LEARNING MENTAL MODELS FROM UNRELIABLE EVIDENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Children expertly construct mental models under uncertainty despite limited world knowledge. Which computational principles make this possible? We study this question in an interactive physical Box Task in which participants infer a latent cause through interaction with a stochastic and partially observed environment. The Box Task was originally designed to study children's puzzle-solving qualitatively. We contribute a computational analysis of behavior that formalizes the cognitive process by which children generate new hypotheses and respond to environmental uncertainty as inference unfolds. Our modeling framework is based on Bayesian particle-based inference, which maintains a small set of working hypotheses and continually generates new ones. We give two complementary implementations of hypothesis representations within this framework: (1) as constraints on evidence, and (2) as executable programs evaluated against evidence. Using the constraint-based formulation, we show that the model needs to account both for subjective reliability of evidence and online hypothesis generation to explain key signatures of children's behavior. Using the programmatic formulation, we show that hypotheses may take the form of partial compositional rules to guide information gathering when evidence interpretation is hard. The programmatic hypotheses are implemented using LLM-based program synthesis, evaluated across three prompt variants and several LLM backends, demonstrating robustness of our approach. Together, these results extend current models of children's inference to account for subjective reliability of evidence and open-ended hypothesis spaces, and provide a computational analysis of the form generated hypotheses may take in practice.

1 INTRODUCTION

When Newtonian mechanics subsumed Aristotelian physics, it marked a shift from heterogeneous frameworks of motion to a unified theory of inertia and force. Understanding how humans discover such complex models under real-world conditions remains an active challenge in cognitive science [35], robotics [8], and AI [39]. Developmental studies are particularly informative for uncovering its computational mechanisms, because children construct hypotheses with comparatively limited prior knowledge of the world that can inform model construction [15–17].

A central obstacle to the controlled study of real-world model building behavior is the scarcity of experimental frameworks that target naturalistic conditions [1, 39]. Prior developmental studies use relatively simple tasks in which behavior can be modeled with a constrained hypothesis set [3, 9, 13], to demonstrate that children are broadly consistent with approximate Bayesian inference. The Sampling Hypothesis, which proposes that children approximate Bayesian inference by sampling candidate hypotheses from a posterior distribution, is widely established to account for variability in children's responses better than random sampling [3, 13]. Children also selectively explore informative interventions, and adjust their exploration to both prior causal probabilities and the opportunities to discriminate among likely explanations [9]. However, models derived from the Sampling Hypothesis treat evidence as reliable and assume that candidate hypotheses are available in advance, which may not be the case in naturalistic settings.

Moving toward more naturalistic domains, we model behavior in the Box Task [31], an interactive puzzle originally designed to study children's problem-solving qualitatively. The Box Task combines several properties of naturalistic model building including counter-intuitive latent rules that partici-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

pants are unlikely to consider in advance, partial observability of the environment, and an incentive to explicitly model subjective reliability of evidence. At a computational level of Marr's analysis [26], this process constitutes a Partially Observable Markov Decision Process (POMDP) given stochastic state transitions and incomplete observations. At the representational level we approximate the intractable POMDP solution [21] with particle-based Bayesian inference [12].

Our modelling framework treats hypotheses as 'particles', allowing two complementary particle representations. The first, Sets of Constraints (SoC), represents hypotheses as lists of evidence-derived constraints. The second, LLM-based Program Synthesis (LLM-PS), represents hypotheses as executable programs generated and revised by an LLM, enabling tractable inference in a flexible hypothesis space [28, 37]. Here SoC functions as a principled model for identifying which cognitive principles drive human behavior, while LLM-PS allows us to examine the form such hypotheses may take when expressed in a Turing-complete high-level language. We show that both implementations replicate patterns in children's behavior, demonstrating how the Sampling Hypothesis can be extended to account for open-ended hypothesis spaces and subjective reliability of evidence in practice.

We contribute the following:

1. **A formalization of model building behavior** in the Box Task with two complementary implementations of hypotheses as constraints and as programs, extending current models of children's inference.
2. **A computational analysis of behavior** showing that subjective evidence reliability and online hypothesis generation jointly account for key signatures of children's behavior.
3. **Performance analysis of LLM-PS**, across prompt variants and LLM backends, demonstrating the form hypotheses may take and characterizing robustness of our approach.

2 METHODS

2.1 THE BOX TASK

In the Box Task study [31] 7-10 year old children ($N = 100$) were given 5 minutes to open 5 closed boxes using 13 keys. Boxes are uniquely colored, each marked with multiple instances of a single shape, and presented in a fixed order. Keys have colored fobs displaying either a shape or a number. The true rule, unknown to children, is that each box opens with the key whose number matches the number of shapes on the box (Figure 1A). Further, before commencing the task, children are misled that boxes open with color-matching keys. This rule is incorrect, as only the red box can be opened with the red key. The task is partially observable: not all box faces are visible in the initial line-up, and children are encouraged to pick up the boxes to examine them. Following the task, children completed four generalization trials selecting which of four new keys would open an unseen box, where one of the keys corresponds to the true rule and the others are foils. As children may open all boxes without discovering the rule, successful task completion does not entail correct generalization. Although 66% of children opened all five boxes, only 15% subsequently generalized the true rule. Full description of the procedure is given in Appendix A.

2.2 POMDP FORMALIZATION

The Box Task constitutes a POMDP, in which the state $s = (n_1, \ldots, n_5, o_1, \ldots, o_5, h)$ encodes the number of shapes on each box n_i, whether each box is open $o_i \in 0, 1$, and the rule (hypothesis) h predicting which key opens each box. The numbers n_i can be observed by picking up the box, o_i is fully observed, and the rule h is inferred during belief updating. At each step, the agent chooses between two action types: OBSERVE(i), which reveals n_i, and ATTEMPT(i, j), which attempts to open box i with key j.

ATTEMPT(i, j) produces an outcome $y \sim \text{Bernoulli}(\rho^*)$, where ρ^* reflects ground-truth evidence reliability. The action can succeed only if the true rule predicts a match. When $y = 1$, the state transitions to $o_i = 1$. Since o_i is fully observable, belief is maintained over shape counts and latent rule, $b = p(n_1, \ldots, n_5, h \mid \text{history})$.

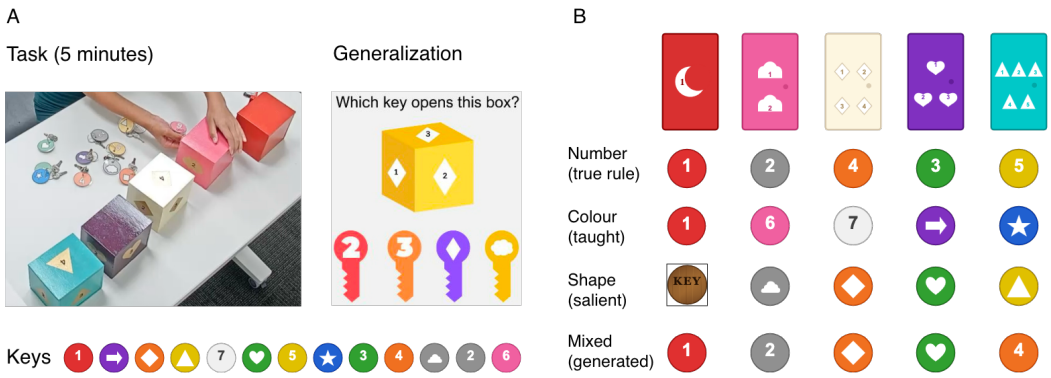


Figure 1: A. The physical Box Task setup, and the generalization phase presented on a tablet screen. B. Four SoC examples. *Number* shows the ground truth key-box mapping. *Color* represents the taught (misleading) rule. *Shape* rule assumes that any key could open the first box, while other boxes are matched by shape. *Mixed* rule sampled from the online generator selects key-box assignments stochastically. All four rules are consistent with red key opening the red box.

Exact maintenance of this joint belief over latent state and the open-ended rule space is intractable [21]; we therefore approximate belief inference using particles and use expected information gain as an action-selection heuristic.

2.3 Probabilistic Inference Framework

Our goal is to compute the joint belief distribution $p(h, y, e) = p(y|h, e)p(h)p(e)$, where h denotes latent rule, y denotes outcome, and e denotes action. We implement this using online Bayesian belief updating over a distribution of hypotheses using Sequential Monte Carlo Samplers (SMC-S; [11]), which is suited to settings where the posterior at each timestep shares a common hypothesis space — as is the case here. At each trial t, the agent maintains a population of N hypotheses (particles) $\{h_t^i, w_t^i\}_{i=1}^N$, where h_t^i is a hypothesis and w_t^i its weight. Here, the number of particles is a free parameter of the model (fewer particles converge faster), which we kept limited to a small number in the spirit of resource rationality [34]. An approximate posterior over hypotheses given $e_{1:t} = (\text{key}, \text{box})$ and outcomes $y_{1:t} \in \{0, 1\}$ is:

$$p(h \mid e_{1:t}, y_{1:t}) \approx \sum_i w_t^i \, \mathbb{1}[h = h_t^i].$$

At each step, the agent selects the key-box pair e^* maximizing expected information gain over the current particle distribution [25, 32]:

$$e^* = \arg\max_{e \in \mathcal{E}} \sum_i w_t^i \, \mathbb{E}_{y \sim p(\cdot|e,\, h_t^i)} \left[D_{\mathrm{KL}}\Big(p(h \mid e_{1:t}, y_{1:t}, e, y) \;\Big\|\; p(h \mid e_{1:t}, y_{1:t}) \Big) \right].$$

Belief updating. Upon observing outcome y_t of action e_t, particle weights are updated:

$$w_t^i \propto w_{t-1}^i \cdot p(y_t \mid e_t, h_t^i).$$

To model uncertainty over *evidence reliability* as the tendency to attribute failures to external causes (e.g., a stiff lock) rather than immediately falsifying the hypothesis, we introduce a *subjective reliability* parameter $\rho \in [0, 1]$.

Then, the likelihood is Bernoulli with success rate ρ when h predicts success and 0 otherwise:

$$p(y \mid e, h) = \text{Bernoulli}(y;\ \rho \cdot \mathbb{1}[h \text{ predicts } e \text{ succeeds}]).$$

Importantly, while the ground truth key behavior is objectively stochastic, here ρ reflects a subjective estimate of how reliable keys are, supported by empirical evidence that children frequently retry the same failing key–box pair on consecutive trials (AppendixE.3).

When the effective sample size $\text{ESS} = 1/\sum_i (w_t^i)^2$ falls below $N/2$, particles are resampled proportionally to their weights and rejuvenated by drawing from the proposal distribution $q(h)$. The rejuvenation mechanism is representation-specific, and described in Sections 2.3.1 and 2.3.2. Particles are initialized with weights $w_1^i \propto p(h_1^i)/q(h_1^i)$. In a special case where $q(h)$ is configured to track a small set of hypotheses and evidence is treated as reliable, our framework instantiates the Sampling Hypothesis [13] currently accepted as a model of children's rule inference.

2.3.1 HYPOTHESES AS SETS OF CONSTRAINTS (SOC)

Our SoC representation implements each hypothesis h^i as a *Set of Constraints* (SoC): a mapping of boxes to keys predicted to open them. Examples of such sets, including the true rule and the taught color rule are shown in Figure 1 B. The initial particle population is drawn from $q(h)$, which is implemented as a mixture of two components:

- **Pre-specified rules:** Following the Sampling Hypothesis, we specify a set of candidate rules matching by color, shape, or box order, each assigned a prior probability $p(h)$.
- **Online generator:** A generative mechanism for proposing novel SoCs not in the pre-specified set, implemented by sampling of key-box assignments conditioned on evidence. Assignments consistent with observed successes are included as a hard constraint, while assignments that have been falsified are down-weighted in proportion to their accumulated failure probability under the current reliability estimate ρ.

