention results for GPT-4o across the three black-box types. We compare the performance of LLM interventions against the Bayesian reference under two conditions: using only observations, and using interventions derived from LLM queries. 20

4.3 Descriptive scores for two different complexity levels. Averaged across three seeds for each of the three black-box types. For math equation, we use descriptive evaluation instead of RMSE only. 21

4.4 Example of how LLM intervene with list-mapping program black box. Before interacting with the black box, LLM reasons about past observation patterns and comes up with the internal mental model of the black box to test. 22

5.1 Results of reverse engineering abilities across different categories of LLMs. We report Llama-3.3-70B-Instruct, Claude 3.5 Sonnet and Deepseek R1 on three black box types. 25

5.2 Transfer to itself (active learning to passive learning). O+I+T: LLM takes the intervention data from others as the passive observations. . 26

5.3 Transfer to another LM: GPT-4o interventions as the passive yoked observations to Llama-3.3-70B-Instruct 26

Chapter 1

Introduction

Developing intelligent systems to accelerate scientific discovery has been a long-standing goal of artificial intelligence research [20, 63]. Despite rapid progress in creating large language models (LLMs) for understanding text and solving reasoning problems (e.g., math and code), automating scientific processes poses a different kind of challenge. A core aspect of scientific discovery is **reverse-engineering** the underlying mechanism behind a black-box system, which requires capabilities beyond generating one-off answers. In particular, reverse-engineering often involves 1) understanding a collection of data (often passively observed) to build hypotheses and 2) experimenting to actively acquire informative data from the black box to either validate or revise those hypotheses.

Existing work using LLMs to automate scientific processes relies heavily on static observational data [46, 51] or emulates scientific workflows using "LLM scientists" with many moving parts [19, 49]. By contrast, research in related fields has used carefully controlled settings to study the abilities of AI systems to perform inductive reasoning [47], the role of observation versus intervention [27], and optimal experimental design [8, 17]. This controlled approach has not yet been applied to understanding the capabilities of LLMs; that is, the fundamental questions of *"How well can a LLM learn from passive observations?"* and *"Can an LLM actively collect informative data to help refine its own hypothesis?"* have not been systematically investigated.

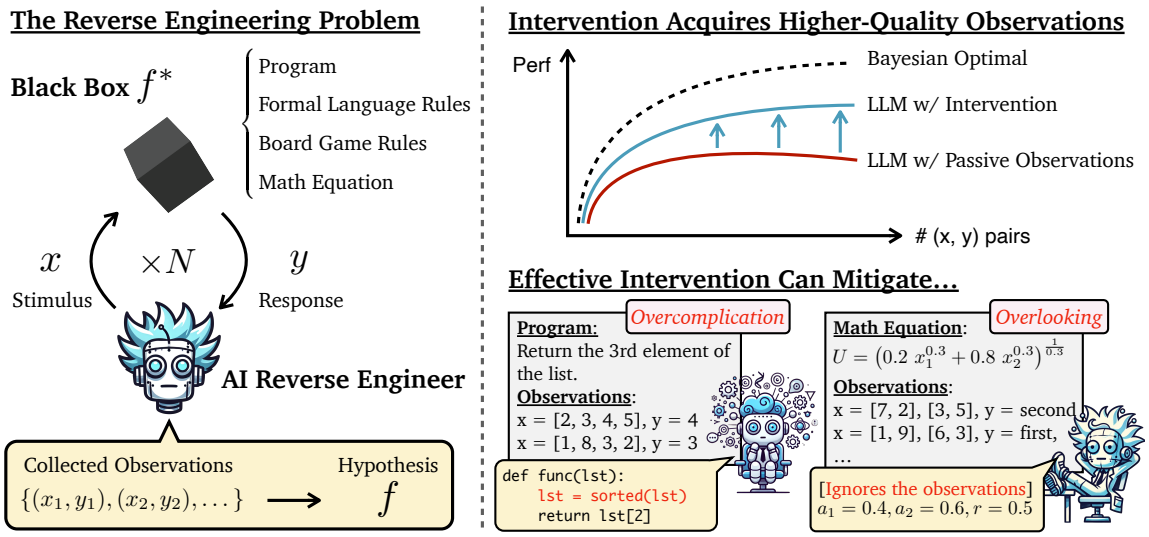

Figure 1.1: Left: the reverse engineering problem. The AI reverse engineer needs to collect observations from the black box and construct a hypothesis. Right (top): with only passive observations, the LLM cannot make effective use of the data and lags behind the optimal by large margin; allowing the LM to intervene bridges the gap. Right (bottom): effective intervention can mitigate two common failure modes: overcomplication and overlooking.

To bridge this gap, we[1] conduct a set of controlled tasks to systematically analyze the strengths and weaknesses of LLMs in reverse-engineering. Specifically, we employ three scaled-up reverse-engineering tasks inspired by cognitive psychology literature, designed to reflect challenges likely to arise in scientific settings: reconstructing list-mapping programs [47], formal languages [40], and mathematical equations [17]. We demonstrate that Bayesian search identifies the optimal hypothesis given the available observation space, whereas LLMs with access to the same observations suffer from significant performance deficits. These deficits arise from their inability to extract adequate information from the observations, which causes their performance to plateau. Allowing LLMs to perform interventions—actively generating test cases or queries to obtain new and informative data—significantly enhances their performance. However, we further show that the benefits from these actively collected

[1]This project was done in collaboration with Howard Chen, Dilip Arumugam, and Thomas L. Griffiths.

intervention data do not readily transfer across different LLMs, suggesting that such data are primarily beneficial only to the collecting model rather than objectively superior. Further analysis reveals that effective interventions mitigate two observed failure modes: (1) *overcomplication*, wherein the LLM constructs unnecessarily complex hypotheses; and (2) *overlooking*, wherein the LLM neglects important observations or prematurely generalizes conclusions. Finally, we find that supplementing LLMs with explicit reasoning processes does not necessarily lead to improved performance of reverse engineering.

The contributions of this thesis are as follows:

- We formalize reverse-engineering as the core problem for understanding the scientific discovery capabilities of LLMs and design four black-box tasks, varying in domain and complexity, for studying this process.
- We demonstrate LLMs' limitations in reverse-engineering when provided with only passive observations, indicating the need for LLMs to perform active interventions.
- We show that LLMs can perform interventions to obtain more-informative observations, and that effective intervention mitigates the common failure modes of overcomplication and overlooking.
- We find that while interventions significantly improve an LLM's own performance, the collected data are not objectively informative or broadly transferable across different models.

Chapter 2

Related Work

Inductive Inference Some of the earliest work on reverse engineering appears under the label of *inductive inference* for "hypothesizing a general rule from examples" [3]. Classic instances of this problem include work on identifying the underlying structure of a finite-state automaton through observations of its input-output behavior [44, 45]. While this problem typically considers passive observations, seminal work on active learning focuses on analyzing the benefits of actively querying inputs to solicit the most-informative outputs from the unknown function of interest [32, 2, 50]. Naturally, the key distinction between these seminal works and ours is the attention towards LLMs and assessing their capacity for successfully identifying different types of black boxes from input-output examples.

Bayesian Optimal Experiment Design An adjacent line of work considers both the individual as well as sequential design of experiments which maximally yield information gain about an unknown parameter of interest [31, 13, 8, 17]; one may interpret these methods as studying a non-LLM-focused, Bayesian analogue of the reverse-engineering problem we formulate in the subsequent section, where a learner begins with a prior distribution over the black box in question and must maximally reduce epistemic uncertainty [15] with a given budget of experiments. To the extent that LLMs may implicitly engage with an underlying approximate posterior inference

scheme [65, 22, 68, 16, 41], one may tie the reverse-engineering capabilities studied in this work to this Bayesian optimal experiment design problem.

