ChatGPT

Bayesian Inference for Deterministic FSM Reconstruction

Problem Formulation and Prior Distribution

We consider an unknown deterministic finite-state machine (FSM) (a Moore machine) with **n** known states, an input alphabet of 6 symbols (e.g. doors labeled 0–5), and 2-bit output labels on states. This corresponds to a labyrinth of *n* rooms each having a label in {0,1,2,3} and 6 doors leading to other rooms [1]. We assume a **uniform prior** over all such automata. Formally, each of the *n* states is assigned an output label uniformly at random from {0,1,2,3}, and for each state and each input symbol (door 0–5) the next-state transition is chosen uniformly from the *n* states. Under this prior, every distinct FSM consistent with the given parameters is *a priori* equally likely. This can be viewed as treating each state's label and outgoing transitions as independent random choices (subject to the structural constraints of determinism). The uniform prior captures maximal initial uncertainty – we start with no bias favoring any particular state labeling or transition structure.

Bayesian Posterior Update from Observations

Each **observation** consists of an input sequence (a route plan of door numbers) and the corresponding output sequence of 2-bit labels seen as the machine transitions state-by-state [2]. For example, an input string "0325" might yield an output record like `[1,3,0,2,0]` (including the label of the starting state and after each move). Given a set of such observations, we update our belief over the automata using Bayes' rule [3]. Since the FSM is deterministic and observations are assumed noise-free, the **likelihood** of a candidate automaton *M* is 1 if *M* produces *exactly* the observed output sequence for each input sequence in the data, and 0 if it disagrees on any query. Thus, the posterior after observations is proportional to prior × likelihood, which in this case means it is uniform over all automata *consistent* with the observed input-output behavior (and zero for all inconsistent hypotheses). Intuitively, each new experiment (observation) rules out all automata that would have behaved differently, shrinking the set of possible models. In a Bayesian sense, we maintain a **posterior distribution** over the space of automata, concentrated on those that fit all data seen so far. As more observations are collected, this posterior concentrates on the true FSM (or the set of indistinguishably equivalent FSMs) assuming our prior included the true model.

In practice, representing the full posterior over all automata is intractable for large *n*, because the model space is enormous (on the order of $4^n \cdot n^{6n}$ possible machines). Instead, one can represent the posterior implicitly or approximately. For example, we might store the set of all remaining **consistent automata** symbolically (e.g. via constraints), or use sampling methods (described below) to approximate the distribution. The key update step for each new observation is to **eliminate** every hypothesized automaton that disagrees with the input-output trace, and leave the probabilities of the survivors proportional to their prior probabilities. With a uniform prior, this means assigning equal weight to all surviving hypotheses. If a non-uniform prior were used (say we had some bias about likely structures), Bayes' rule would also reweight by how likely each model made the observed outputs.

Selecting the Next Experiment – Information Gain

Given the current posterior uncertainty, we want to choose the next input sequence (experiment) *optimally* to gain information about the true FSM. A principled way is to maximize the **expected information gain** or **mutual information** between the experiment's outcome and the automaton model. In essence, we seek a query whose possible output results will best **split** the remaining hypotheses. This can be quantified by the reduction in posterior entropy:

- For each candidate input sequence q, consider all possible output sequences it could produce under different automata consistent with our current belief. Based on the current posterior, we can calculate the probability of each possible output for query q (by summing the probabilities of all automata that would yield that output).
- For each possible output o, we can determine the posterior (a subset of models) that would remain if o is observed (those automata consistent with o on input q). We can compute the entropy (uncertainty) of that posterior.
- The **expected entropy** after query q is the weighted average of the posterior entropy over all output outcomes (weighted by their probability). The **information gain** is the difference between the current entropy and this expected entropy [4]. Equivalently, this is the mutual information $I(\text{Model}; \text{Output}|q)$, the expected reduction in uncertainty about the automaton gained by observing the output of query q [5].