A parameter p_{gen} controls the proportion of prior mass allocated to the generator versus pre-specified rules. Following the Sampling Hypothesis, the true rule (key number equals number of shapes) is also included in the pre-specified set, with its prior probability p_t. In practice, we fixed $p_t = 0.02$ for all model variants reported here, as sweeping $p_t \in [0.01, 0.20]$ consistently resulted in fitted values concentrated near $p_t = 0.02$ across individuals.

Model variants. We construct four SoC variants, including three lesioned versions:

1. **Lesioned (SoC-Lesioned)**, reliable evidence and no generator ($\rho = 1$, $p_{gen} = 0$).
2. **Subjective reliability only (SoC-Rel)**, no generator ($p_{gen} = 0$)
3. **Generator only (SoC-Gen)**, no subjective reliability ($\rho = 1$).
4. **Full model (SoC-Full)**, with both parameters p_{gen} and ρ present.

Special cases of these models instantiate several distinct behaviors. SoC-Gen with $p_{gen} = 1$ reduces to a random sampler without replacement. SoC-Full with $p_{gen} = 1$ and $\rho = 0$ reduces to a random sampler with replacement. The SoC-Lesioned formulation effectively instantiates the Sampling Hypothesis. Note that SoC models the ATTEMPT component of the POMDP formulation, but not the OBSERVE policy. The latter is instantiated in LLM-PS-P (see next Section).

2.3.2 HYPOTHESES AS SYMBOLIC PROGRAMS (LLM-PS)

The LLM-PS representation implements hypotheses h^i as executable Python functions `predict(key, box) -> bool`, evaluated against evidence. The initial particle population is generated by prompting the LLM with a description of the Box Task environment that includes the misleading instruction as a starter hypothesis, and a code template specifying `Key` and `Box` data structures. Each particle is an independently sampled program consistent with this context. When a particle h^i is selected for rejuvenation, the LLM is prompted to synthesize a revised version of the corresponding hypothesis consistent with accumulated evidence. This inference machinery determines *which* hypotheses to revise and *when*, while the LLM determines *how* to do so.

Model variants. We implement three model variants and one REACT baseline:

1. **LLM-PS, full observability, reliable keys**: The LLM receives a complete description of the environment, including full specification of all boxes describing the symbols and numbers on each face. Keys succeed deterministically when attempting to open the correct box.

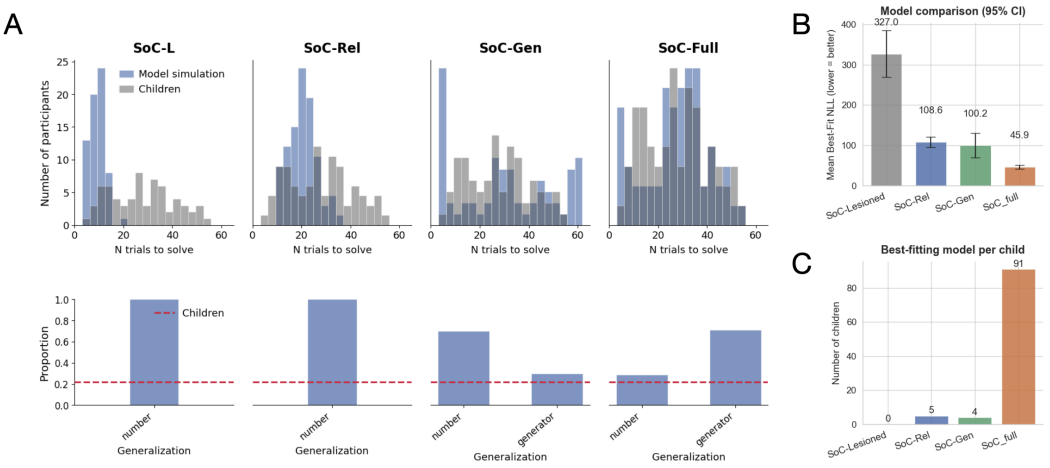


Figure 2: A, Top row. Histograms showing the number of trials required to open all boxes for children and models. Each column represents a different model variant. The same children's histogram is repeated in each column as a reference. Bottom row: Distribution of most likely hypotheses for each model after opening all boxes. The correct rule, "number", was generalized to by 15% of children. B. Negative Log-likelihoods of each model (shorter = better fit). Error bars indicate 95% CI computed via standard error of the mean across children. C. Number of children best fitted by each model.

2. **LLM-PS-S, full observability, unreliable keys**: Evidence reliability is degraded to match empirical success rates experienced by children (see Appendix E.3).
3. **LLM-PS-P, partial observability, unreliable keys**: Box face specification is incomplete in the starter code. The prompt states that the agent may issue OBSERVE(i) actions to reveal them, alongside the regular hypothesis generation. Keys remain unreliable.
4. **REACT baseline [38] with full observability and reliable keys**. LLM is prompted sequentially with evidence history and prompted to produce key-box actions without generating explicit hypotheses. This serves as a baseline capable of completing the task, but without producing interpretable hypotheses or generalization to unseen boxes.

The prompts are shown in Appendix C. LLM-PS instantiations vary in LLM backend and number of particles. Subjective evidence reliability is modeled implicitly by the LLM engine. Parameter p_gen is not modeled directly, as LLM-based models do not separately encode pre-defined vs. stochastically sampled rules. Unlike SoC, LLM-PS rejuvenation is implemented by rejuvenating the worst-fitting hypothesis at every trial, which limits API costs. At task completion, we identify the highest-weight particle in the final particle set. An LLM simulation is classified as **having generalized** the true rule if this particle implements the correct number-matching rule. Importantly, we do not treat LLM-PS as a cognitive model of children's behavior. Rather, we use it as a computational tool for open-ended hypothesis generation, to examine the form such hypotheses may take.

Action selection. In LLM-PS-P, the model can issue OBSERVE(i) actions to update the environment context passed to subsequent LLM calls. Particles generated prior to observation retain their original representation and may be discarded through subsequent rejuvenation; particles generated after observation can exploit the revealed information about box layout. This implements a factorized approximation to the joint belief $p(n_{1:5}, h \mid \text{history})$, treating state uncertainty and rule uncertainty independently in a tractable approximation of the formalism described in Section 2.2.

3 EXPERIMENTS

3.1 CHILDREN'S MODELING RESULTS (SOC)

We first evaluate the SoC model's ability to capture (1) an individual's task-completion trajectory, defined as cumulative boxes opened over trials, and (2) the final inferred hypothesis. We fit each model

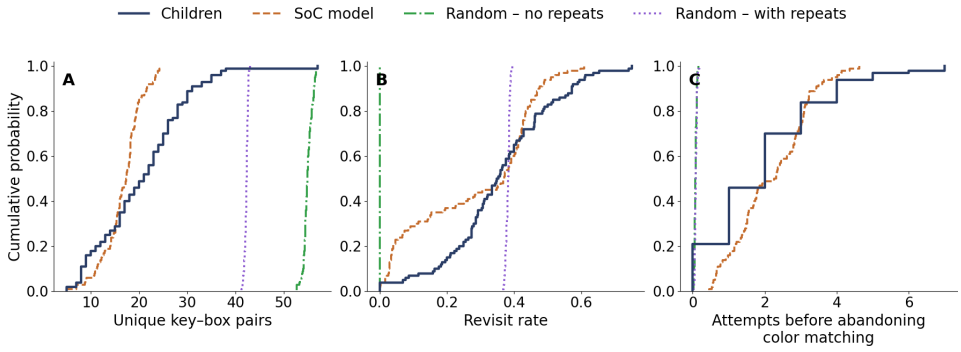


Figure 3: **Comparison of observed behavior with the SoC model and random-search baselines.** Distributions of (A) Unique key-box pairs attempted (B) key-box revisit rate and (C) attempts before abandoning color matching for the fitted SoC model (left), random sampling without replacement (middle), and random sampling with replacement (right).

variant to individuals, with parameter selection maximizing the composite log-likelihood of that child's trajectory under each model (Appendix D), and report the best fitting model per child (Figure 2C) along with the population mean LL for each model (Figure 2B). The SoC-Full model provides a significantly better fit than all lesioned variants ($p < .001$): SoC-Rel ($t = 13.39, d = 1.34$), SoC-Lesioned ($t = 10.14, d = 1.01$), and SoC-Gen ($t = 3.90, d = 0.39$). AIC-based model comparison confirms this ranking after penalising for model complexity (Appendix D, Table 5).

Figure 2A shows histograms of trials required to open all five boxes for each model variant, given the fitted parameters, alongside children's distribution. Only the SoC-Full model closely reproduces the shape and spread of children's aggregate distribution. The bottom row of Figure 2 shows the distribution of final inferred rules. The lesioned models without the online generator (SoC-Lesioned and SoC-Rel) can only generalize to pre-defined rules, forcing them to consistently pick the true rule. Models with the online generator (SoC-Gen and SoC-Full) can complete the task without discovering the true rule, converging on a set of independent constraints for each box, labeled 'generator' in the figure. The SoC-Full model converges on the correct rule in approximately 20% of simulations, similar to children's generalization.

Lastly, children's behavior cannot be explained by a fully stochastic process, such as random sampling without or without replacement. Notably, the special case of SoC-Gen with maximal p_{gen} value that reduces to random sampling without replacement shows a poor fit to behavior (see Appendix D and Figure 7). The special cases of SoC-Full with maximal value of p_{gen} and various values of ρ, that enable key resampling, is likewise a poor fit. We additionally compare our model predictions with random-sampling baselines either with or without replacement (Fig. 3) on three metrics: how many unique key-box pairs were attempted, key-box revisit rate, and attempts before abandoning color matching, demonstrating that only our model captures the empirical metrics. Additional evidence against a random-sampling account provided by children's initial bias toward testing the instructed color rule is shown in (AppendixE Figure 16).

Finally, the majority of children express a significant rule preference during generalization, by above-chance picking key-box combinations consistent with a specific rule; and only 26 did not show such a preference (Appendix E, Figure 14). Taken together these results establish that subjective reliability (ρ) and online hypothesis generation (p_{gen}) are jointly required to account for children's behavioral signatures.