Reinforcement Learning The fundamentals of the reverse-engineering problem also connect with various ideas studied in the context of reinforcement learning (RL) [59]. Any model-based RL agent [57, 58, 6, 54] naturally engages with a particular instance of the reverse-engineering problem where the black-box function in question is the transition function and/or reward function of a Markov Decision Process (MDP) [4, 43]. The distinction explored in this work between a LLM that passively observes versus actively intervenes on the black box in question has a direct connection to the exploration challenge in RL, which has profound impact on an agent's ability to recover an accurate model of the world [61, 14, 55, 42]. The KWIK learning framework of [30] provides a theoretical analysis for reverse-engineering a MDP transition function when a learner must either confidently estimate the environment dynamics or say "I don't know" [62, 29, 48, 60, 1]. Finally, there is a connection between intervention for effective reverse-engineering and meta RL [33], with recent work showing that passive learning can be effective with LLMs once there is an effective exploration strategy capable of yielding high-quality observations [27]; naturally, the latter problem is precisely what we demonstrate interventions allow LLMs to solve for themselves.

LLMs for Automating Scientific Process With the rapid advances in LLMs, recent work has explored using them to automate different parts of the scientific process such as ideation [52], assistance [21], writing research papers [36], or emulating AI scientists in simulated environments [49]. These works utilize the abundant knowledge stored in the LLMs to directly tackle real-world complexity in science. However, the disadvantage of these approaches is the difficulty in dissolving all of the engineering choices. Our work instead focuses on using simple and controllable black boxes

to study the core capabilities of LLMs.

Understanding the Failure Modes in LLMs Recently, many works have examined the failure modes of formal reasoning in LLMs. People observed that LLMs can exhibit failure modes of both "overthinking" [9] and "underthinking" [64] when tackling mathematical problems and code generation [23, 53, 12, 53, 56, 7]. To understand LLM abilities beyond the formal reasoning tasks, many studies leverage insights and datasets from cognitive psychology [18, 5, 11, 67]. In particular, researchers have started to use cognitive science to explore the failure behaviors in LLM [26]. Using these methods, in my other papers, I have found that LLMs sometimes overestimate human rationality [34], CoT reasoning impairs performance[35]. In addition, [69] have found that LLMs exhibit inconsistencies in probability judgments. In designing our deterministic black box, we drew on datasets from cognitive psychology and scaled them to study the reverse-engineering abilities of LLMs.

Chapter 3

Reverse Engineering

3.1 Problem Formulation

We define a black box $f^* : \mathcal{X} \to \mathcal{Y}$ as a deterministic function that maps a stimulus $x \in \mathcal{X}$ to a response $y \in \mathcal{Y}$ through its internal dynamics. The **reverse-engineering** problem is for a model to infer the internals of a black box f^* (e.g., a set of rules, mathematical equations, or programs) from a sequence of stimulus-response pairs $\mathcal{O} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$. We consider two cases: **observation-only** and **observation-intervention**. In the observation-only scenario, all the stimuli are *randomly sampled* from $\mathcal{X}$ and the corresponding response $y_i = f^*(x_i)$ is generated by the black box to construct the observation set. A large language model $\mathcal{M}$ must generate a hypothesis $f = \mathcal{M}(\mathcal{O})$ without further interaction with the black box. This setting assesses the model's ability to perform inductive reasoning [3]. In the observation-intervention scenario, the LLM is first given a set of observations $\mathcal{O}$ obtained in the observation-only scenario and is instructed to interact with the black box in a multi-round fashion. In each round, the LLM chooses one of the following actions: (1) construct a new stimulus x_{N+1} to query the black box and obtain the response y_{N+1}, (2) construct a new stimulus-response pair (x_{N+1}, y'_{N+1}) and check it's validity using the black box ($\not\models [y'_{N+1} = f^*(x_{N+1})]$), or (3) stop and conclude with a hypothesis f about the black box. Before the LLM chooses to stop or reaches the

maximal number of rounds, the stimuli-response pairs obtained during intervention are appended to the observation set $\mathcal{O}$ for the next round.

3.2 Evaluation Protocol

A black box can be represented in multiple ways, rendering evaluation challenging. For example, two black boxes can be compared through their descriptions in natural language (descriptive evaluation) or whether they respond similarly to the same stimuli (functional evaluation). In this paper, we focus on the **descriptive evaluation** where the black box f^*_{NL} is expressed in natural language due to its communicative nature and their real-world use [10, 19]. The LLM's generated hypothesis f_{NL} is scored by an LLM judge against the black box on a $0-10$ scale based on the criteria of each black box type ($\texttt{score} = \texttt{LM-Judge}(f_{\text{NL}}, f^*_{\text{NL}})$).

For all the experiments, we use descriptive evaluation as the primary evaluation metric. We analyze functional evaluations in §5.3 where the hypothesis is tested on unseen stimuli-response pairs.

3.3 Black Box Types

Drawing on the literature on inductive inference in cognitive psychology, we selected the tasks commonly used to study human learning of complex logical reasoning to design our balck box systems, and scaling them up for evaluation with LLMs. These three distinct black-box types, programs, formal languages, and mathematical equations, simulate various problems encountered in scientific reverse-engineering scenarios. Each type includes multiple instances of varying complexity. We provide the design details in Appendix 8.

Program We use list-mapping programs [47] for the program black box. Each program implements a lambda expression (e.g., `(lambda(singleton(third $0)))`) in Python, where the stimulus is a list of integers and the response is an integer. The complexity level is determined based on the number of operations ranging from 1 to 12. Instances with fewer than 6 operations are considered low complexity, and the rest to be grouped as high complexity.

Formal Language The formal language black box is defined by a production rule over a set of symbols. For example, language A^nB^n can generate a sequence of A's followed by B's of the same length. The black box allows the LLM to intervene by validating if a string is allowable under the rule. We take 46 languages each as a black box instance from [66, 40]. We count regular language instances as low complexity black boxes, whereas context-free and context-sensitive languages are of higher complexity.

Math Equation We use the Constant Elasticity of Substitution (CES) formulation from economics [17] as the math equation black box. The utility $U = (\sum_i a_i x_i^r)^{\frac{1}{r}}$ is given by the weights a_i, the ratio r, and the good quantities x_i. The LLM can query the black box with two lists of good quantities as stimulus and obtain the response indicating which list has is preferred based on higher utility. We group the black box instances involving 2 or 3 goods as lower complexity and those with 5 or 6 goods as higher complexity.

Chapter 4

Experiments: How Well LLM Do Reverse-Engineering?

4.1 Experiments Setup

We use different versions of GPT-4o [24] as the LLM reverse engineer (*gpt-4o-2024-08-06*) and judge LLM (*gpt-4o-2024-05-13*) in our experiments. To minimize the variance of our experiments, we set the temperature of both the LLM reverse engineer and the judge LLM to 0.0. For Observation-only, we run $nobs = 2, 5, 10, 20, 50, 100$ with 5 different seeds to observe two types of failure behaviors, "overcomplication" and "overlooking" in reverse engineering. In Intervention with observation experiments, we provide 10 observations and run $nintv = 5, 10, 20, 50$ with 3 different seeds for each black box instance. During the interventions, we allow LLM reverse engineer to either query the black box instance to get more information or test its own hypothesis. We provide prompt examples for each type of black box in the Appendix 8.4.