We then choose the input query that **maximizes this expected information gain**. In intuitive terms, this strategy favors experiments that, in the worst case, still significantly cut down the hypothesis space. Queries that would produce the same output for all remaining automata are useless (zero info gain), whereas a query whose outcome is uncertain (different models predict different outputs) and would drastically narrow down possibilities is highly informative. This is analogous to the 20-questions or decision tree strategy of asking questions that split the possibilities roughly in half (in terms of probability mass) [5].

Entropy-based criteria: One common approach is to maximize the **Shannon entropy** of the predicted output distribution for a query, which is equivalent to maximizing expected information gain when each output leads to a nearly deterministic identification of the model. Another related criterion is maximizing the expected **Kullback–Leibler divergence** between the prior and posterior (which in fact equals the mutual information). All these criteria are equivalent formulations of the same principle: prefer the experiment that best differentiates among the remaining hypotheses. In this discrete setting, it often means picking an input that *splits* the set of possible automata into outcome-specific subsets that are as balanced and as distinct as possible.

Approximation Strategies: Monte Carlo Sampling and Particle Filters

Computing the exact posterior or the exact information gain for each possible input can be computationally prohibitive. **Monte Carlo methods** are therefore employed to approximate these calculations. The idea is to represent the posterior distribution over automata by a **set of sampled automata (particles)** with associated weights, rather than enumerating all possibilities. Initially, we can sample a large number of automata from the prior (uniformly random FSMs with *n* states and given alphabet/output specs) and then **filter** out or down-weight those that conflict with observations. After each experiment, we resample the pool to focus on the surviving hypotheses. This is analogous to a *particle filter* for state-space models, except here the "state" we are estimating is the static automaton structure itself (or structure + current state) [6]. Each particle is one candidate FSM hypothesis, and it

carries a weight representing the posterior probability of that hypothesis. When a new input/output observation arrives, we update weights: any particle whose predicted output differs from the observation is assigned weight 0 (eliminated), and others keep their weight (or are reweighted if we had non-uniform priors). A resampling step can be used to concentrate the particle set on the high-weight hypotheses [7] [8]. This sequential Monte Carlo approach provides an **approximate Bayesian update** in a tractable way, at the cost of some approximation error and variance in the results. It is particularly useful if the number of surviving hypotheses is still enormous – by sampling, we only explicitly track a manageable subset of them.

When using such sampling, we can also approximate the **expected information gain** of a candidate query by Monte Carlo. Instead of summing over *all* automata, we simulate the query on each sampled hypothesis to see what output it would produce. This yields an empirical distribution of outputs for that query. From the sample, we can estimate the probabilities of each output and what the posterior (subset of particles) would be for each outcome. Then we compute the entropy reduction approximately. This Monte Carlo estimation of information gain can guide experiment selection without exhaustive enumeration of the hypothesis space.

Another possible approximation is to maintain a **factored or parametric representation** of uncertainty. For example, one might keep separate distributions for uncertain transition mappings or state label assignments. However, because the variables (transitions and labels) are coupled by the constraint of a single consistent automaton, factoring is non-trivial. Particle-based representation is flexible in that each particle implicitly carries a fully consistent set of transitions and labels.

A challenge with particle filtering in model space is **particle depletion** – a single unexpected observation can rule out the vast majority of sampled automata, leaving few survivors. To mitigate this, one can use targeted resampling or proposal distributions that prefer hypotheses already consistent with the data (e.g. generate new hypotheses by random modifications of surviving ones, rather than blind sampling). This is akin to **MCMC** (Markov chain Monte Carlo) or genetic algorithms in the space of automata: propose mutations to an existing FSM hypothesis (such as tweaking a transition or relabeling a state) and accept moves that maintain consistency with observations. Over time, this can explore the space of automata consistent with data. Such methods can approximate the posterior when direct sampling is ineffective.