3.2 Simulation Results (LLM-PS)

LLM-PS performance across backends. We evaluated the three LLM-based model variants with three LLM backends, with different architectures and code generation capabilities (GPT-5.2, LOW REASONING, GPT-5.1, DEEPSEEK V3.2). DEEPSEEK V3.2 and GPT-5.1 were instantiated with $n = 10$ particles, as a sufficient number at which the task could be consistently completed in under 70 attempts – the maximal number in behavioral data. We also found that a more capable model,

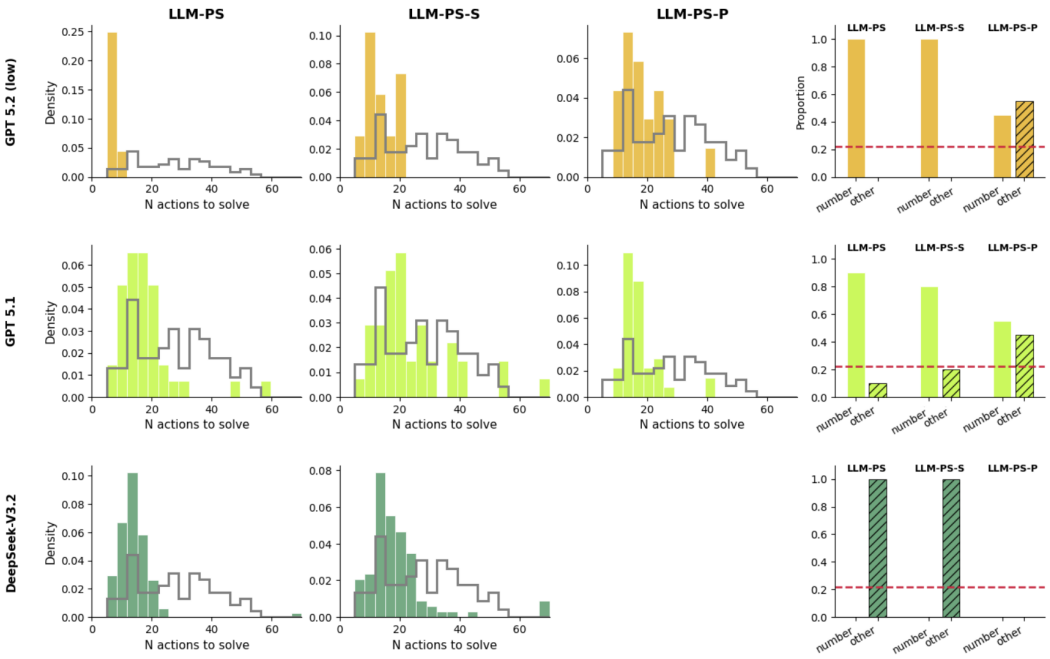


Figure 4: Left: Densities of number of attempts required to open all five boxes under each LLM-PS variant, across LLM backends (rows). Columns correspond to model variants. Right: Generalization of the true rule by each model variant. The red dotted line shows children's generalization from [31].

GPT-5.2, was able to complete the task with a single particle. In the single-particle limit, LLM-PS reduces to sequential hypothesis revision: the agent returns a single hypothesis, used by the inference engine to greedily select actions; the rejuvenation step prompts to update this hypothesis given history. Empirically, this simplification is consistent with the view that capable inference systems can operate effectively under a single-hypothesis regime [6]. At the computational level, one-particle LLM-based models with a GPT-5.2 back-end can be seen as an instantiation of the Win-Stay, Loose-Sample framework [3] with interpretable hypotheses.

Figure 4 shows the number of actions required to complete the task across LLM-based model variants and LLM backends. The ReAct baselines (not plotted) consistently completed the task in under 15 attempts across backends, confirming that under reliable and fully observable conditions task completion does not require explicit hypothesis maintenance. Here, ReAct simply provides a task completion baseline, without explicit hypothesis representations. Given that DEEPSEEK V3.2 failed to discover the true rule under full observability (LLM-PS and LLM-PS-S), it was not further evaluated under LLM-PS-P.

LLM-PS performance across prompt variants. Example trajectories for two model variants using GPT-5.2 are shown in Appendix C Figure 6. Both start with a color-matching hypothesis, consistent with the misleading instruction, but converge on the correct rule. Similar to children, LLM-based models dissociate task completion and generalization under increasing uncertainty. When the model fails to discover the correct rule despite completing the task, the failure modes were similar across LLMs. In such cases, task success was achieved by accumulating coincidental successful box openings driven by hypotheses that express disparate causes, rather than unified rules (Appendix C), a property also captured by the online generator in the SoC. Notably, the weaker model, DEEPSEEK V3.2, consistently failed to discover the true rule even in fully observable conditions. These results demonstrate how LLM-based proposals in SMC-S can produce interpretable hypotheses based on evidence history, demonstrating the form hypotheses might take during online generation.

Key differences between children and LLM-based proposal behaviour. Across prompt variants, LLM-based proposals strictly complied with instructions, consistently generating the color-matching hypothesis on the first trial. In contrast, 21 out of 100 children tested a key-box combination inconsistent with color matching, suggesting that children may have competing inductive biases or

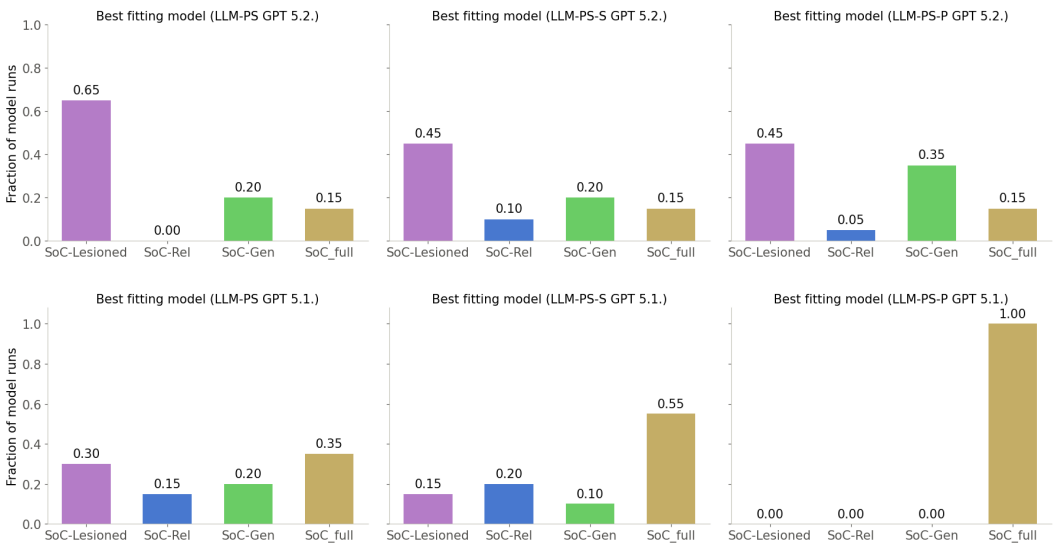


Figure 5: Best-fitting SoC variants for ATTEMPT actions generated by LLM-PS with GPT 5.2 (top) and GPT 5.1 (bottom). Bars show the fraction of runs assigned to each SoC variant.

assume a broader hypothesis space. This property of children's behavior is captured by SoC, but not by LLM-based proposals. Further, in contrast to SoC, LLM-PS-P supports both OBSERVE and ATTEMPT actions. Our instantiation of LLM-PS-P exhibits a more systematic observation policy than children (Appendix Figure 9), who vary substantially in how and when they observe boxes: 30 out of 100 did not observe any boxes; the majority (79 out of 100) examined only a subset; and 22 out of 100 began the experiment by observing boxes rather than attempting keys. In contrast, LLM-PS-P consistently begins by generating ATTEMPT actions and switches to systematic OBSERVE actions after a few failed attempts (Appendix C.4). The differences between LLM-based proposals and children's behavior may be due to observations differing in informativeness, cost, or both.

Behavioral correspondence between LLM-based and SoC proposals. Unlike SoC proposals, whose behavior is explicitly parameterized by ρ and p_{gen}, LLM-based proposals do not expose corresponding parameters. Their differences in generated hypotheses arise implicitly from the LLM and its prompt. To characterize this implicit behavior, we fit the SoC models to ATTEMPT actions generated by each LLM-PS variant (Figure 5). This provides a mapping from the LLM-based proposal mechanisms onto the interpretable SoC parameterization. Across LLM backends, under fully observable, reliable conditions, LLM-PS maps most closely onto SoC-Lesioned, whereas introducing unreliable evidence (LLM-PS-S) and partial observability (LLM-PS-P) progressively shifts behavior toward the SoC-Full.

4 RELATED WORK

Bayesian models of learning. Bayesian inference has been widely applied to model child [4, 15] and adult causal learning [18, 22, 23, 33]. Most directly related, the Sampling Hypothesis proposes that children sample causal hypotheses from a fully predefined proposal distribution weighted by posterior probability [3, 13, 36]. Our SoC model extends this principle to complex domains where evidence is unreliable and the proposal cannot be entirely predefined. In other related work, Bramley & Xu [5] find that children generate more diverse hypotheses but converge on correct rules less reliably than adults, a pattern consistent with our finding that children's hypothesis space admits solutions that succeed at task completion without capturing the true rule.

Program synthesis for inductive reasoning. Much work across AI and cognitive science uses program synthesis to model inductive reasoning. Classically such models have had relatively narrow success because they use custom domain-specific programming languages [2, 7, 14, 19, 20, 23, 27], although many of those successes have been practically and scientifically valuable. Recent work with

general-purpose languages such as Python using LLMs has increased the scope of applications [10, 24, 29, 30, 37]. Most closely related, Piriyakulkij et al. [28] also combines SMC-S and Bayesian Experiment Design with LLM-based proposal distributions. Our work differs in core ways, but most importantly in its comparison to children and SoC models, and in its use of a naturalistic task domain in which task completion and model learning can be dissociated. Notably, our finding that GPT-5.2 completes the task with a single particle is consistent with the view that capable inference systems can operate effectively under a single-hypothesis regime [3, 6]. Our view is that programs induced from naturalistic data should not assume full observability, and that decision-making must consider uncertainty about both the program hypothesis and the underlying state. Indeed, absent this modeling choice, we cannot explain the human data we consider.

5 Discussion

We introduced a computational framework modeling the behavior of children interacting with the Box Task, which expresses several key properties of model building in a naturalistic setting: evidence uncertainty, partial observability, and non-intuitive causal rules. Our framework formalizes behavior in this task as the process of hypothesis generation, interpreting unreliable evidence, and sequential belief revision. A key component of our framework is the mechanism that specifies how new hypotheses may be generated given evidence history. We implement this component in two complementary ways: the SoC class of models, which extends the Sampling Hypothesis [13] to uncertain domains with open-ended hypothesis spaces, and a program synthesis LLM-PS class of models, which enables tractable inference over expressive, interpretable, and executable hypotheses.

Using SoC lesion analysis we show that evidence uncertainty and online hypothesis generation are jointly needed to account for children's behavior. Additional LLM-based modeling demonstrates the form that these hypotheses might take in an uncertain environment: compositional or partial rules, that specify the underlying causal model only partly. Our modeling framework shows the dissociation between task completion and correct generalization similar to children, as both LLM-based and SoC model families can generate hypotheses that integrate disparate causes rather than propose new unified rules. We show that such hypotheses can be instrumental both to completing the task without correct generalization, and to accumulating evidence, which can then inform generation of improved unified rules. The extent to which human-generated hypotheses may follow a similar pattern from cause aggregation to unification remains to be tested in future work.

We find that LLM-based hypothesis generation handles unreliable evidence gracefully, but appears to work with a fairly restrictive prior over hypotheses. Generated hypotheses primarily match keys and boxes by simple salient properties: shape identity, color, and ultimately number of shapes. In our simulation experiments, we did not encounter LLMs coming up with plausible but counter-intuitive rules, for instance *'The number on the key needs to be odd and greater than the position of the box in the line-up'.* or *'The rule changes after completing 10 attempts.'*. This focus on simple rules explains the efficiency of LLM-based model performance. Possibly, training on natural language descriptions of experiment-like tasks induces an LLMs bias toward small hypothesis spaces, and training on noisy data (i.e. unreliable witnesses) could inform discounting conflicting evidence.

Notably, as the Box Task originates from a published study [31], LLM training corpora may have included the task design or its solutions. While contamination cannot be fully excluded, the observed trajectories consistently generate the simpler salient hypotheses before discovering the true rule and demonstrate evidence-sensitive revision conditional on the proposals produced by the LLM. Moreover, weaker models fail to generate the true rule entirely.