4.2 Inferring Hypotheses from Observations

Bayesian Model as the "Optimal" Reference We employ Bayesian agents as a baseline to rigorously assess the reverse-engineering capabilities of LLMs. Unlike LLMs, Bayesian agents explicitly perform probabilistic inference within a well-defined

symbolic hypothesis space, utilizing Bayes' rule to systematically update posterior beliefs and identify the most explanatory programs given observed data. Under the critical assumption that the true underlying rule resides within this hypothesis space, Bayesian models serve as an optimal reference under our settings. We hypothesize that, given only passive observational data, LLMs cannot effectively leverage the available information due to the strong prior knowledge, consequently resulting in significantly inferior performance compared to the Bayesian optimal standard. We posit that enabling LLMs to actively intervene, gather higher-quality, more informative data can substantially mitigate this performance gap. For each of the three black-box systems evaluated, we replicated the Bayesian search settings described in their original studies, adapting them to align with our experimental setup by providing the observations generated from our black-box systems and LLM-driven intervention data. To ensure rigorous comparability, we applied identical evaluation methodologies to both Bayesian agents and LLMs. We show more details of the Bayesian model in Appendix 8.2

4.3 How Well Can LLM Use Observations?

We first establish an optimal reference by providing observations from our black-box systems to a Bayesian model conducting search within the given observations space. As shown by the green curves in Figure 4.1, the Bayesian results consistently improve as the number of observations increases across three tasks. However, we observe that LLM performance peaks at 10 observations, after which the LLM struggles to obtain additional informative knowledge by utilizing more observations, causing performance to plateau or even degrade. With increasing numbers of observations, this degradation gradually widens the gap between the LLM and Bayesian reference. We calculated repeated measures ANOVAs for each black box type and found signifi-

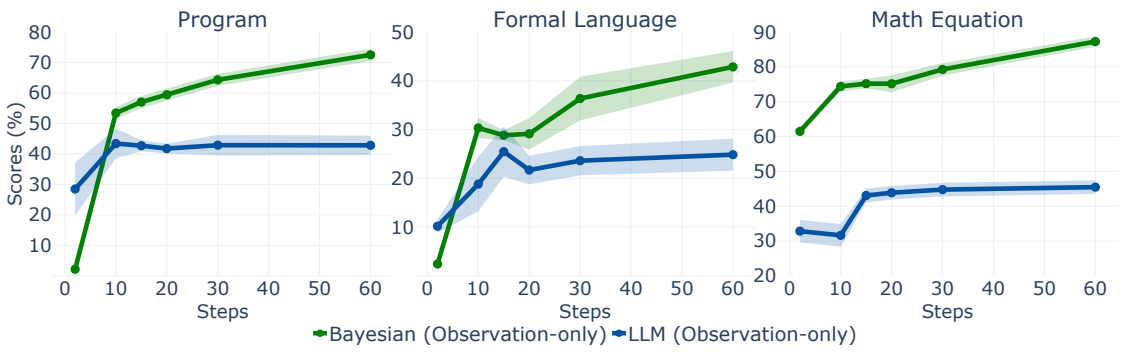

Figure 4.1: Observation-only results for GPT-4o across three black box types. We compare the LLM performance to Bayesian reference. For Program and Formal Language, we use LLM to judge whether the reverse rules can describe each black box instance. For math equation, to better compare with Bayesian model, we directly calculate the flipped RMSE.

cant Model × Step interactions for Programs ($F(5,10)=51.91$, $p < 0.001$), Formal Language ($F(5,10)=11.83$, $p=0.001$), and Math Equation ($F(5,10)=8.69$, $p=0.002$), showing that the Bayesian method increasingly outperformed LLM as the number of observations increased. We provide the ANOVA calculation details in the Appendix 8.3.1.

4.4 Effective Intervention Is Crucial for Solving Reverse Engineering Problems

In Figure 4.2, we first compare performance when the LLM is allowed to intervene versus when it relies solely on observations. We observe that the LLM acquires valuable knowledge during the intervention phase, enabling it to better understand the dynamics of the black box. After LLM begins interventions, performance consistently improves in all three black-box systems. To evaluate the quality of these intervention data, we provided LLM-generated data to the Bayesian model as the passive yoked data [38, 39]. We found that these data are not objectively optimal but rather specifically beneficial for the LLM itself. This finding indicates that ac-

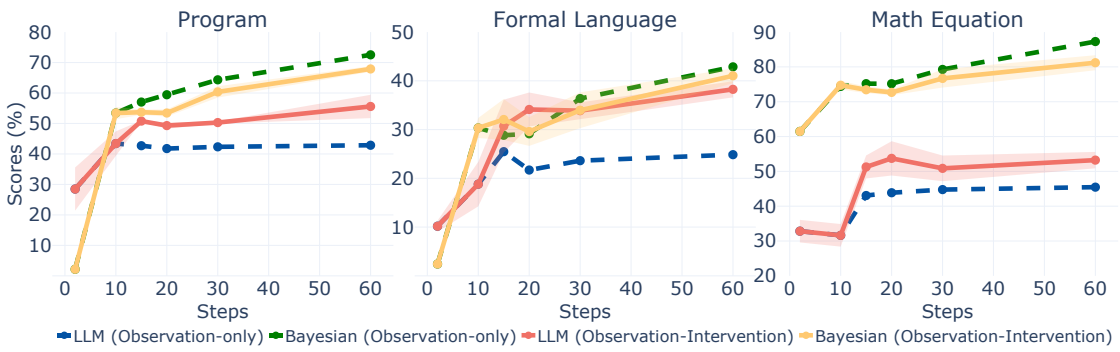

Figure 4.2: Observation-intervention results for GPT-4o across the three black-box types. We compare the performance of LLM interventions against the Bayesian reference under two conditions: using only observations, and using interventions derived from LLM queries.

tively selecting observations to test LLM's own hypotheses facilitates learning more effectively than passive observation alone. To understand the significance of this gap, we report the ANOVA scores in each black box type, Programs (F (5,10) = 23.6822, $p < 0.001$), Formal language (F (5,10) = 7.8601, p = 0.003) and Math Equation (F(5,10)=14.6845, $p < 0.001$). The results of ANOVA test indicate that the Bayesian models outperformed the LLM more and more as the number of data increased.

4.5 Two Common Failure Modes: Overcomplication & Overlooking

Upon delving deeper into black-box complexity, we identify two common failure modes exhibited by GPT4-o in reverse-engineering tasks: (1) overcomplication, where the LLM overinterprets the provided data before formulating a hypothesis about the black box, and (2) overlooking, where the LLM becomes "lazy" in leveraging available information and presents a hypothesis without sufficient reasoning. As the examples shown in Figure 1.1, we categorize the failure behaviors based on how the LLM effectively utilizes its observations. Through extensive experiments across different

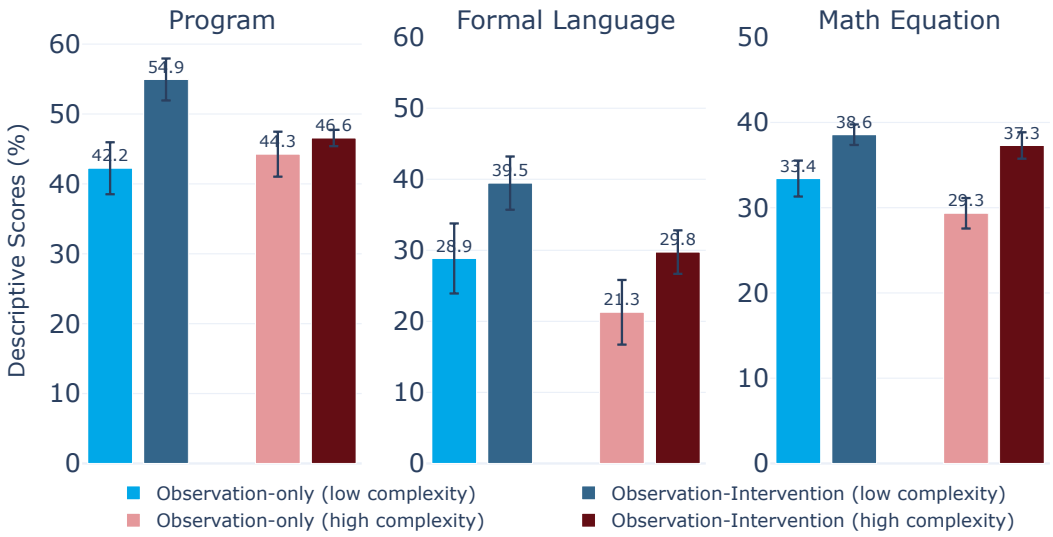

Figure 4.3: Descriptive scores for two different complexity levels. Averaged across three seeds for each of the three black-box types. For math equation, we use descriptive evaluation instead of RMSE only.

seeds of the black boxes, we observe that within the same black box type, the effect of the active intervention can vary depending on the complexity of the tasks. In Figure 4.3, we see that for program, the intervention yields a 12.9% improvement on low-complexity instances while only a 2.2% improvement on high-complexity instances. A similar pattern emerges in formal languages, with a 10.6% gain on low-complexity instances versus 8.5% on high-complexity instances. In contrast, for math equations, the intervention boosts performance by 8.0% more on high-complexity tasks compared to low-complexity tasks.