Active Learning and Bayesian Experimental Design for FSMs

This reconstruction task is an **active learning** problem: the learner (William and Adso, in the story context [9]) can choose which input sequences (experiments in the labyrinth) to perform, in order to learn the model fastest. Bayesian optimal experimental design provides a powerful general framework for this: at each step, pick the experiment maximizing utility (information gain, as discussed). In practice, fully optimal selection may be intractable if we consider very long input sequences or need to do lookahead planning. Often a **greedy one-step lookahead** (choose the query with highest immediate expected info gain) is used, which is myopic but effective. This is analogous to strategies in active decision tree learning or 20-questions where each question is chosen optimally given current knowledge [5]. More advanced strategies could consider sequences of queries (multi-step planning), but the search space of query sequences grows quickly. In many cases, greedy selection is a reasonable heuristic that coincides with intuitive strategies domain experts use (e.g. “test this door because the models disagree on what happens there”).

Active automata learning has a rich literature of **algorithmic approaches** outside the Bayesian framework. For instance, Angluin’s L* algorithm and its variants actively learn a minimal DFA or Mealy

machine by making membership queries (feeding inputs) and conjecturing a hypothesis automaton, using counterexamples to refine it. Those algorithms aren't explicitly Bayesian (they don't maintain a probability distribution over hypotheses), but they similarly choose queries that distinguish hypothesized states or machines. A Bayesian view complements these by quantifying uncertainty and allowing non-deterministic strategies (like random sampling of hypotheses). In fact, many classical algorithms can be seen as implicitly performing an information-gain-driven search: e.g. asking queries to split equivalence classes of states.

Bayesian experimental design methods adapted to FSM inference would also consider the cost of experiments. In the ICFPC contest scenario, each query (expedition) has a cost and the goal is to minimize the number of queries [10]. An information-theoretic approach naturally aligns with this, seeking to maximize knowledge gained per query. If some experiments are more costly (e.g. longer paths might be limited [11]), we could incorporate a cost-weighted utility (information gain per step or per query).

Practical Implementation and ICFPC 2025 Context

ICFPC 2025 Task – "The Name of the Binding": The contest task essentially framed this problem as mapping an unknown labyrinth (finite-state graph) by querying an oracle with route plans [9]. In the **base version** of the task, although multiple rooms can share the same 2-bit label, it was guaranteed that the map could be identified without any extra tricks (the structure and label combinations were unique enough to distinguish each room by some route) [12]. A practical solution could be implemented by systematically exploring and merging states consistent with observations. For example, one could perform targeted explorations: try moving through certain door sequences to see if you return to a known room or discover a new one, thereby building the graph incrementally. Many teams likely used intelligent search strategies or active learning algorithms to minimize queries – effectively applying the above principles. For instance, if two different rooms both show label "2", the explorer would devise a plan to differentiate them (go out and back via a path that exploits a difference in their connectivity). This aligns with information gain: the query is chosen to tell those states apart.

In the **extended version** of the task (after the lightning round), the labyrinth introduced *indistinguishable subsets of rooms*: multiple rooms were exact duplicates of each other in structure and label, making it *impossible* to distinguish them by normal observations alone [13] [12]. To deal with this, a new action was allowed – **marking a room's label with chalk** (temporarily changing its 2-bit label) during a route plan [14]. This ability adds a form of **experimentation**: by relabeling one instance of an otherwise indistinguishable room, the explorer can later recognize if they encounter that same room again (seeing the marked label) versus another copy (which would still have the original label). In Bayesian terms, marking increases the observability of the system – it's like augmenting the experiment so that previously symmetric hypotheses yield different outcomes. The experimental design problem thus expands: now a "query" can include not just moving through doors but also strategically placing marks to break symmetry. The posterior update logic remains Bayesian (we eliminate any automaton hypothesis inconsistent with the sequence of observed labels, considering that we ourselves might have altered some labels en route). The information gain computation for a plan would take into account the possible outcomes of encountering or not encountering a chalk mark in certain positions. Essentially, marking is an action that ensures that two previously equivalent models (e.g. two maps where room A and room B were swapped) would produce different observable outcomes, thus allowing the learner to distinguish them. This mirrors techniques in active learning where the experimenter can intervene to perturb the system for disambiguation.