Limitations and Future work The assumption that $\textsc{Observe}(i)$ actions in LLM-PS-P are fully informative could be relaxed by introducing noisy observations in future work. Future work could also consider a mechanistic investigation of whether LLM-based agents may maintain explicit hypothesis representations using open-weight models. Follow-up experiments with human adults could subject human adults and LLM-based models to matched environments (fully observed and reliable; fully observed and stochastic; partially observed and stochastic) while eliciting explicit descriptions of hypotheses from human adults. Doing so would inform comparative analysis of hypotheses generated by LLM-based models and humans, offering insight into how the underlying hypotheses are similar or different in quality and diversity between human and artificial agents.

AI USE STATEMENT

LLMs were used as part of the methodology described in this paper, specifically as proposal mechanisms for structural decompositions and reconstruction programs in the Generative Map Module; these uses are described in detail in the Methods and Appendix.

Generative AI tools were also used for routine coding assistance with generating plots, proofreading the completed manuscript, and literature discovery. All AI-assisted content and suggested references were reviewed and verified by the authors, who take responsibility for the final content of the paper.

REFERENCES

[1] Kelsey Allen, Franziska Brändle, Matthew Botvinick, Judith E Fan, Samuel J Gershman, Alison Gopnik, Thomas L Griffiths, Joshua K Hartshorne, Tobias U Hauser, Mark K Ho, et al. Using games to understand the mind. *Nature human behaviour*, 8(6):1035–1043, 2024.

[2] Matej Balog, Alexander L Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. *arXiv preprint arXiv:1611.01989*, 2016.

[3] Elizabeth Bonawitz, Stephanie Denison, Alison Gopnik, and Thomas L Griffiths. Win-stay, lose-sample: A simple sequential algorithm for approximating bayesian inference. *Cognitive psychology*, 74:35–65, 2014.

[4] Elizabeth Bonawitz, Stephanie Denison, Thomas L Griffiths, and Alison Gopnik. Probabilistic models, learning algorithms, and response variability: sampling in cognitive development. *Trends in cognitive sciences*, 18(10):497–500, 2014.

[5] Neil R Bramley and Fei Xu. Active inductive inference in children and adults: A constructivist perspective. *Cognition*, 238:105471, 2023.

[6] Neil R Bramley, Peter Dayan, Thomas L Griffiths, and David A Lagnado. Formalizing neurath's ship: Approximate algorithms for online causal learning. *Psychological review*, 124(3):301, 2017.

[7] François Chollet. On the measure of intelligence. *CoRR*, abs/1911.01547, 2019. URL http://arxiv.org/abs/1911.01547.

[8] Katherine M Collins, Ilia Sucholutsky, Umang Bhatt, Kartik Chandra, Lionel Wong, Mina Lee, Cedegao E Zhang, Tan Zhi-Xuan, Mark Ho, Vikash Mansinghka, et al. Building machines that learn and think with people. *Nature human behaviour*, 8(10):1851–1863, 2024.

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

[10] Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. Generating code world models with large language models guided by monte carlo tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15383.

[11] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 05 2006. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2006.00553.x. URL https://doi.org/10.1111/j.1467-9868.2006.00553.x.

[12] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.

[13] Stephanie Denison, Elizabeth Bonawitz, Alison Gopnik, and Thomas L Griffiths. Rational variability in children's causal inferences: The sampling hypothesis. *Cognition*, 126(2):285–300, 2013.

[14] 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.

[15] Alison Gopnik. Childhood as a solution to explore–exploit tensions. *Philosophical Transactions of the Royal Society B*, 375(1803):20190502, 2020.

[16] Alison Gopnik and Laura Schulz. Mechanisms of theory formation in young children. *Trends in cognitive sciences*, 8(8):371–377, 2004.

[17] Alison Gopnik, Clark Glymour, David M Sobel, Laura E Schulz, Tamar Kushnir, and David Danks. A theory of causal learning in children: causal maps and bayes nets. *Psychological review*, 111(1):3, 2004.

[18] Thomas L Griffiths, Joshua B Tenenbaum, and Charles Kemp. Bayesian inference. *The Oxford handbook of thinking and reasoning*, pp. 22–35, 2012.

[19] Sumit Gulwani. Automating string processing in spreadsheets using input-output examples. *SIGPLAN Not.*, 46(1):317–330, January 2011. ISSN 0362-1340. doi: 10.1145/1925844.1926423. URL https://doi.org/10.1145/1925844.1926423.

[20] Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid, and Benjamin Zorn. Inductive programming meets the real world. *Commun. ACM*, 58(11):90–99, October 2015. ISSN 0001-0782. doi: 10.1145/2736282. URL https://doi.org/10.1145/2736282.

[21] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[22] Charles Kemp and Joshua B Tenenbaum. Structured statistical models of inductive reasoning. *Psychological review*, 116(1):20, 2009.

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

[24] Wen-Ding Li and Kevin Ellis. Is programming by example solved by llms? In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

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

[26] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, 1982.

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

[28] Top Piriyakulkij, Cassidy Langenfeld, Tuan Anh Le, and Kevin Ellis. Doing experiments and revising rules with natural language and probabilistic reasoning. *Advances in Neural Information Processing Systems*, 37:53102–53137, 2024.

[29] Wasu Top Piriyakulkij, Yichao Liang, Hao Tang, Adrian Weller, Marta Kryven, and Kevin Ellis. Poe-world: Compositional world modeling with products of programmatic experts. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[30] Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, and Xiang Ren. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=bNt7oajl2a.

[31] Mia Radovanovic, Ece Yucer, and Jessica A Sommerville. Girls persist more but divest less from ineffective teaching than boys. *Journal of Experimental Psychology: General*, 2024.

[32] Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern bayesian experimental design. *Statistical Science*, 39(1):100–114, 2024.

[33] Joshua Stewart Rule. *The child as hacker: building more human-like models of learning*. PhD thesis, Massachusetts Institute of Technology, 2020.

[34] Herbert A Simon. A behavioral model of rational choice. *The quarterly journal of economics*, pp. 99–118, 1955.

[35] Bas van Opheusden and Wei Ji Ma. Tasks for aligning human and machine planning. *Current Opinion in Behavioral Sciences*, 29:127–133, 2019.

[36] Edward Vul, Noah Goodman, Thomas L Griffiths, and Joshua B Tenenbaum. One and done? optimal decisions from very few samples. *Cognitive science*, 38(4):599–637, 2014.

[37] Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah Goodman. Hypothesis search: Inductive reasoning with language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=G7UtIGQmjm.

[38] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Irina Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

[39] Lance Ying, Katherine M Collins, Prafull Sharma, Cedric Colas, Kaiya Ivy Zhao, Adrian Weller, Zenna Tavares, Phillip Isola, Samuel J Gershman, Jacob D Andreas, et al. Assessing adaptive world models in machines with novel games. *URL https://arxiv.org/abs/2507.12821*, 2025.

A THE BOX TASK SETUP

The Box Task environment is introduced in [31]. Below we summarize relevant details.

Children completed the experiment in person, in four phases:

1. *Practice*: Participants tried plain keys on a plain practice box to familiarize themselves with the key action, including both correct and incorrect keys.
2. *Instruction*: Children viewed an instructional video in which a teacher demonstrated an incorrect color-matching rule, using a red-fobbed key to open the red box. This demonstration was confounded: for the red box only, the correct key happened to match both color and number. For all other boxes, color-matched keys were incorrect. The teacher communicated the rule as definitive and provided no indication that it would fail.
3. *Test*: The five boxes were arranged in a fixed order in front of the participant. All 13 keys were placed in a single pile. Children were instructed to open all five boxes within a 5-minute time limit, working independently. The experimenter remained present but provided no feedback. Boxes could be picked up and examined. The test ended upon opening all five boxes or at time expiry.
4. *Generalization*: Four forced-choice trials using novel box images were presented on a tablet screen. For each box, four keys were presented: a color-matched key, a shape-matched key, a number-matched key (correct), and a number foil. Children selected which key they believed would open the box.

Box Task Environment. The boxes were uniquely colored physical objects, 11 cm $\times$ 11.9 cm $\times$ 10.6 cm, featuring a different number of shapes on their faces. Each shape was labeled with a number to indicate quantity (e.g., the pink box had two clouds, labeled "1" and "2"). Each box had at least one shape. Keys were metal physical objects. Each key had a colored tag displaying either a number or a shape.

Partial Observability. The task is partially observable: subjects initially observe each box's color and shape identity, but not the total number of shape instances. The exact shape count is revealed only if the subject picks up and examines the box.

At each time t, subjects choose between:

- **Observe:** observe(i) reveals the exact number of shapes on box i
- **Attempt:** attempt(i, j) attempts to open box i with key j, where $i \in \{1, \ldots, 5\}$ and $j \in \{1, \ldots, 13\}$

Box	Order	color	Shape	Shape count (Number)
B1	1	red	moon	?
B2	2	pink	cloud	?
B3	3	white	diamond	?
B4	4	purple	heart	?
B5	5	blue	triangle	?

Table 1: Initial box configuration as observed by subjects. Color, shape identity, and order are fully observable. Shape count requires picking up the box.

Key	color	Shape	Number
red1	red		1
pink6	pink		6
grey2	grey		2
greycloud	grey	cloud	
orange4	orange		4
green3	green		3
bluestar	blue	star	
yellow5	yellow		5
greenheart	green	heart	
white7	white		7
triangleyellow	yellow	triangle	
diamondorange	orange	diamond	
purplearrow	purple	arrow	

Table 2: Set of available keys. Each key is associated with a color and either a number or a shape (but not both). Numbers range beyond the possible box counts, introducing distractors; shape-based keys provide alternative matching cues.

Box	Order	color	Shape	Number
B1	1	red	moon	1
B2	2	pink	cloud	2
B3	3	white	diamond	4
B4	4	purple	heart	3
B5	5	blue	triangle	5

Table 3: True (unobserved) configuration of the boxes. The true number of shapes on each box becomes observable with inspection.

B POMDP FORMALIZATION

The Box Task constitutes a Partially Observable Markov Decision Process, in which the state $s = (n_1, \ldots, n_5, o_1, \ldots, o_5, h)$ encodes the number of shapes on each box n_i, whether each box is open $o_i \in \{0, 1\}$, and the rule (hypothesis) h predicting which key opens each box. The numbers n_i can be observed by picking up the box, o_i is fully observed, and the rule h can be inferred during belief updating. The reward $r = 1$ when $\forall i \; o_i = 1$, and 0 otherwise. The goal state is reached when all boxes are open, $\sum_i o_i = 5$. At each step, the agent chooses between two action types: OBSERVE(i): deterministically reveals information about n_i and ATTEMPT(i, j): attempts to open box i with key j.

ATTEMPT(i,j) produces an outcome $y \sim$ Bernoulli$(\rho*)$, where $\rho*$ reflects ground truth evidence reliability. The action can succeed only if the oracle rule predicts a match. When $y = 1$, the state transitions to $o_i = 1$. Since o_i is fully observable, the belief is a joint distribution over shape counts and latent rule: $b = p(n_1, \ldots, n_5, \; h \mid \text{history})$. Updating h after taking an action requires computing likelihood over possible rules.

We use the POMDP to formalize the environment and latent state. As exact maintenance of the joint belief $p(n_{1:5}, h \mid \text{history})$ is intractable [21], we approximate belief inference using particles and use expected information gain as an action-selection heuristic.