4.6 Effective Intervention via Iterative Hypothesis Refinement

Case Study. Under the observation and intervention setting, the LLM actively tests its hypotheses during the reverse-engineering process. As illustrated in Figure

Black box instances: (lambda (singleton (third $0)))
Observations:
Input: [46, 21] Output: invalid input
Input: [32, 77, 27, 77, 4] Output: [27]
Input: [87, 20, 55, 81, 50, 92, 65, 47, 69, 56] Output: [55]

… (more observations)

Input: [45, 46, 57, 20, 96, 51, 91, 94, 59, 83] Output: [57]

Interventions:
Before gathering more data, I think: To determine the inner workings of the black box, let's analyze the observations from the previous turns:
Invalid Input: The black box returns "invalid input" for inputs with fewer than three integers.
Output Pattern: For inputs with three or more integers, the black box returns a single integer.
Hypothesis: The black box seems to return the third integer from the input list. This hypothesis is consistent with all the outputs.
To further confirm this hypothesis, I will perform a test with a new input list where the third integer is distinct and easily identifiable.
```test
[10, 20, 30, 40, 50]
[30]
```
Input: [10, 20, 30, 40, 50] Output: [30] is correct
… (more interventions)

Figure 4.4: Example of how LLM intervene with list-mapping program black box. Before interacting with the black box, LLM reasons about past observation patterns and comes up with the internal mental model of the black box to test.

4.4, in a list-mapping black-box scenario, GPT-4o strategically designs its next query to assess an internal hypothesis, drawing from previously observed valid cases in which the output consistently corresponds to the third integer of the input list. This contrasts with the Observation only scenario, where the model is limited to passively inferring patterns from existing data. By intervening actively, the LLM iteratively tests and refines its hypotheses upon encountering failures, enabling self-verification and reducing uncertainty about the black-box system.

Different Intervention Approaches. We investigate how different reasoning approaches impact the effectiveness of the intervention. Specifically, we compare a zero shot intervention baseline with three structured reasoning conditions: hypoth-

esis refinement, hypothesis description, and function-structured hypothesis. In our experiments, we introduce a reasoning step every five intervention rounds, with interventions lacking explicit reasoning serving as the baseline. As shown in Table 4.1, our findings suggest that effective intervention typically requires the LLM to carefully analyze past observations and strategically plan subsequent steps to acquire more informative knowledge from the black box. Interestingly, while structured reasoning generally improves performance in formal reasoning tasks [28, 37], it did not produce additional improvements in the reverse-engineering context. This indicates that the reverse-engineering abilities of an LLM may differ fundamentally from its formal reasoning capabilities.

Black box	Complexity	Intv	Hypothesis-refinement Intv	Descriptive Intv	Functional Intv
Program	low	43.6	**54.9**	49.8	15.9
	high	43.1	**46.6**	45.3	22.5
Formal Language	low	30.6	39.5	**40.4**	30.6
	high	17.6	**29.8**	16.7	14.9
Math Equation	low	36.5	**38.6**	36.8	38.4
	high	33.0	37.3	40.8	**41.4**

Table 4.1: Different reasoning approaches during the LLM intervention. Intv refers to the intervention

Chapter 5

Analysis

5.1 Reverse Engineering Abilities Across Different Categories of LLMs

In Figure 5.1, we report the results of observation-only and observation + intervention across different LLMs: Llama-3.3-70B-Instruct, Claude 3.5 Sonnet, and DeepSeek R1. Across nearly all black-box types and models, actively intervening with iteratively refining hypotheses consistently enhances models' understanding of the underlying black-box dynamics. In particular, we show that DeepSeek R1, utilizing Long CoT reasoning, has the potential to continuously extract informative knowledge even from passive learning scenarios. This detailed and long reasoning allows the model to explore various potential hypotheses. However, despite these advantages, DeepSeek R1 does not significantly outperform models without explicit reasoning (e.g., GPT-4o, Llama-3.3-70B-Instruct, Claude 3.5 Sonnet) in reverse-engineering tasks. This observation highlights the inherent limitations of current reasoning steps for existing LLMs.

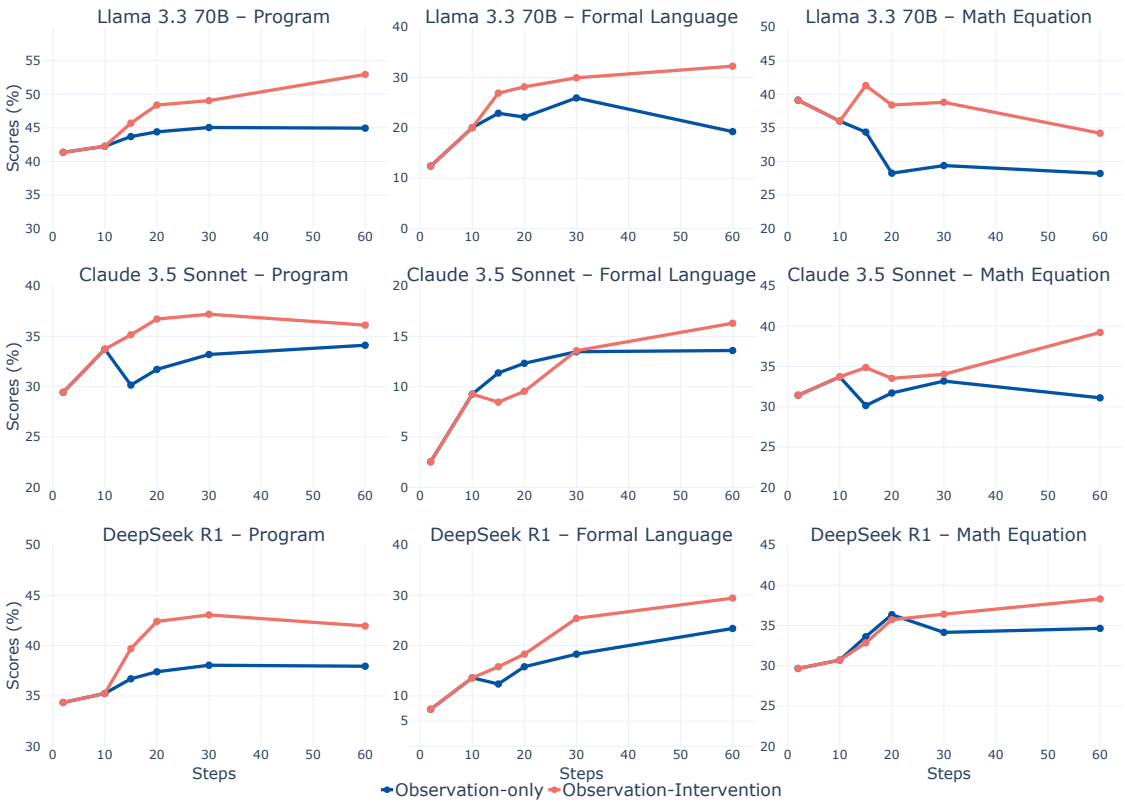

Figure 5.1: Results of reverse engineering abilities across different categories of LLMs. We report Llama-3.3-70B-Instruct, Claude 3.5 Sonnet and Deepseek R1 on three black box types.