Known methods and tools: To implement these ideas, one could draw on algorithms from automata learning and model inference. For example, algorithms for learning Mealy/Moore machines from queries (such as those in the LearnLib library [15]) could be adapted. These algorithms systematically gather observations and construct a hypothesis machine, using additional queries to resolve uncertainties. While not overtly Bayesian, they are effective practical solutions to FSM reconstruction. On the Bayesian side, one could implement a custom particle filter or MCMC search over automata. However, because the hypothesis space is so large, a full Bayesian sampler might be less efficient than problem-specific heuristics. In practice, many solutions likely combined **systematic exploration** (to ensure every door from each discovered room is tried at least once, etc.) with **targeted tests** for ambiguities (to check if two paths lead to the same state or not). These can be seen as greedy information-gain-driven actions.

For the base task, a program could perform something akin to **incremental map building**: treat each unique sequence of outputs as potentially discovering a new room until a contradiction is found, and merge states when you determine two sequences actually led to the same room. For example, if entering door sequence X and sequence Y yield the same cycle of labels, you might deduce they ended in the same room. If uncertain, design another sequence that extends X and Y and see if the outputs diverge. This is effectively active automata identification. For the extended task with indistinguishable rooms, the program must incorporate the **marking action**. One practical approach is: when the mapping procedure finds that two hypothesized rooms remain interchangeable (all experiments so far can't tell them apart), use a mark – assign one a new temporary label and explore – to see if that mark is encountered again. The moment a mark "sticks" on a re-encounter, you know it's the same room you marked, resolving the ambiguity [16] [17].

In summary, reconstructing the FSM can be cast as a Bayesian inference problem where we maintain a distribution over possible automata and update it with each observation using Bayes' theorem [3]. A uniform prior over all *n*-state machines [1] reflects initial ignorance. Observations (input/output sequences) refine this to a posterior concentrated on consistent hypotheses. Active experiment selection via expected information gain guides us to the most informative input queries [5]. Because of the astronomically large hypothesis space, we rely on approximate methods like sampling (particle filters) [6] or heuristic search to represent and update our beliefs. These methods, combined with classic automata learning techniques, enable practical programs to infer the FSM efficiently, as demonstrated by solutions to the ICFPC 2025 labyrinth-mapping challenge in both its base and extended (with marking) versions. The theoretical Bayesian framework ensures that each step of exploration is justified in terms of reducing uncertainty, and the practical implementations show that these principles can be applied to successfully recover the hidden state machine model of the environment.

Sources:

- ICFPC 2025 Task Specification [1] [2] (deterministic labyrinth structure and observation model)
- ICFPC 2025 Addendum [12] [14] (identical room copies and the introduction of marking actions)
- Wikipedia: *Information Gain* [4] (mutual information and entropy reduction for query selection)
- Wikipedia: *Bayesian Inference* [3] (updating hypothesis probability with Bayes' rule)
- Wikipedia: *Particle Filter* [6] (using weighted samples to represent a posterior distribution)

---

[1] [2] [9] [10] [11] - https://icfpcontest2025.github.io/specs/task_from_tex.html

3 Bayesian inference - Wikipedia
https://en.wikipedia.org/wiki/Bayesian_inference

4 5 Information gain (decision tree) - Wikipedia
https://en.wikipedia.org/wiki/Information_gain_(decision_tree)

6 7 8 Particle filter - Wikipedia
https://en.wikipedia.org/wiki/Particle_filter

12 13 14 16 17 icfpcontest2025.github.io
https://icfpcontest2025.github.io/specs/addendum.pdf

15 A Quick Survey of Active Automata Learning # | Active-Automata-Learning
https://wcventure.github.io/Active-Automata-Learning/