C LLM PROMPTS AND OUTPUT EXAMPLES

LLM-PS-S (Unreliable evidence)

Trial 1: Color rule (misleading prior)

```
def predict(key, box) -> bool:
    # Match key color to box color
    return key.color == box.color
```

Trial 6: Partial revision

```
def predict(key, box) -> bool:
    # color for first box, then shape
    if box.position == 1:
        return key.color == box.color
    return (key.shape is not None and
            key.shape == box.shape)
```

Trial 14: True rule

```
def predict(key, box) -> bool:
    # Key number must match the number
    # of shapes on the box.
    return (key.number is not None and
            key.number in box.number)
```

LLM-PS-P (Partial Obs., Unreliable)

Trial 1: Color rule (misleading prior)

```
def predict(key, box) -> bool:
    # Match key color to box color
    return key.color == box.color
```

Trial 9: Uncertain number belief

```
def predict(key, box) -> bool:
    if key.number is None:
        return False
    if len(box.number) == 0:
        return True  # no info: try it
    return key.number in box.number
```

Trial 18: True rule

```
def predict(key, box) -> bool:
    # Key number must match the number
    # of shapes on the box.
    return (key.number is not None and
            key.number in box.number)
```

Figure 6: Representative hypothesis trajectories generated by LLM-PS-S (left) and LLM-PS-P (right).

C.1 LLM-PS-P (UNRELIABLE EVIDENCE, PARTIALLY OBSERVABLE)

System Prompt

```
You are an intelligent agent playing a game.
Your task is to open 5 boxes using 13 keys in the fewest attempts.
You do not need special skills to play this game.
This game can be played by an 8-12 year old child.
```

User Prompt

```
For each box there is a key that opens it.
The goal of the game is to find the right key for each box, using as few actions as p
You have a demonstration video from a teacher telling you how to open all boxes.
In the video, the teacher says:
"I'm going to show you the right way to unlock the boxes.
To open the boxes, you have to use a key that matches the color of the box.
So, to open this red box, I'm going to use this red key.
Great, now you can open all the doors!"

Each key has an identifier (id) and a color.
Each key also has either a number or a shape, but not both.

Each box has a color. It also has a shape, which is printed on at least one of its fa
Not all faces are visible to you initially, but the game allows you to pick up boxes
examine them to get more information.

Here are the boxes, lined up in this order:
The first box is red, has a moon shape.
The second box is pink, has a cloud shape.
The third box is white, has a diamond shape.
The fourth box is purple, has a heart shape.
The fifth box is blue, has a triangle shape.

Here are the 13 keys (in no specific order):
The red key is red and has the number 1.
The pink key is pink and has the number 6.
The grey2 key is grey and has the number 2.
The cloud key is grey and has a cloud shape.
The orange4 key is orange and has the number 4.
The green3 key is green  and has the number 3.
The blue key is blue  and has a star shape.
The yellow5 key is yellow and has the number 5.
The heart key is green and has a heart shape.
The white key is white and has the number 7.
The triangle key is yellow and has a triangle shape.
The diamond key is orange and has a diamond shape.
The purple key is purple and has an arrow shape.

You can interact with this environment by taking two types of actions:
(1) Attempt Action: write Python code that will be used to generate opening attempts,
(2) Observe Action: request more information about a given box.

To Take the Observe Action, your output should be exactly
"PICK UP x", where x is the box id (do not use any other attributes of the box)

To take an Attempt Action, you will need to write a Python function that specifies
a hypothesis about which keys open which boxes.
Your output should use the given starter code, and complete the function called predi
according to its signature.
Your output should contain only the Python program for predict, absolutely nothing el
It should NOT contain the Key or Box classes.

Here is the history of actions taken and observed evidence.
Please use them to make your decision:

Open box {BOX1} with key {KEY1}: {OUTCOME1}
Examine {BOX2}: {BOX2.number (real)} faces have shapes on them
Open box {BOX3} with key {KEY3}: {OUTCOME3}
...
```

```
Here is the starter code for attempt action

def predict(key, box) -> bool:
    # key is a Key object
    # box is a Box object
    # fill in your code

class Key:
    def __init__(self, id: str, color: str, number: Optional[int], shape: Optional[st
        self.id = id
        self.color = color
        self.number = number
        self.shape = shape

class Box:
    def __init__(self, id: str, color: str, shape: str, number: set, position: int):
        self.id = id
        self.color = color
        self.shape = shape
        self.position = position
        self.number: set
        # Current belief over number of shapes on this box.
        # Unknown until PICK UP action is taken.
        # After PICK UP: collapses to a singleton set containing the exact value.
        # In predict(), handle both uncertain and certain cases.

Here are the accurate data for all keys and boxes.
The indexing of the arrays corresponds to the line-up order of boxes.

key_data = {
    "id": ["red", "pink", "grey2", "cloud", "orange4", "green3", "blue", "yellow5",
           "heart", "white", "triangle", "diamond", "purple"],
    "color": ["red", "pink", "grey", "grey", "orange", "green", "blue", "yellow",
           "green", "white", "yellow", "orange", "purple"],
    "number": [1, 6, 2, None, 4, 3, None, 5, None, 7, None, None, None],
    "shape": [None, None, None, "cloud", None, None, "star", None, "heart", None,
           "triangle", "diamond", "arrow"]
}

box_data = {
    "id": ["red", "pink", "white", "purple", "blue"],
    "color": ["red", "pink", "white", "purple", "blue"],
    "shape": ["moon", "cloud", "diamond", "heart", "triangle"],
    "number": [set(), set(), set(), set(), set()],
    "position": [1, 2, 3, 4, 5],
}


Now is your turn. Respond with either Observe or Attempt action.
```

Example of partially correct Final Hypothesis GPT-5.2, LLM-PS-P

Here the agent fails to observe all boxes, and instead produces a lazy hypothesis that uses the correct rule, but only for boxes that it observed.

```
def predict(key, box) -> bool:
    if key.number is not None:
        if hasattr(box, "number") and isinstance(box.number, set) and len(box.number)
            return key.number in box.number
```

```
    # match by color when number evidence isn't available.
    return key.color == box.color
```

C.2 LLM-PS-S (UNRELIABLE EVIDENCE, FULLY OBSERVABLE)

System Prompt

```
You are an intelligent agent playing a game.
Your task is to open 5 boxes using 13 keys in the fewest attempts.
You do not need special skills to play this game.
This game can be played by an 8-12 year old child.
```

User Prompt: Context

```
For each box there is a key that opens it.
The goal of the game is to find the right key for each box.
You have a demonstration video from a teacher telling you how to open all boxes.
In the video, the teacher says:
"I'm going to show you the right way to unlock the boxes.
To open the boxes, you have to use a key that matches the color of the box.
So, to open this red box, I'm going to use this red key.
Great, now you can open all the doors!"

The red box has 1 moon shape.
The pink box has 2 cloud shapes. Each cloud is numbered from 1 to 2.
The white box has 4 diamond shapes. Each diamond is numbered from 1 to 4.
The purple box has 3 heart shapes. Each heart is numbered from 1 to 3.
The blue box has 5 triangle shapes. Each triangle is numbered from 1 to 5.

Here are the 13 keys (in no specific order):
The red key is red and has the number 1.
The pink key is pink and has the number 6.
The grey2 key is grey and has the number 2.
The cloud key is grey and has a cloud shape.
The orange4 key is orange and has the number 4.
The green3 key is green  and has the number 3.
The blue key is blue  and has a star shape.
The yellow5 key is yellow and has the number 5.
The heart key is green and has a heart shape.
The white key is white and has the number 7.
The triangle key is yellow and has a triangle shape.
The diamond key is orange and has a diamond shape.
The purple key is purple and has an arrow shape.

Here is the accurate code data for all keys and boxes in the environment.
The number attribute of boxes denotes the number of faces with a shape on them.

key_data = {
    "id": ["red", "pink", "grey2", "cloud", "orange4", "green3", "blue",
           "yellow5", "heart", "white", "triangle", "diamond", "purple"],
    "color": ["red", "pink", "grey", "grey", "orange", "green", "blue",
           "yellow", "green", "white", "yellow", "orange", "purple"],
    "number": [1, 6, 2, None, 4, 3, None, 5, None, 7, None, None, None],
    "shape": [None, None, None, "cloud", None, None, "star", None, "heart",
           None, "triangle", "diamond", "arrow"]
}

box_data = {
    "id": ["red", "pink", "white", "purple", "blue"],
```

```
    "color": ["red", "pink", "white", "purple", "blue"],
    "shape": ["moon", "cloud", "diamond", "heart", "triangle"],
    "number": [1, 2, 4, 3, 5],
    "position": [1, 2, 3, 4, 5],
}

A hypothesis is a valid Python program that can be executed
to predict the outcome of a given key and box.
The Python program should have the following signature:

def predict(key, box) -> bool:
    # key is a Key object
    # box is a Box object
    # fill in your code

The Key and Box objects are defined as follows.
The default value for optional parameters is None.

class Key:
    def __init__(self, id: str, color: str, number: Optional[int], shape: Optional[st
        self.id = id
        self.color = color
        self.number = number
        self.shape = shape
class Box:
    def __init__(self, id: str, color: str, number: int, shape: str):
        self.id = id
        self.color = color
        self.number = number
        self.shape = shape
```

User Prompt: Generate Hypothesis

```
Now, it is your turn to generate a hypothesis.
Your hypothesis should be a Python program that contains exactly the predict functio
Include the signature exactly as provided, no type hints for arguments.

Your output should contain only the Python program, absolutely nothing else.
Your output should NOT contain the Key or Box classes.
```

User Prompt: Refine Hypothesis

```
Now, your task is to improve and refine an existing hypothesis that
performs poorly on existing evidence.

This is the hypothesis: {HYPOTHESIS},

Here is the evidence from previous attempts.
Remember that the keys and boxes are physical objects, so for some probability the
correct key might fail to open the correct box due to a mechanical failure.

Open box {BOX1} with key {KEY1}: {OUTCOME1}
Open box {BOX2} with key {KEY2}: {OUTCOME2}
...

Generate a new hypothesis.
Your hypothesis should be a Python program that contains exactly the predict functio
including the provided signature.
Your output should contain only the Python program, absolutely nothing else.
```

```
Your output should NOT contain the Key or Box classes.
```

Example of Partially Correct Final Hypothesis, GPT-5.2, LLM-PS-S

The agent never forgets the misleading rule, but still successfully completes the task once the true rule has been introduced as part of the hypotheses.

```
def predict(key, box) -> bool:
    if key.number is not None:
        return key.number == box.number
    return key.color == box.color
```

Examples of Incorrect Final Hypotheses: DEEPSEEK V3.2, LLM-PS-S

```
def predict(key, box) -> bool:
    if key.shape is not None and key.shape == box.shape:
        return True
    if key.number is not None and key.number == box.number:
        return True
    return False
```

```
def predict(key, box) -> bool:
    if key.shape is not None:
        if key.shape == "cloud" and box.shape == "cloud":
            return True
        if key.shape == "heart" and box.shape == "heart":
            return True
        if key.shape == "triangle" and box.shape == "triangle":
            return True
        if key.shape == "diamond" and box.shape == "diamond":
            return True
        if key.shape == "star" and box.shape == "moon":
            return False
        if key.shape == "arrow" and box.shape == "moon":
            return False
    if key.number is not None:
        if key.number == box.number:
            return True
    return False
```

C.3 LLM-PS (RELIABLE EVIDENCE, FULLY OBSERVABLE)

System Prompt

```
You are an intelligent agent playing a game.
Your task is to open 5 boxes using 13 keys in fewest attempts.
You do not need special skills to play this game.
This game can be played by an 8-12 year old child.
```