5.2 Transferability of intervention data

Transfer to LLM itself. We follow the passive-yoked design in [38, 39] to study whether intervention data collected by an LLM during active learning remains effective when reused as passive observational data. In Figure 5.2, we compare three scenarios of GPT-4o over 3 seeds: **Observation-only**, **Observation-Intervention**, and **Intervention-Transfer**, where the model passively observes the intervention data collected through active interaction. Across the three black-box systems, intervention-transfer yields a lower performance than the actively collected Observation-Intervention setting. This suggests that active learning is more effective than passive-yoked learning, in part since it allows the LLM to dynamically refine its hypothesis in response

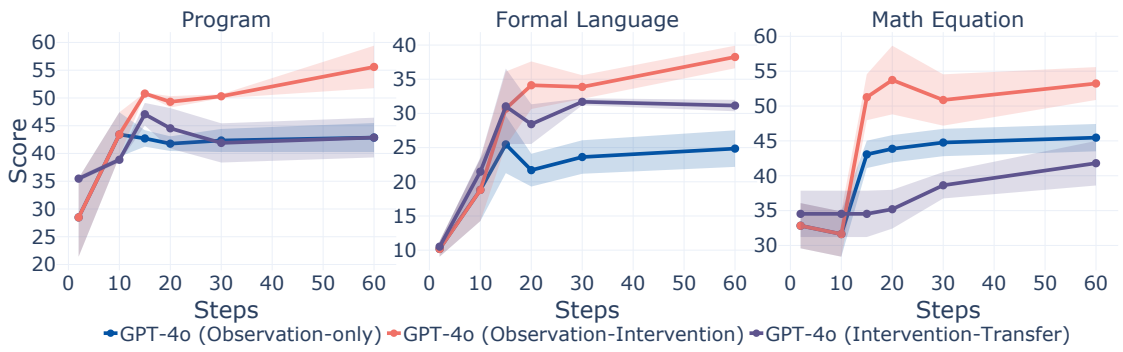

Figure 5.2: Transfer to itself (active learning to passive learning). O+I+T: LLM takes the intervention data from others as the passive observations.

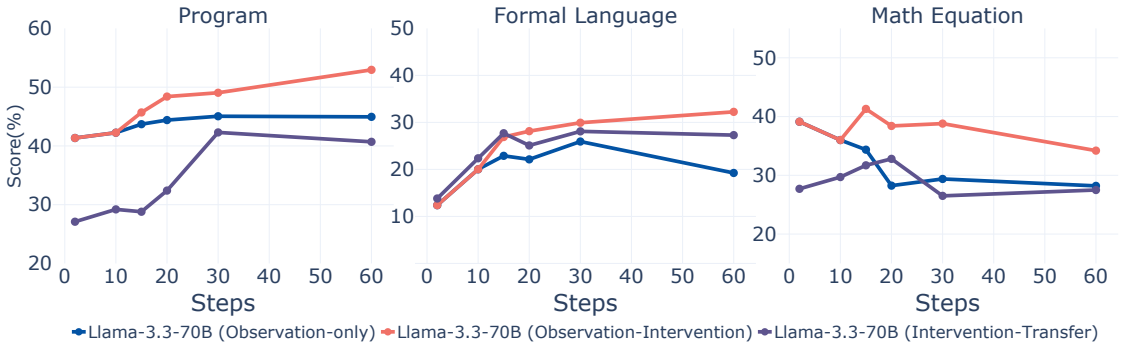

Figure 5.3: Transfer to another LM: GPT-4o interventions as the passive yoked observations to Llama-3.3-70B-Instruct

to its own interventions.

Transfer to different LLM. Next, we examine whether intervention data actively collected by an LLM (GPT-4o) can effectively transfer and benefit another model (Llama-3.3-70B-Instruct). Following a similar passive-yoked paradigm, we also compare three scenarios for Llama-3.3-70B-Instruct: **Observation-only**, **Observation-Intervention**, and **Intervention-Transfer**, where the latter utilizes interventions actively selected by GPT-4o. As shown in Figure 5.3, the Intervention-Transfer scenario performs comparably or slightly better than the Observation-only baseline, yet it consistently underperforms compared to Llama's own actively collected interventions (Observation-Intervention). This indicates that intervention data collected actively

by GPT-4o holds some general value, but its effectiveness is reduced when transferred across different LLMs, further highlighting that the informational advantage of active learning for existing LLM is model-specific.

5.3 How to Evaluate Reverse-Engineering?

Black box	Complexity	Descriptive Scores	Implicit Functional	Explicit Functional
Program	low	42.2	**62.7**	**55.8**
	high	**44.3**	54.1	52.1
Formal Language	low	17.6	**28.9**	**28.8**
	high	**27.1**	9.2	10.9
Math Equations	low	**36.5**	25.2	**51.3**
	high	33.0	**47.6**	47.3

Table 5.1: Different trends between descriptive evaluation and functional evaluation for Observation-only.

We examine two types of functional evaluations where the LLM is assessed based on how well it can fit the unseen data generated by the black box f^*. **Explicit functional evaluation** evaluates whether the LLM can implement the black box that is functionally indistinguishable from the black box. The LLM is instructed to generate an executable implementation of f which shares the same interface as f^*. The accuracy is computed as the percentage of the test responses that the LLM's implementation can correctly predict ($\texttt{Acc} = \frac{1}{M}\sum_{i=1}^{M} \mathbb{1}[f(x_i^{\text{test}}) = y_i^{\text{test}}]$). **Implicit Functional Evaluation** evaluates the same thing as EFE without generating the black-box implementation and uses the LLM itself to predict the response. The accuracy is computed by LLM directly predicting the response conditioned on the test stimuli and the past observations ($\texttt{Acc} = \frac{1}{M}\sum_{i=1}^{M} \mathbb{I}[y_i^{\text{test}} = \mathcal{M}(x_i^{\text{test}}, \mathcal{O})]$). This is effectively the canonical in-context learning setting, where the function is implicitly learned in the model activations.

The objective of traditional machine learning evaluation usually focuses on the ability of a model to generalize unseen examples [25]. To evaluate the generalization, we ask the LLM to both implicitly and explicitly generate a mental model of the black box as a generative model and evaluate its performance on unseen examples. We used the original samples from the black box of the list mapping program as test cases [47] and ensured that none of these input–output pairs were included in the observations. For formal language and math equation, we use our deterministic black box to produce 20-25 test cases per black box instance. In Table 5.1, we observe that both functional evaluations align with general intuition, where models perform better in low-complexity tasks. However, function evaluations fail to reveal clear trends for the two types of failure behavior in reverse engineering.

Chapter 6

Future Directions

6.1 Training LLMs

We show that current LLMs struggle to effectively utilize observed data and fail to consistently collect objectively informative observations. One natural direction for future research is to enhance LLM capabilities in two complementary aspects: (1) improving their ability to interpret and leverage available observations more accurately, and (2) training them to proactively select or generate more informative observational data. Advancements in these two aspects could significantly improve the performance of LLM in reverse engineering tasks.

6.2 Controlled Experiments vs Realistic Simulations

An inherent limitation of this thesis lies in the gap between controlled experimental settings and realistic, complex simulations. Although our paper systematically identifies and analyzes LLM limitations within controlled scenarios, future work should extend these analyses to more realistic simulations. Bridging this gap will enable a deeper and more general understanding of how LLMs perform under practical reverse

engineering conditions.

6.3 Multi-Agent Collaboration

It is important to explore whether LLMs can generate data that are not only informative for themselves, but also objectively transferable across different contexts or collaborative settings. This transferability is particularly critical in multi-agent systems, where successful reverse engineering depends on effective sharing of useful information. Investigating the ways to overcome existing barriers to transfer will be key to improve the performance of reverse-engineering in the multi-agent collaboration settings.

Chapter 7

Conclusion

In this thesis, we investigated the reverse-engineering capabilities of LLMs, specifically focusing on their ability to infer the underlying mechanisms from observational and interventional data. We demonstrated that when limited to passive observations, LLMs struggle significantly to reverse-engineer transition functions due to their inherent biases and inability to effectively leverage observational information. Introducing active interventions improved their performance by allowing models to refine their hypotheses through targeted exploration. However, extensive analysis revealed that even with interventions, LLMs still fall short of optimal Bayesian inference, frequently exhibiting issues such as overcomplication and overlooking crucial information. Furthermore, our analysis revealed that the interventions collected by LLMs were primarily beneficial to the models themselves, rather than being objectively informative or transferable to other models. Our findings highlight that future research should prioritize improving the transferability and objectivity of the data collected by LLMs, reducing their inherent biases, enhancing their capacity to objectively interpret data, and ultimately advancing their fundamental reverse-engineering capabilities.

Bibliography

[1] Jacob Abernethy, Kareem Amin, Michael Kearns, and Moez Draief. Large-Scale Bandit Problems and KWIK Learning. In *International Conference on Machine Learning*, pages 588–596. PMLR, 2013.

[2] Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2:319–342, 1988.