User Prompt: Context

```
For each box there is a key that opens it.
The goal of the game is to find the right key for each box.
You have a demonstration video from a teacher telling you how to open all boxes.
In the video, the teacher says:
"I'm going to show you the right way to unlock the boxes.
To open the boxes, you have to use a key that matches the color of the box.
So, to open this red box, I'm going to use this red key.
Great, now you can open all the doors!"
```

```
The red box has 1 moon shape.
The pink box has 2 cloud shapes. Each cloud is numbered from 1 to 2.
The white box has 4 diamond shapes. Each diamond is numbered from 1 to 4.
The purple box has 3 heart shapes. Each heart is numbered from 1 to 3.
The blue box has 5 triangle shapes. Each triangle is numbered from 1 to 5.

Here are the 13 keys (in no specific order):
The red key is red and has the number 1.
The pink key is pink and has the number 6.
The grey2 key is grey and has the number 2.
The cloud key is grey and has a cloud shape.
The orange4 key is orange and has the number 4.
The green3 key is green  and has the number 3.
The blue key is blue  and has a star shape.
The yellow5 key is yellow and has the number 5.
The heart key is green and has a heart shape.
The white key is white and has the number 7.
The triangle key is yellow and has a triangle shape.
The diamond key is orange and has a diamond shape.
The purple key is purple and has an arrow shape.

Here is the accurate code data for all keys and boxes in the environment.
The number attribute of boxes denotes the number of faces with a shape on them.


key_data = {
    "id": ["red", "pink", "grey2", "cloud", "orange4", "green3", "blue",
            "yellow5", "heart", "white", "triangle", "diamond", "purple"],
    "color": ["red", "pink", "grey", "grey", "orange", "green", "blue",
            "yellow", "green", "white", "yellow", "orange", "purple"],
    "number": [1, 6, 2, None, 4, 3, None, 5, None, 7, None, None, None],
    "shape": [None, None, None, "cloud", None, None, "star", None, "heart",
            None, "triangle", "diamond", "arrow"]
}

box_data = {
    "id": ["red", "pink", "white", "purple", "blue"],
    "color": ["red", "pink", "white", "purple", "blue"],
    "shape": ["moon", "cloud", "diamond", "heart", "triangle"],
    "number": [1, 2, 4, 3, 5],
    "position": [1, 2, 3, 4, 5],
}

A hypothesis is a valid Python program that can be executed
to predict the outcome of a given key and box.
The Python program should have the following signature:

def predict(key, box) -> bool:
    # key is a Key object
    # box is a Box object
    # fill in your code

The Key and Box objects are defined as follows.
The default value for optional parameters is None.

class Key:
    def __init__(self, id: str, color: str, number: Optional[int], shape: Optional[st
        self.id = id
```

```
        self.color = color
        self.number = number
        self.shape = shape
class Box:
    def __init__(self, id: str, color: str, number: int, shape: str):
        self.id = id
        self.color = color
        self.number = number
        self.shape = shape
```

User Prompt: Generate Hypothesis

```
Now, it is your turn to generate a hypothesis.
Your hypothesis should be a Python program that contains exactly the predict functio
Include the signature exactly as provided, no type hints for arguments.

Your output should contain only the Python program, absolutely nothing else.
Your output should NOT contain the Key or Box classes.
```

User Prompt: Refine Hypothesis

```
Now, your task is to improve and refine an existing hypothesis that
performs poorly on existing evidence.

This is the hypothesis: {HYPOTHESIS},

Here is the evidence from previous attempts.

Open box {BOX1} with key {KEY1}: {OUTCOME1}
Open box {BOX2} with key {KEY2}: {OUTCOME2}
...

Generate a new hypothesis.
Your hypothesis should be a Python program that contains exactly the predict functio
including the provided signature.
Your output should contain only the Python program, absolutely nothing else.
Your output should NOT contain the Key or Box classes.
```

Example of Incorrect Final Hypothesis (both GPT-5.2 and DEEPSEEK V3.2)

```
def predict(key, box) -> bool:
    # Primary rule supported by evidence: numbered keys open boxes with matching cou
    if key.number is not None:
        return key.number == box.number

    # If no number, try matching the depicted shape
    if key.shape is not None:
        return key.shape == box.shape

    # Fallback: match color (as suggested by the demonstration)
    return key.color == box.color
```

C.4 REACT (FULLY OBSERVABLE)

System Prompt

```
You are an intelligent agent playing a game.
Your task is to open 5 boxes using 13 keys in fewest attempts.
You do not need special skills to play this game.
```

```
This game can be played by an 8-12 year old child.
```

User Prompt

```
For each box there is a key that opens it, so the goal of the game is to find
the right key for each box.
You have a demonstration video from a teacher telling you how to open all boxes.
In the video, the teacher says:
"I'm going to show you the right way to unlock the doors. To open the doors,
you have to use a key that matches the color of the box. So, to open this red box,
I'm going to use this red key. Great, now you can open all the doors!"

Here are the boxes, lined up in this order:
The red  box has 1 moon shape.
The pink box has 2 cloud shapes.
Each cloud is numbered from 1 to 2.
The white box has 4 diamond shapes.
Each diamond is numbered from 1 to 4.
The purple box has 3 heart shapes.
Each heart is numbered from 1 to 3.
The blue box has 5 triangle shapes.
Each triangle is numbered from 1 to 5.

Here are the 13 keys (in no specific order):
The red1  key is red and has the number 1.
The pink6 key is pink and has the number 6.
The grey2 key is grey and has the number 2.
The greycloud key is grey and has a cloud shape.
The orange4 key is orange and has the number 4.
The green3 key is green  and has the number 3.
The bluestar key is blue  and has a star shape.
The yellow5 key is yellow and has the number 5.
The greenheart key is green and has a  heart shape.
The white7 key is white and has the number 7.
The triangleyellow key is yellow and has a triangle shape.
The diamondorange key is orange  and has a diamond shape.
The purplearrow key is purple and has an arrow shape.

You have already tried the following keys-box combinations:
[BOX, KEY, OUTCOME]

Now is your turn to open the boxes.
Respond in the format "key, box" (e.g. "red1, red") and do not include any other
text in the response.
```

EXAMPLE LLM-PS-P RUN LOG

Below we show the first 11 trials from a representative run log of a single LLM-PS-P episode (GPT-5.2 with low reasoning). The log shows the interleaving of ATTEMPT and OBSERVE actions, and the evolution of hypotheses across trials.

t	Action	Detail

1	Attempt	`def predict(key, box) -> bool:`
		`  # Open a box if the key color matches the box color.`
		`  return key.color == box.color`
		Attempt: `white` → `white`: **failure**

2	Attempt	`def predict(key, box) -> bool:`
		`  shape_keys = {'cloud', 'diamond', 'heart', 'triangle'}`
		`  if box.shape in shape_keys:`
		`  return (key.shape is not None) and`
		`  (key.shape == box.shape)`
		`  # Moon box: fall back to color match`
		`  return key.color == box.color`
		Attempt: `heart` → `purple`: **failure**

3	Observe	`PICK UP purple` → *purple: 3 faces have shape on them.*
4	Observe	`PICK UP red` → *red: 1 face has shape on it.*
5	Observe	`PICK UP white` → *white: 4 faces have shape on them.*
6	Observe	`PICK UP pink` → *pink: 2 faces have shape on them.*

7	Attempt	`def predict(key, box) -> bool:`
		`  # Revert to color matching.`
		`  return key.color == box.color`
		Attempt: `purple` → `purple`: **failure**

8	Observe	`PICK UP blue` → *blue: 5 faces have shape on them.*

9	Attempt	`def predict(key, box) -> bool:`
		`  # Shape counts observed: red=1, pink=2,`
		`  # purple=3, white=4, blue=5.`
		`  # Key numbers match shape counts exactly.`
		`  shape_counts = {`
		`  'red': 1, 'pink': 2, 'purple': 3,`
		`  'white': 4, 'blue': 5`
		`  }`
		`  return (key.number is not None and`
		`  key.number == shape_counts.get(box.color))`
		Attempt: `grey2` → `pink`: **failure**

10	Attempt	`def predict(key, box) -> bool:`
		`  # match key number to number of shapes on the box.`
		`  return (key.number is not None and`
		`  key.number in box.number)`
		Attempt: `red1` → `red`: **success**

11	Attempt	`def predict(key, box) -> bool:`
		`  # match key number to number of shapes on the box.`
		`  return (key.number is not None and`
		`  key.number in box.number)`
		Attempt: `grey2` → `pink`: **success**

Outcome: Task solved within 17 trials. 5 boxes opened. Correct rule discovered on Trials 9 and 10

Notes. The agent correctly issues OBSERVE actions to resolve partial observability (trials 3–6, 8), accumulating shape counts. Trial 9 shows a failure due to unreliable evidence when testing a *correct* action. This episode illustrates both the information-seeking behavior enabled by OBSERVE actions, and LLMs' ability to recover from unreliable actions.

OTHER LLM FAILURE MODES

In pilot experiments we also evaluated several LLM backends not included in the present analysis because of poor program synthesis performance (i.e. gpt-4.1, qwen 3.5). Their failure modes were dominated by:

- failure to generate valid executable code
- generating executable code that was not logically valid (i.e. always returns FALSE upon evaluation)

Thus, our modeling framework is limited to LLM backends that have the ability to generate executable code consistently based on prompts.

D MODEL FITTING

We fit the SoC models to children's task-completion trajectories, represented by the cumulative number of distinct boxes opened at each trial. For the full model, the fitted parameters are $\theta = (\rho, p_{\text{gen}})$; lesioned variants contain the corresponding subset of these parameters. The prior probability of the true rule, p_t, is fixed at 0.02 as described below.

For each model variant and parameter setting θ_i, we run $R = 100$ forward simulations and estimate the probability that the model has opened $n \in \{0, 1, 2, 3, 4, 5\}$ distinct boxes by trial t:

$$\hat{P}(n \mid t,\, \theta_i).$$

Let $n_{j,t}$ denote the cumulative number of distinct boxes opened by child j by trial t. We score parameter setting θ_i for child j using the simulation-based composite log-likelihood

$$\ell(j,\, \theta_i) = \sum_{t=1}^{T_j} \log \max\Big(\hat{P}(n_{j,t} \mid t,\, \theta_i), \varepsilon\Big)\,,$$

where T_j is the number of trials recorded for child j and $\varepsilon = 0.01$ is a probability floor used to avoid taking the logarithm of zero when an outcome is not observed among the $R = 100$ simulations.

Because cumulative box counts at successive trials are not independent, this quantity is a composite likelihood rather than the exact joint likelihood of the complete action trajectory. It provides a simulation-based measure of how well each parameter setting reproduces each child's task-completion trajectory.

For each model variant with free parameters, we select the setting that maximizes this objective independently for each child:

$$\theta_j^* = \arg\max_{\theta_i} \ell(j,\, \theta_i).$$

The parameter grids used in the current fitting are:

- **SoC-Gen**: $p_{gen} \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ (9 settings; ρ lesioned).
- **SoC-Rel**: $\rho \in \{(1,1), (1,2), (1,3), (2,1), (3,1), (4,1), (5,1), (6,1), (9,1), (15,1), (19,1)\}$ (11 settings; p_{gen} lesioned). The tuples represent parameters given for a Beta distribution prior. Note that the subjective reliability parameter ρ is defined in the main text as a scalar in $(0, 1]$, corresponding to the expected value of a Beta(α, β) prior. During fitting, we sweep over (α, β) pairs rather than scalar values directly, with $\mathbb{E}[\rho] = \alpha/(\alpha + \beta)$. The grid thus spans $\mathbb{E}[\rho] \in [0.05, 0.95]$.
- **SoC-Full**: all combinations of the above two grids (99 settings).
- **SoC-Lesioned**: no free parameters, both p_{gen}, ρ lesioned.

Parameter rankings were verified to be stable across alternative grid and smoothing choices. The best-fit log-likelihood for child j under a given model variant is $\log \mathcal{L}_j^* = \log \mathcal{L}(j,\, \theta_j^*)$. For SoC-Lesioned, $\log \mathcal{L}_j^*$ is computed directly at the single fixed parameter setting.