[3] Dana Angluin and Carl H Smith. Inductive Inference: Theory and Methods. *ACM Computing Surveys (CSUR)*, 15(3):237–269, 1983.

[4] Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

[5] Marcel Binz and Eric Schulz. Using cognitive psychology to understand GPT-3. *Proceedings of the National Academy of Sciences*, 120(6):e2218523120, 2023.

[6] Ronen I Brafman and Moshe Tennenholtz. R-MAX – A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[7] Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

[8] Kathryn Chaloner and Isabella Verdinelli. Bayesian Experimental Design: A Review. *Statistical Science*, pages 273–304, 1995.

[9] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do Not Think that Much for 2+ 3=? On the Overthinking of o1-like LLMs. *arXiv preprint arXiv:2412.21187*, 2024.

[10] Sahil Chopra, Michael Henry Tessler, and Noah D Goodman. The first crank of the cultural ratchet: Learning and transmitting concepts through language. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 41, 2019.

[11] Julian Coda-Forno, Marcel Binz, Jane X Wang, and Eric Schulz. Cogbench: A Large Language Model Walks into A Psychology Lab. *arXiv preprint arXiv:2402.18225*, 2024.

[12] Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, et al. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.

[13] MH DeGroot. Uncertainty, Information, and Sequential Experiments. *The Annals of Mathematical Statistics*, 33(2):404–419, 1962.

[14] Marc Deisenroth and Carl E Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 465–472, 2011.

[15] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or Epistemic? Does it Matter? *Structural Safety*, 31(2):105–112, 2009.

[16] Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models Bayesian? A martingale perspective. *arXiv preprint arXiv:2406.00793*, 2024.

[17] Adam Foster, Martin Jankowiak, Elias Bingham, Paul Horsfall, Yee Whye Teh, Thomas Rainforth, and Noah Goodman. Variational Bayesian optimal experimental design. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] Michael C Frank. Baby steps in evaluating the capacities of large language models. *Nature Reviews Psychology*, 2(8):451–452, 2023.

[19] Kanishk Gandhi, Michael Y Li, Lyle Goodyear, Louise Li, Aditi Bhaskar, Mohammed Zaman, and Noah D Goodman. BoxingGym: Benchmarking progress in automated experimental design and model discovery. *arXiv preprint arXiv:2501.01540*, 2025.

[20] Yolanda Gil, Mark Greaves, James Hendler, and Haym Hirsh. Amplify scientific discovery with artificial intelligence. *Science*, 346(6206):171–172, 2014.

[21] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, Artiom Myaskovsky, Felix Weissenberger, Keran Rong, Ryutaro Tanno, Khaled Saab, Dan Popovici, Jacob Blum, Fan Zhang, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Pushmeet Kohli, Yossi Matias, Andrew Carroll, Kavita Kulkarni, Nenad Tomasev, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R D Costa, José R Penadés, Gary Peltz, Yunhan Xu, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Towards an AI co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

[22] Thomas L Griffiths, Jian-Qiao Zhu, Erin Grant, and R Thomas McCoy. Bayes in the Age of Intelligent Machines. *Current Directions in Psychological Science*, 33(5):283–291, 2024.

[23] Yancheng He, Shilong Li, Jiaheng Liu, Weixun Wang, Xingyuan Bu, Ge Zhang, Zhongyuan Peng, Zhaoxiang Zhang, Wenbo Su, and Bo Zheng. Can large language models detect errors in long chain-of-thought reasoning? *arXiv preprint arXiv:2502.19361*, 2025.

[24] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[25] Katie Kang, Amrith Setlur, Dibya Ghosh, Jacob Steinhardt, Claire Tomlin, Sergey Levine, and Aviral Kumar. What do learning dynamics reveal about generalization in llm reasoning? *arXiv preprint arXiv:2411.07681*, 2024.

[26] Alexander Ku, Declan Campbell, Xuechunzi Bai, Jiayi Geng, Ryan Liu, Raja Marjieh, R Thomas McCoy, Andrew Nam, Ilia Sucholutsky, Veniamin Veselovsky, et al. Using the tools of cognitive science to understand large language models at different levels of analysis. *arXiv preprint arXiv:2503.13401*, 2025.

[27] Andrew Kyle Lampinen, Stephanie C Y Chan, Ishita Dasgupta, Andrew J Nam, and Jane X Wang. Passive learning of active causal strategies in agents and language models. *Advances in Neural Information Processing Systems*, 2023.

[28] Jia Li, Ge Li, Yongmin Li, and Zhi Jin. Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–23, 2025.

[29] Lihong Li and Michael L Littman. Reducing Reinforcement Learning to KWIK Online Regression. *Annals of Mathematics and Artificial Intelligence*, 58:217–237, 2010.

[30] Lihong Li, Michael L Littman, and Thomas J Walsh. Knows What It Knows: A Framework for Self-Aware Learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 568–575, 2008.

[31] Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

[32] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[33] Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling Exploration and Exploitation for Meta-Reinforcement Learning Without Sacrifices. In *International Conference on Machine Learning*, pages 6925–6935. PMLR, 2021.

[34] Ryan Liu, Jiayi Geng, Joshua C Peterson, Ilia Sucholutsky, and Thomas L Griffiths. Large language models assume people are more rational than we really are. *arXiv preprint arXiv:2406.17055*, 2024.

[35] Ryan Liu, Jiayi Geng, Addison J Wu, Ilia Sucholutsky, Tania Lombrozo, and Thomas L Griffiths. Mind your step (by step): Chain-of-thought can reduce performance on tasks where thinking makes humans worse. *arXiv preprint arXiv:2410.21333*, 2024.

[36] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[37] Yijia Luo, Yulin Song, Xingyao Zhang, Jiaheng Liu, Weixun Wang, GengRu Chen, Wenbo Su, and Bo Zheng. Deconstructing long chain-of-thought: A structured reasoning optimization framework for long cot distillation. *arXiv preprint arXiv:2503.16385*, 2025.

[38] Doug Markant and Todd Gureckis. Category learning through active sampling. In *Proceedings of the annual meeting of the cognitive science society*, volume 32, 2010.

[39] Douglas B Markant and Todd M Gureckis. Is it better to select or to receive? learning via active and passive hypothesis testing. *Journal of Experimental Psychology: General*, 143(1):94, 2014.

[40] R Thomas McCoy and Thomas L Griffiths. Modeling rapid language learning by distilling bayesian priors into artificial neural networks. *arXiv preprint arXiv:2305.14701*, 2023.

[41] R Thomas McCoy, Shunyu Yao, Dan Friedman, Mathew D Hardy, and Thomas L Griffiths. Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *Proceedings of the National Academy of Sciences*, 121(41):e2322420121, 2024.

[42] Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) Efficient Reinforcement Learning via Posterior Sampling. *Advances in Neural Information Processing Systems*, 26:3003–3011, 2013.

[43] Martin L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[44] Ronald L Rivest and Robert E Schapire. Diversity-Based Inference of Finite Automata. In *28th Annual Symposium on Foundations of Computer Science (SFCS 1987)*, pages 78–87. IEEE, 1987.

[45] Ronald L Rivest and Robert E Schapire. Inference of Finite Automata Using Homing Sequences. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 411–420, 1989.

[46] Milena Rmus, Akshay K. Jagadish, Marvin Mathony, Tobias Ludwig, and Eric Schulz. Towards automation of cognitive modeling using large language models. *arXiv preprint arXiv:2502.00879*, 2025.

[47] 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.

[48] Amin Sayedi, Morteza Zadimoghaddam, and Avrim Blum. Trading off Mistakes and Don't-Know Predictions. *Advances in Neural Information Processing Systems*, 23, 2010.

[49] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*, 2025.

[50] Burr Settles. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[51] Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. Llm-sr: Scientific equation discovery via programming with large language models. *International Conference on Learning Representations*, 2025.

[52] Chenglei Si, Diyi Yang, and Tatsunori Hashimoto. Can LLMs generate novel research ideas? A large-scale human study with 100+ NLP researchers. *arXiv preprint arXiv:2409.04109*, 2024.

[53] Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Dur-

rett. To CoT or not to CoT? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*, 2024.

[54] Alexander L Strehl and Michael L Littman. An Analysis of Model-based interval estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[55] Malcolm JA Strens. A Bayesian framework for reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 943–950, 2000.

[56] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.

[57] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224, 1990.

[58] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[59] Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[60] István Szita and Csaba Szepesvári. Agnostic KWIK learning and Efficient Approximate Reinforcement Learning. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 739–772. JMLR Workshop and Conference Proceedings, 2011.

[61] Sebastian B Thrun and Knut Möller. Active exploration in dynamic environments. *Advances in Neural Information Processing Systems*, 4, 1991.

[62] Thomas J Walsh, István Szita, Carlos Diuk, and Michael L Littman. Exploring Compact Reinforcement-Learning Representations with Linear Regression. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 591–598, 2009.

[63] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific Discovery in the Age of Artificial Intelligence. *Nature*, 620(7972):47–60, 2023.

[64] Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, et al. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*, 2025.

[65] 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*, 2021.

[66] Yuan Yang and Steven T. Piantadosi. One model for the learning of language. *Proceedings of the National Academy of Sciences*, 119(5):e2021865119, 2022.

[67] Lance Ying, Katherine M Collins, Lionel Wong, Ilia Sucholutsky, Ryan Liu, Adrian Weller, Tianmin Shu, Thomas L Griffiths, and Joshua B Tenenbaum. On benchmarking human-like intelligence in machines. *arXiv preprint arXiv:2502.20502*, 2025.

[68] Jian-Qiao Zhu and Thomas L Griffiths. Eliciting the Priors of Large Language

Models using Iterated In-Context Learning. *arXiv preprint arXiv:2406.01860*, 2024.

[69] Jian-Qiao Zhu and Thomas L Griffiths. Incoherent probability judgments in large language models. *arXiv preprint arXiv:2401.16646*, 2024.

Chapter 8

Appendix

8.1 Black Box Designs

Program We used 100 list-mapping program instances from [47] to design the Program black box API. Each black box instance represents as a symbolic program defined in a domain-specific language (DSL). We implemented an interpreter pipeline that parses DSL expressions into abstract syntax trees and compiles them into executable Python code.

Each black box supports two modes: `observation` and `intervention`. In the `observation` mode, the black box takes a random input list and returns the output produced by the underlying symbolic program, generating paired observational data:

$$\text{input list} \rightarrow \text{program execution} \rightarrow \text{output list}$$

In the `intervention` mode, an LLM queries an input or explicitly specifies an input-output pair. The black box generates the output list or evaluates whether the

given output matches the internally computed output and provides clear feedback:

$$\text{Feedback} = \begin{cases} \text{"output} \Rightarrow \text{Correct"}, & \text{if the provided output matches the program output,} \\ \text{"output} \Rightarrow \text{Incorrect"}, & \text{otherwise.} \end{cases}$$

Formal Language We followed [66, 40] to implement a collection of 46 formal language instances to construct our formal language black-box, each instance being capable of generating strings according to specific symbolic rules (e.g. A^nB^n). Each black box instance behaves as an API from a generative model, operating in two modes: `observation` and `intervention`.

In the `observation` mode, the black box randomly produces valid strings from its underlying rule, explicitly labeling each as generated output, for example:

"AAAABBBB" is generated by the black box.

In the `intervention` mode, an LLM submits a specific string query to the black box, which evaluates whether the string complies with its rule. The black box responds clearly, indicating either acceptance or rejection:

$$\text{Response} = \begin{cases} \text{"[string] is generated by the black box"}, & \text{if the strings compile with the ru} \\ \text{"[string] cannot be generated by the black box"}, & \text{otherwise.} \end{cases}$$

To avoid generating infinite strings, we imposed a maximum character length of 64 for all single characters generated by the black box.

Math Equation For the math equation, we implemented the CES utility model as the black box, designing it as a generative model capable of generating observational data or responding to queries from an LLM. The utility function of CES is

mathematically defined as:

$$U = \left(\sum_i a_i x_i^r \right)^{\frac{1}{r}},$$

where the weights a_i satisfy the constraint $\sum_i a_i = 1$, the parameter r controls the substitution elasticity, and x_i represents the quantities of goods in a basket.

CES black box also provides two operational modes: `observation` and `intervention`. In the `observation` mode, the black box randomly samples two baskets (each a list of good quantities) and computes their utilities using the CES formulation. It then returns the preference outcome indicating which basket is preferred based on higher utility:

$$\text{Preference} = \begin{cases} \text{Basket1}, & U(\text{Basket1}) > U(\text{Basket2}), \\ \text{Basket2}, & U(\text{Basket1}) < U(\text{Basket2}), \\ \text{equal utility}, & U(\text{Basket1}) = U(\text{Basket2}). \end{cases}$$

In the `intervention` mode, an external model explicitly queries the black box by specifying two baskets. In addition, the external model can also provide an estimated preference. The CES black box internally evaluates the utilities based on the specified baskets and returns the actual preference outcome or feedback indicating whether the provided estimate was "correct" or "incorrect".

8.2 Bayesian models as the 'Optimal' Reference

Program We used the Bayesian inference approach from [47] to establish an optimal reference for list-mapping program black box. Specifically, we utilized their MetaProgram Learner, which performs Bayesian inference over symbolic metaprograms that generate target programs from observed data.

Given observational data D, consisting of input-output pairs generated by the symbolic programs, the MPL computes the posterior distribution over candidate hy-

potheses (metaprograms) H according to Bayes' theorem:

$$P(H \mid D) \propto P(D \mid H) \cdot P(H).$$

The prior distribution $P(H)$ integrates two complementary sources of simplicity bias: the metaprogram prior $P_{\mathcal{M}}(H)$ and the induced program prior $P_{\mathcal{P}}(\widetilde{H})$. This combined prior is defined as:

$$P(H) \propto \exp\left(\frac{\ln P_{\mathcal{M}}(H) + \ln P_{\mathcal{P}}(\widetilde{H})}{2}\right),$$

where $\widetilde{H}$ denotes the program compiled from the metaprogram H.

The likelihood $P(D \mid H)$ measures the consistency of a metaprogram H with the provided observational data, incorporating a noise model to accommodate minor discrepancies between model predictions and observations.

Formal Language We adopted the Bayesian inference approach from [66] as an optimal reference model to determine the theoretical upper bound on the learnability of formal language rules from the observations generated by our black boxes or from the interventions queried by LLM. Specifically, we provided strings generated by our formal language black boxes as observational data to the Bayesian model, which then inferred the underlying symbolic grammar rules.

The Bayesian inference framework defines the posterior distribution over candidate hypotheses conditioned on observed data using Bayes' theorem:

$$P(H \mid D) \propto P(D \mid H)\, P(H),$$

where H represents a candidate hypothesis (grammar or probabilistic program), D represents the observed string data generated by the black box, $P(H)$ represents

the prior probability reflecting initial beliefs about the simplicity and plausibility of hypotheses, and $P(D \mid H)$ denotes the likelihood of observing data D given hypothesis H.

The Bayesian model uses a structured prior $P(H)$, assigning higher probabilities to simpler, more concise grammars or symbolic programs. As observational data increases, Bayesian updating systematically refines prior beliefs into posterior distributions, enhancing the probability assigned to grammars that best explain the data. Formally, each new observed string updates the posterior, shifting probability mass toward hypotheses consistent with the cumulative dataset. By leveraging this Bayesian inference mechanism, we quantify the upper bound of the learnability of the observations, thus providing a rigorous baseline to evaluate LLM's effectiveness in utilizing the same observational data.

Math Equation To infer the parameters of the CES utility model from the observations provided, we followed [17] to employ a Bayesian inference approach explicitly conditioned on these observations. Bayesian inference integrates observed data with prior beliefs, updating these beliefs into posterior distributions to progressively improve parameter estimates. Initially, we set prior distributions for model parameters:

$$\begin{aligned} \rho &\sim \text{Beta}(\rho_0, \rho_1), \\ \alpha &\sim \text{Dirichlet}(\alpha_{\text{conc}}), \\ \text{slope} &\sim \text{LogNormal}(\text{slope}_\mu, \text{slope}_\sigma). \end{aligned}$$

Given pairs of consumption bundles (d_1, d_2) and the corresponding observed user preferences y, the Bayesian framework models these preferences probabilistically through a censored sigmoid-normal likelihood:

$$y \sim \text{CensoredSigmoidNormal}\left(\text{slope} \cdot (U(d_1) - U(d_2)),\ \text{slope} \cdot \text{obs_sd} \cdot (1 + \|d_1 - d_2\|_2)\right),$$

where $U(d_1) - U(d_2)$ indicates the utility difference between the two bundles.