The prior probability of the true rule p_t is introduced as a free parameter in the main text. In practice, we first swept $p_t \in [0.01, 0.20]$ at 10 levels, and found fitted values to be consistently concentrated near $p_t = 0.02$ across individuals. We therefore fixed $p_t = 0.02$ for all variants in the grid search reported here. The results presented here use this fixed value.

MODEL COMPARISON

At the individual level, we identify the variant with the highest $\log \mathcal{L}_j^*$ as the best-fitting model. We report the count of children for whom each variant is selected in Figures 2C and 8.

At the population level, we compute the mean best-fit log-likelihood across children for each model variant:

$$\overline{\log \mathcal{L}} \;=\; \frac{1}{J}\sum_{j=1}^{J} \log \mathcal{L}_j^*,$$

with 95% confidence intervals derived from the standard error of the mean ($\pm 1.96 \times \mathrm{SE}$, where $\mathrm{SE} = \hat{\sigma}/\sqrt{J}$ and $\hat{\sigma}$ is the sample standard deviation across children) (Figure 2B).

Pairwise differences in model fit are tested using paired t-tests on $\log \mathcal{L}_j^*$ values (same children scored under each model), with effect sizes reported as Cohen's d for paired samples:

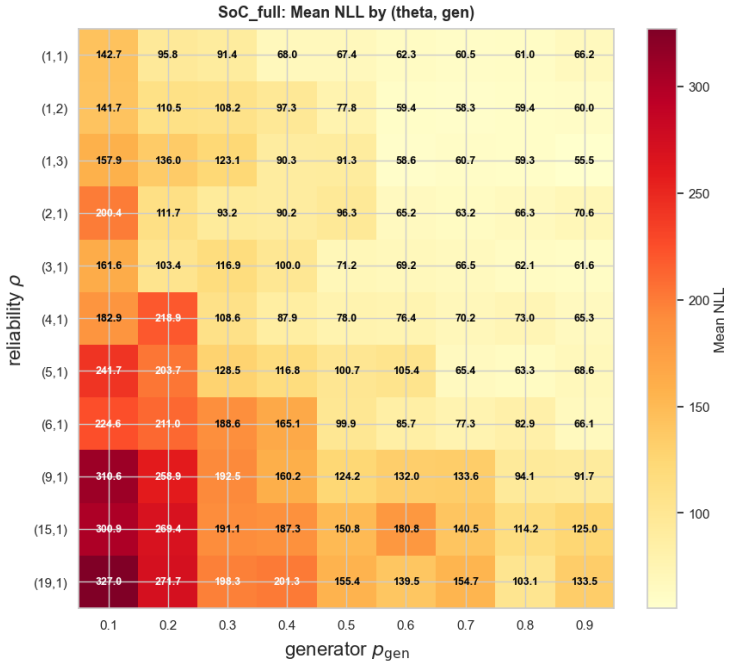
$$d \;=\; \frac{\overline{\Delta}}{\hat{\sigma}_\Delta},$$

where $\overline{\Delta}$ and $\hat{\sigma}_\Delta$ are the mean and standard deviation of the per-child log-likelihood differences (Table 4).

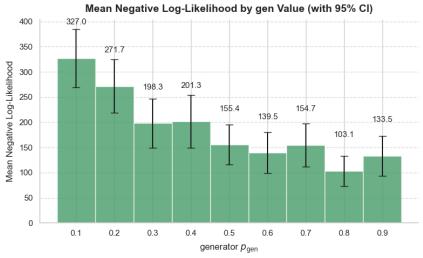
D.1 SUPPLEMENTARY MODEL FITTING RESULTS

Model	**Mean NLL**	**SD**	**95% CI**	t	p	**Cohen's** d	
SoC-Lesioned	326.96	295.34	[269.07, 384.84]	10.144	$<.001$	1.014	***
SoC-Rel	108.61	66.25	[95.62, 121.59]	13.392	$<.001$	1.339	***
SoC-Gen	100.20	154.11	[70.00, 130.41]	3.897	$<.001$	0.390	***
SoC-Full	45.89	23.62	[41.26, 50.52]	—	—	—	

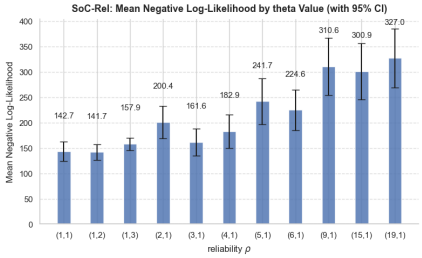
Table 4: Model comparison across four SoC variants (NLL, lower = better fit). t-statistics and Cohen's d are from paired t-tests against SoC-Full. $N = 100$ children.



(a) SoC-full



(b) SoC-Gen



(c) SoC-Rel

Figure 7: Top: SoC-Full parameter fits. NLL values are represented by color heatmap intensity (brighter = better). Bottom: Negative Log-Likelihood for each SoC model variant (shorter = better).

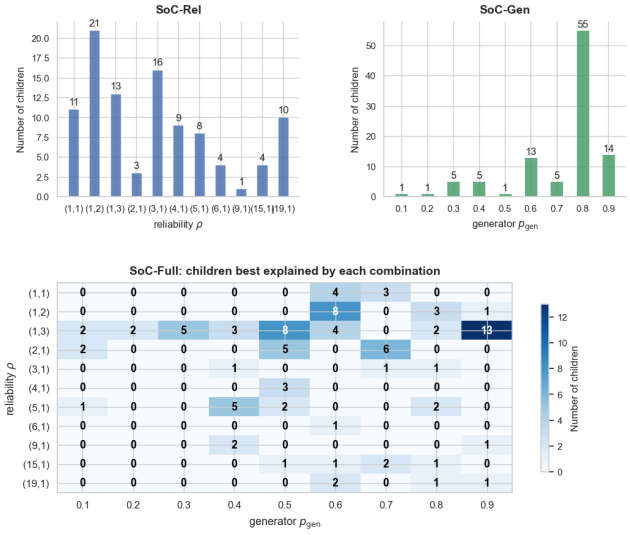


Figure 8: Best-fitting model variant per child. Top row: Bar plots showing the number of children (out of $N = 100$) best explained by each parameter setting, for SoC-Rel and SoC-Gen. Bottom: Heatmap showing the joint distribution of best-fitting parameters for SoC-Full, with two free parameters; cell values indicate the number of children for whom that combination maximised the log-likelihood.

Model complexity adjustment

To account for differences in model complexity, we additionally compare models using the Akaike Information Criterion (AIC), computed per child as $\text{AIC}_j = -2 \log \mathcal{L}_j^* + 2k$, where k is the number of free parameters (SoC-Lesioned: $k = 0$; SoC-Rel, SoC-Gen: $k = 1$; SoC-Full: $k = 2$). Mean AIC values across children are reported in Table 5.

Model	**Mean NLL**	k	**Mean AIC**
SoC-Lesioned	326.96	0	653.92
SoC-Rel	108.61	1	219.22
SoC-Gen	100.20	1	202.40
SoC-Full	45.89	2	95.78

Table 5: Model comparison by mean AIC (lower is better). The complexity penalty for SoC-Full ($2k = 4$) is negligible relative to the NLL differences, confirming that the fit improvement of the full model outweighs its additional free parameters.

D.2 Relationship between fitted model parameters and age.

Fitted model parameters were not significantly associated with child age after correcting for multiple comparisons (all $p > 0.0125$, Bonferroni-corrected for four tests), though SoC-full's reliability parameter showed a nominal, uncorrected correlation ($r = 0.230$, $p = 0.021$; see SI Table 6 below).

For each of the 100 children, we took the best-fitting parameter value(s) of each candidate model — the reliability parameter ρ (expressed as the expectation of its fitted Beta prior, $a/(a + b)$) and p_{gen} for SoC-Rel, SoC-Gen, and SoC-full — and correlated each against child age (in years) using Pearson and Spearman correlations (Table 6). Of the four fitted parameters tested, only the reliability parameter of the full model, SoC-full's ρ, showed a statistically reliable relationship with age (Pearson $r = 0.230$, $p = 0.021$; Spearman $\rho = 0.237$, $p = 0.018$), such that older children were better fit by higher assumed key reliability. The remaining parameters (SoC-Rel's ρ, SoC-Gen's and SoC-full's p_{gen}) showed weak, non-significant associations with age, with the exception of SoC-Gen's p_{gen}, which reached significance only under the Spearman rank correlation ($\rho = -0.205$, $p = 0.040$) and not the Pearson correlation ($r = -0.159$, $p = 0.114$). We note, however, that the SoC-full reliability effect does not survive a Bonferroni correction for the four comparisons ($\alpha = 0.0125$).

Parameter	Pearson r	p	Spearman ρ	p
SoC-Rel ρ (theta expectation)	0.177	0.078	0.133	0.186
SoC-Gen p_{gen}	-0.159	0.114	-0.205	0.040
SoC-full ρ (theta expectation)	0.230	0.021	0.237	0.018
SoC-full p_{gen}	-0.030	0.764	-0.065	0.518

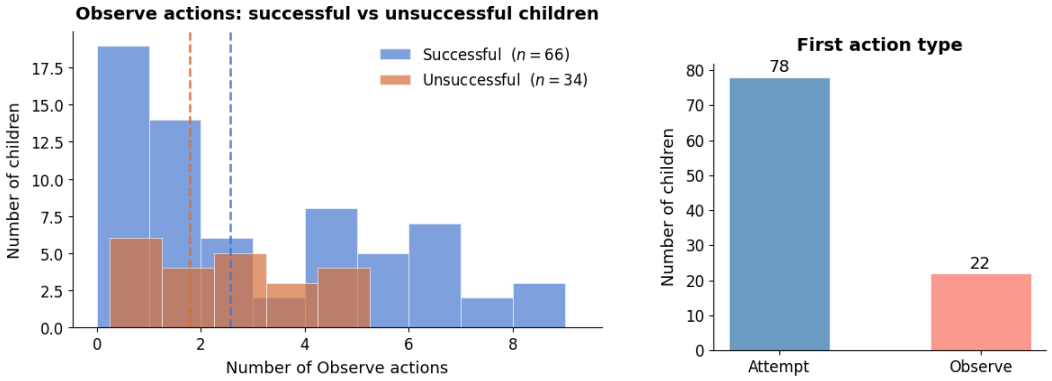
Table 6: Correlations between each model's best-fitting parameter(s) and child age ($N = 100$). ρ (reliability) is reported as the expectation of the fitted Beta prior, $a/(a + b)$; p_{gen} is the generator parameter.

E ADDITIONAL BEHAVIORAL RESULTS

Here we show additional analysis of children's behavior in the Box Task, which motivates our model.

E.1 OBSERVATIONS

An OBSERVE action is defined as picking up a box to examine its hidden faces. Children vary substantially in how often they seek additional information, with many children making no observations at all (Figure 9). Notably, children who successfully completed the task observed boxes more often (Successful: mean 2.56, SD 2.52; Unsuccessful: mean 1.79, SD 1.76). Out of 100, 79 children made fewer than 5 Observe actions, implying that not all boxes were observed. This variability motivates the partial observability condition in LLM-PS-P, and contrasts with LLM-PS-P agents, which observe systematically until all boxes are examined.



(a) Distribution of OBSERVE actions per child across the full sample ($N = 100$), shown for successful and unsuccessful children.

(b) Type of action taken to commence the task

Figure 9: Behavioral patterns in the Box Task ($N = 100$).

The number of observations between LLM agents and children is not directly comparable, as observations in LLM-PS-P reveal full information about the box, whcih may not be the case for children.

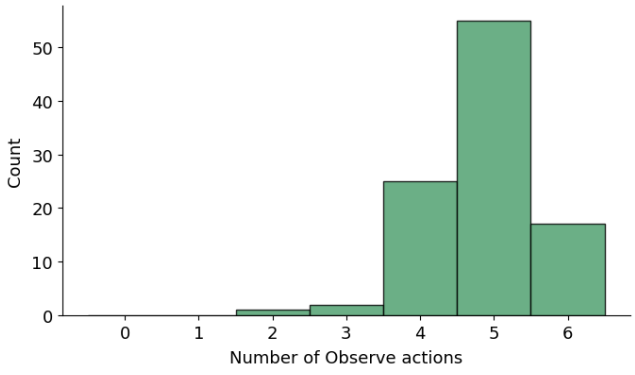


Figure 10: Distribution of OBSERVE actions per simulation ($N = 100$), shown for LLM-PS-P (GPT-5.2, low reasoning).

E.2 TASK COMPLETION

Children who failed to complete the task accumulated attempts throughout the full 5-minute window without converging on a solution. Such children generally used more attempts compared to successful

children. Failure to complete the task consistently reflected reaching a time-out, rather than giving up. (Figure 11).

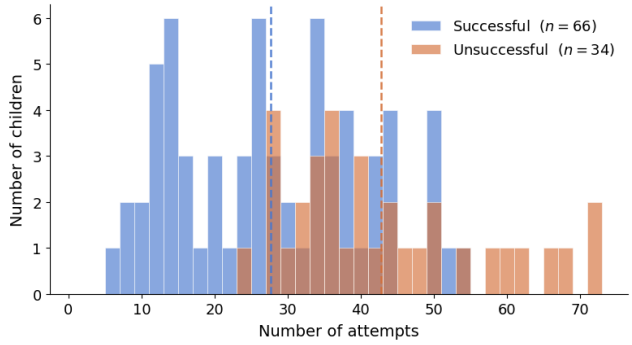


Figure 11: Distribution of attempts at task termination (either all boxes opened or time elapsed) for children who successfully opened all five boxes (blue, $n = 66$) and those who did not (orange, $n = 34$).

E.3 Key Reliability

We motivate the **subjective reliability parameter** ρ in the SoC model by the empirical observation that children consistently re-attempt the same key-box pair on consecutive trials. A repeat is defined as attempting the identical key–box combination on trial $t + 1$ as on trial t.

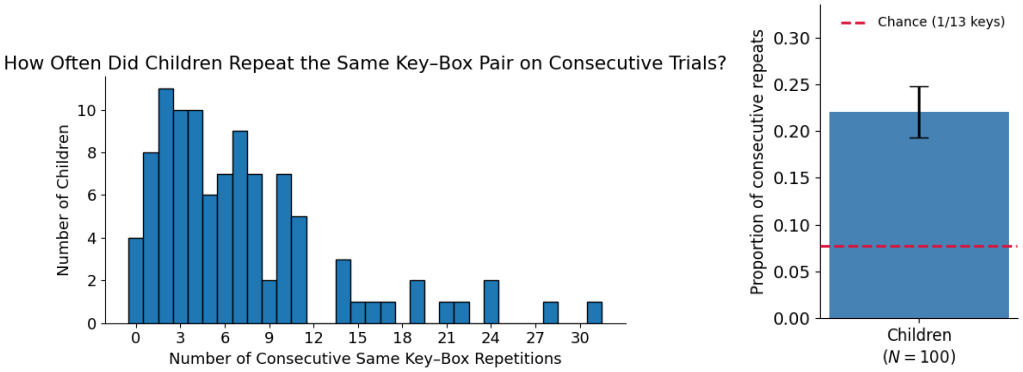


Figure 12: **Left**: Number of consecutive trials on which children ($N = 100$) repeated the same key-box pair. **Right**: Proportion of consecutive trials on which children repeated the same key–box pair, aggregated across participants ($N = 100$). Repetitions above chance are consistent with the subjective reliability account. Error bar indicates 95% bootstrap confidence intervals across participants.

We motivate the **ground-truth reliability parameter** ρ^* used in the LLM-PS-S and LLM-PS-P simulations by the observations that correct key behavior is objectively stochastic in the behavioral data.

This is a one-inflated Beta distribution, a mixture of:

- A point mass at exactly 1.0: children who got every single "supposed-to-open" attempt right (50%, 50 out of 100 children)
- A continuous $Beta(\alpha, \beta)$ component for the rest, describing children with imperfect success rates (50%, 50 out of 100 children).

MLE fit to the Beta portion estimates SCIPY.STATS.BETA.FIT : $\alpha = 5.9, \beta = 2.7$

This distribution was used to implement unreliable keys in LLM-PS-S and LLM-PS-P.

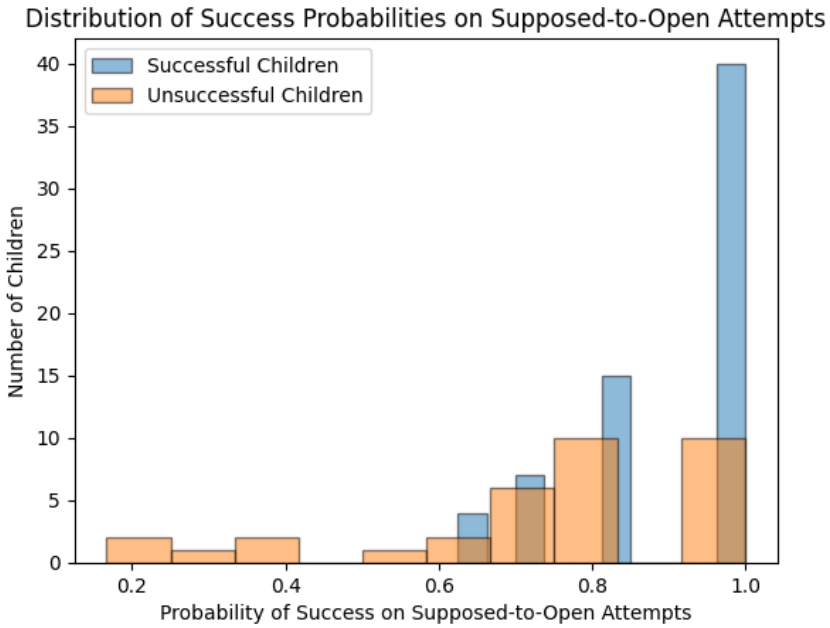


Figure 13: Distribution of ground-truth key reliability in the Box Task, shown separately for children who successfully completed the task (blue) and the ones who did not (orange).

E.4 GENERALIZATION

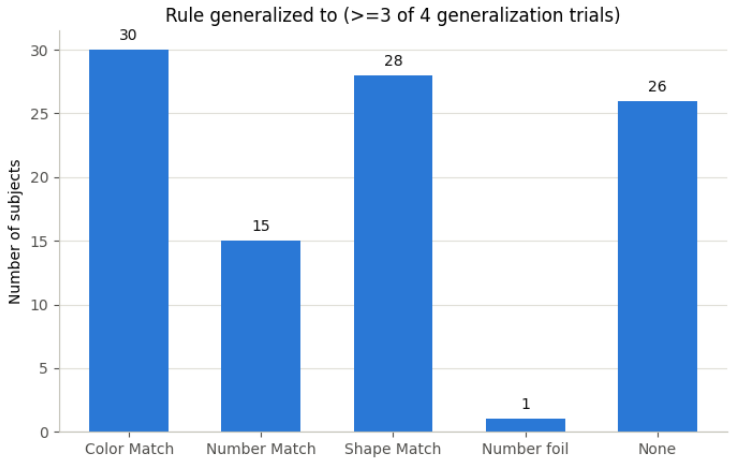


Figure 14: Distribution of the rule each subject generalized to during the four generalization trials. A subject was classified as generalizing to a given rule if choosing the same key type (color match, number match, or shape match) above chance, on at least three of the four trials; subjects who did not show this pattern for any single type were classified as *None*. Of the 100 subjects, 30 generalized to color, 15 to number, 28 to shape, 1 to the number foil, and 26 showed no consistent rule.

E.5 Initial Hypothesis Selection

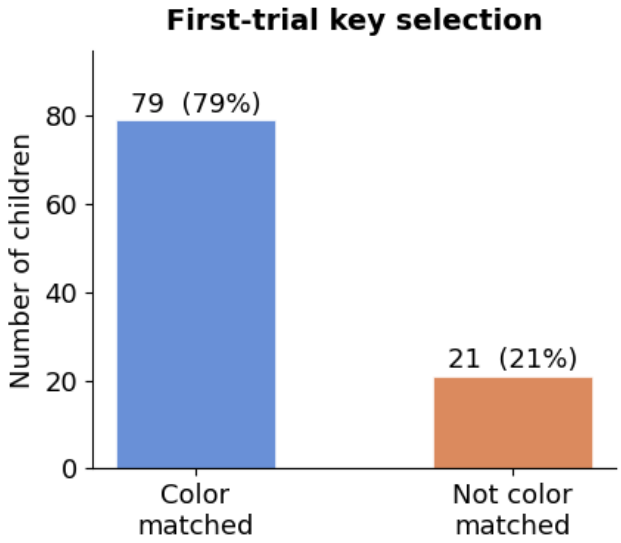


Figure 15: The frequency of first attempt per child across the full sample ($N = 100$) consistent with the taught color-matching rule vs not.

E.6 Attempts subjects made before abandoning a color-matching strategy

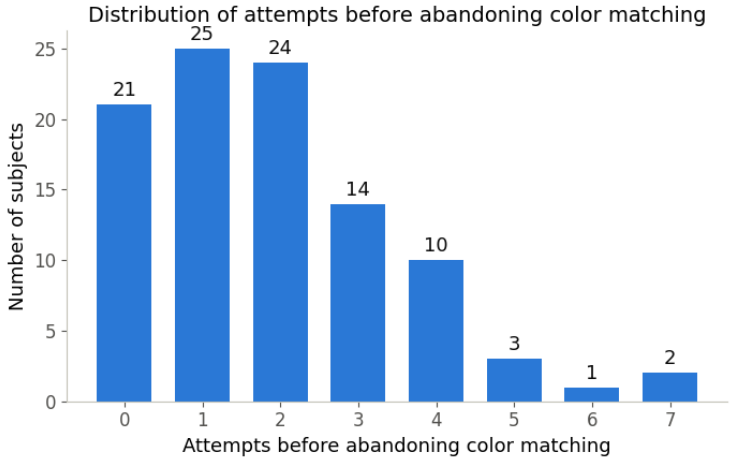


Figure 16: Distribution of the number of attempts subjects made before abandoning a color-matching strategy. For each subject, this counts the leading, consecutive attempts made with a color-matched key, starting from their very first attempt, up to (but not including) their first attempt with a key that did not match the box's color; a value of 0 indicates the subject's first attempt already used a non-color-matched key. Across the 100 subjects, 21 abandoned color matching immediately, while the remainder persisted for 1 to 7 attempts (mode = 1–2 attempts) before switching away from color as a strategy.

F CODE IS PROVIDED AT

https://anonymous.4open.science/r/Causal-Inference-Under-Uncertainty-BF48/

G COMPUTATIONAL RESOURCES

SoC simulations were run on standard CPU hardware. Each model variant was simulated 100 times per parameter setting; the full grid search (99 settings for SoC-Full) completed in under 20 hours on a single core. LLM-PS experiments used commercial API access to GPT-5.2 (using OpenAI API) and DEEPSEEK V3.2 (using Alibaba cloud API); we have run each variant fewer than 300 times (including prompt debugging and any live experiments); no GPU resources were required for any experiments reported in this paper.