The posterior distributions are then updated through Bayes' theorem, explicitly integrating the observational data:

$$p(\rho, \alpha, \text{slope} \mid y, d) \propto p(y \mid \rho, \alpha, \text{slope}, d)\, p(\rho, \alpha, \text{slope}),$$

where $p(\rho, \alpha, \text{slope})$ represents prior distributions and $p(y|\rho, \alpha, \text{slope}, d)$ represents the likelihood function given the observations.

Parameter estimation was conducted via variational inference, iteratively optimizing the evidence lower bound (ELBO) defined as:

$$\text{ELBO}(\phi) = \mathbb{E}_{q_\phi}\left[\log p(y \mid \rho, \alpha, \text{slope}, d)\right] - D_{\text{KL}}\left(q_\phi(\rho, \alpha, \text{slope}) \parallel p(\rho, \alpha, \text{slope})\right),$$

where q_ϕ denotes the variational posterior distribution used to approximate the true posterior distribution.

Thus, as more observational data are incorporated, Bayesian inference continuously updates prior beliefs to posterior distributions, systematically refining parameter estimates towards their true underlying values.

8.3 Statistical Significant Tests

8.3.1 Repeated-measures ANOVA

To statistically evaluate the interaction between models (Bayesian vs. LLM) and steps, we calculated the repeated-measures ANOVAs. Each black box instance involved multiple repeated measurements corresponding to different steps. Letting Y_{ijk} represent the performance score for subject i, models j (Bayesian or LLM), and step k, the repeated-measures ANOVA model can be expressed as:

$$Y_{ijk} = \mu + S_i + M_j + T_k + (M \times T)_{jk} + \epsilon_{ijk}$$

where μ is the mean in all measurements, S_i represents the random effect of the subjects (individual seeds), M_j denotes the main effect of the model, T_k is the main effect of steps, $(M \times T)_{jk}$ is the interaction between the model and the step, and ϵ_{ijk} represents residual error.

The ANOVA decomposes the total variance into these distinct sources. Specifically, the significance of the interaction of the Step Method $\times$ was determined by calculating the corresponding F-statistic:

$$F = \frac{MS_{(M \times T)}}{MS_{error}}$$

where $MS_{(M \times T)}$ is the mean square for the Method $\times$ Step interaction, and MS_{error} is the residual mean square. Significance was assessed using an F-distribution with numerator degrees of freedom equal to $(J-1)(K-1)$, where J is the number of method levels and K is the number of steps, and denominator degrees of freedom equal to $(I-1)(K-1)$, where I is the number of subjects.

8.4 Prompts

8.4.1 Intervention prompt

```
In this task, you are given a ‘‘black box’’ and need to determine its inner
workings.
{black box information}
You will have a series of turns to interact with the black box.  On each
turn, you can either gather more information or test your hypothesis.  To
gather more information, {query instruction}, and obtain a result.
```

To test your hypothesis, {test instruction}. All the information gathered across all the turns is used to reverse engineer the black box. Throughout the process, you can decide whether the gathered information is sufficient to correctly identify the workings of the black box, in which case you can stop. Otherwise, you need to continue the interaction. Concretely, you can perform one of the following actions at each turn: 1) query, 2) test, or 3) stop.

Below is the format for each action.

Query:

```query
List[int]
```

Test:

```test
List[int]
List[int]
```

Stop:

```stop
```

Note that you should only perform one of the actions above with one input example in your response.

Below are your past observations of the black box:

{observations}

Response:

8.4.2 Evaluation Prompts

Program (judge):

In this task you will be given a ground truth program and pseudocode that you need to evaluate. You will output a score for the quality of the pseudocode based on a set of assessment criteria.

Below is the ground truth program:

{ground_truth}

Evaluate the quality of the following pseudocode:

{response}

Score the above pseudocode against the ground truth program based on the following criteria (total 10 points):

1. Does the provided pseudocode correctly specify the implementation of the ground truth program and manipulate the variables in the same way? Ignore the programming language difference. [5 point]

2. Does the provided pseudocode specify the implementation in the most simple and straightforward way without extra unused parts (Occam's Razor principle) [5 point]

Explain your judgement and return the final score with the type float and following the format below:

```judgement
YOUR JUDGEMENT HERE
```

```score
YOUR SCORE HERE
```

Response:

Formal Language (judge):

In this task, you will be given a ground truth formal language and a proposed rule describing that formal language, which you need to evaluate for quality. You will then output a score based on a set of assessment criteria.

Below is the ground truth formal language:
{ground_truth}

Evaluate the quality of the following formal language rule:
{response}
Score the above formal language rule against the ground truth formal language based on the following criteria (total: 10 points):
1. Does the provided rule correctly generate the examples given in the ground truth? Your score is determined by how many examples are correctly generated out of the total number of examples. [3 points]

2. Does the provided rule correctly reverse engineer the ground truth formal language? [5 point]
3. Is the provided rule in the most simple and straightforward way without extra unused parts (Occam's Razor principle)? Note: If the provided rule is completely incorrect, you should give 0 point for this criterion. [2 point]

Explain your judgement and return the final score with the type float and following the format below:

```judgement
YOUR JUDGEMENT HERE
```

```score
YOUR SCORE HERE
```

Response:

Math Equation (judge):

In this task, you are provided with a ground truth CES utility function and a CES utility function predicted by a model.

Your task is to evaluate the quality of the predicted utility function based on a set of assessment criteria and output a score.

The ground truth utility takes this form:

```
U(\\mathbf{{x}}) = \\left(\\sum_{{i=1}}^n a_i \\cdot
x_i^{{\\text{{rho}}}}\\right)^{{1/\\text{{rho}}}}
```

The utility depends on the following parameters:
1. a_i: float rounded to the first decimal point and should sum up to 1. (Note that there will be multiple a_i's.)
2. rho: float rounded to the first decimal point.

Below is the information about the ground truth utility function:
{ground_truth}

Evaluate the quality of the following predicted the parameters of the utility function:
{response}

Score the predicted utility function against the ground truth using the following criteria (total 10 points):
1. Is the predicted utility function has a correct rho? [2 points]
2. Compare the predicted utility function to the ground truth, how many a_i's are correct (order matters)? This will give us an accuracy percentage. The score for this bullet should be the accuracy percentage times the total allocated 6 points [6 points]
3. In the predicted utility function, do the unknown parameters a_i sum up to 1 and do the number of a_i's match the number of goods? [1 point]
4. Does the predicted utility function express the function in a simple and straightforward way without any unnecessary elements (adhering to the Occam's Razor principle)? [1 point]

Explain your judgement and return the final score with the type float and following the format below:

```judgement
YOUR JUDGEMENT HERE
```

```score
YOUR SCORE HERE
```

Response:

Descriptive Evaluation:

In this task, you are given a ``black box`` and need to determine its inner workings.
{black box information}

Below are some past observations from the black box:
{observations}

Your task is to reverse engineer the rule underlying {more detailed instructions} in the following format:

```Rule
YOUR RULE HERE
```

Response:

Implicit Function Evaluation:

In this task, you are given a ``black box`` and need to determine its inner workings.

{black box information}

Below are some past observations from the black box:

{observations}

{More detailed instructions}

Output your generated string in the following format:

```output
YOUR RESPONSE HERE
```

Response:

Explicit Function Evaluation:

In this task, you are given a ``black box`` and need to determine its inner workings.

{black box information}

Below are some past observations from the black box:

{observations}

Your task is to reverse engineer the rule underlying this {black box} by writing an executable Python function to generate strings that correspond to the rule of the black box.

{Specific Python function instruction}

The Python program should be in the following format:

```python
YOUR CODE HERE
```

Response: