Learning Automata from Positive Examples using OpenAI's GPT Models

William Bushell, Ian McQuillan

Abstract. As Large Language Models (LLMs) like OpenAI's GPT continue to demonstrate increasingly sophisticated capabilities, we begin asking questions about their ability to reason about formal systems. This project investigates whether OpenAI's GPTs can infer a deterministic finite automata (DFA) from only a limited set of positive examples, without access to the formal definition of the target language. Positive examples were generated from simple classical regular expressions (CREs), representing foundational patterns in regular language theory. For each CRE, GPT was prompted to construct a DFA that accepts the same language. The inferred DFAs were compared to the original using novel structural metrics: state-level precision and recall. These metrics reflect how much of the original language's structure was correctly captured (recall) and how often the inferred automaton accepted valid strings (precision). Results show that GPT is capable of approximating regular languages to varying degrees, with some DFAs matching perfectly and others exhibiting overgeneralization or overfitting. GPT-4o achieved the best overall performance among the models tested. This exploratory work suggests that LLMs can infer formal structures from data, though further refinement and constraints are needed to improve reliability and consistency.

Introduction

LLMs such as OpenAI's GPT series have transformed the landscape of machine learning, demonstrating broad capabilities in tasks ranging from natural language understanding to code generation. These models are trained on vast collections of human-generated data, allowing them to produce responses that often appear intelligent or structured. As their use expands across fields, questions arise about the extent to which they can engage with formal systems, especially those governed by strict, rule-based behavior like finite state automata.

Traditionally, the construction of DFAs requires a well-defined grammar or a complete specification of the language to be recognized. In this context, grammatical inference presents a

compelling challenge. While grammatical inference has been widely studied in theoretical computer science, applying it to LLMs introduces a new and largely unexplored dimension.

Unlike standard language models that focus primarily on linguistic coherence, OpenAI's o1 models (o1-mini and o1-preview) are specifically optimized for reasoning tasks [1, 2]. These models are trained and fine-tuned to handle structured problem-solving, logical deduction, and abstract pattern recognition. Rather than relying solely on surface-level statistical associations, reasoning-tuned models are designed to internalize causal relationships, hierarchical structure, and step-by-step inference patterns. This suggests a shift from language completion to structured reasoning, making them a natural subject for exploring whether LLMs can construct formal systems from limited data.

The goal is not to achieve perfect reconstruction, but to analyze how closely the inferred DFA aligns with the true language. This project introduces novel structural metrics to evaluate the inferred automata. These metrics allow for partial correctness and provide insight into how the model performs when reasoning about formal patterns. By applying this approach across multiple test cases and models, including GPT-4o and other recent variants, this study aims to shed light on how effectively LLMs can generalize structural rules from limited, example-based data.

This project explores whether an LLM can infer the structure of a regular language from a limited set of positive examples alone. Specifically, the model is given strings generated from simple CREs and asked to construct a DFA that accepts the same language. Crucially, the model is never shown the regular expression itself.

Background

A regular language is the simplest class in the Chomsky hierarchy of formal languages. Regular languages can be described using regular expressions and recognized by finite-state machines. They are widely used in applications such as text pattern matching and compiler construction due to their simplicity and efficiency.

A regular expression is a symbolic representation that describes a regular language. Classical regular expressions (CREs) are built using a small set of operators: concatenation (e.g., ab), union (+), and the Kleene star (*). These expressions can be converted into equivalent DFAs

using algorithms such as Thompson's construction, which shows that the expressive capabilities of CREs and DFAs is equivalent.

A deterministic finite automaton (DFA) is a formal machine defined by a finite set of states, an input alphabet, a transition function, a start state, and a set of final states. A DFA processes an input string one symbol at a time, transitioning between states according to its transition function. If the DFA ends in a final state after consuming the entire string, the string is said to be accepted by the DFA. Every regular language has a corresponding DFA that accepts it.

Grammatical inference is the process of constructing a formal grammar or automaton based on a set of sample strings from the language. Most traditional approaches require both positive and negative examples to avoid overgeneralization or overfitting. However, inferring a DFA from only positive examples, as explored in this project, is significantly more difficult and more relevant to real-world scenarios where negative data may be unavailable or hard to define.

Methodology

1. Selecting CREs

A curated set of simple CREs was used to define the source languages for testing. These CREs employed only basic regular operations: concatenation, union (+), and the Kleene star (*). Examples include patterns such as a*, a+b, ab*, and (ab)*. These expressions were chosen to reflect foundational patterns in regular language theory while remaining simple enough for human interpretability.

2. Generating Positive Examples

For each CRE, a small set of strings that belong to the language was automatically generated. These strings served as positive examples of the language. The generation process ensured that examples spanned typical usage of the CRE while avoiding repetition of any string. Generation worked by constructing a tree for each CRE, then traversing the tree to create positive examples. Since the language of a CRE can be infinite, we limited Kleene star repetitions to 3 or less. For example, the CRE a* yields these examples: ε, "a", "aa", and "aaa".

3. Prompting the GPT Model

Each set of positive examples was fed into an OpenAI GPT model with a prompt requesting it to construct a DFA that accepts all the provided strings. The model had no access to the original CRE or any structural information about the language. The prompt given for each CRE was: “You are a strictly mathematical model that outputs DFA definitions in a consistent single line with no formatting. For example, a DFA that accepts 'a' is defined as DFA = {{q0, q1}, {a}, {(0, a, 1)}, q0, {1}}. Final states must be integer indices of states. Here are some positive examples from a language, create a small DFA that randomly generates the language: ” Following this would be a list of positive examples representing the CRE.

4. Parsing the Generated DFA

The raw output from GPT was parsed into a formal DFA object using custom code to transform it into a FAdo DFA object. This process included extracting states, alphabet, transitions, start state, and final states from the output string, constructing the DFA using FAdo’s deterministic automaton API, and minimizing the DFA using FAdo’s minimalIncremental() method for consistent comparison

5. Comparing DFAs

To evaluate how closely the inferred DFA matched the original, the original DFA (constructed from the CRE using str2regexp().toDFA()) was compared to the predicted DFA using three additional DFAs:

True Positive (TP) DFA	Accepts strings accepted by both the original and the predicted DFA
False Positive (FP) DFA	Accepts strings accepted by the predicted DFA but not the original
False Negative (FN) DFA	Accepts strings accepted by the original DFA but not the predicted one

6. Calculating Structural Metrics

The DFAs were evaluated using two custom metrics:

State-level precision = $\frac{TP\ states}{TP\ states + FP\ states}$

State-level recall = $\frac{TP\ states}{TP\ states+FN\ states}$

These metrics assess how much of the original DFA's behavior was preserved and whether the inferred DFA generalized beyond the target language.

Results and Discussion

The evaluation focused on comparing the performance of four GPT models: GPT-4o, GPT-4o-mini, o1-mini, and o1-preview. Each model was provided with positive examples and prompted to infer a DFA that accepts the same language.

Overview of Test Data

The test suite contained 46 CREs. CREs are ordered in ascending complexity, as shown. The table below shows the categories that the CREs fall into:

1-3	Single symbol (e.g. a)
4-10	Concatenation (e.g. ab)
11-13	Union (e.g. a+b)
14-16	Kleene star (e.g. a*)
17-37	Concatenation and Kleene star (e.g. a*b)
38-46	Union, and Kleene star (e.g. a+b*)

Summary Statistics

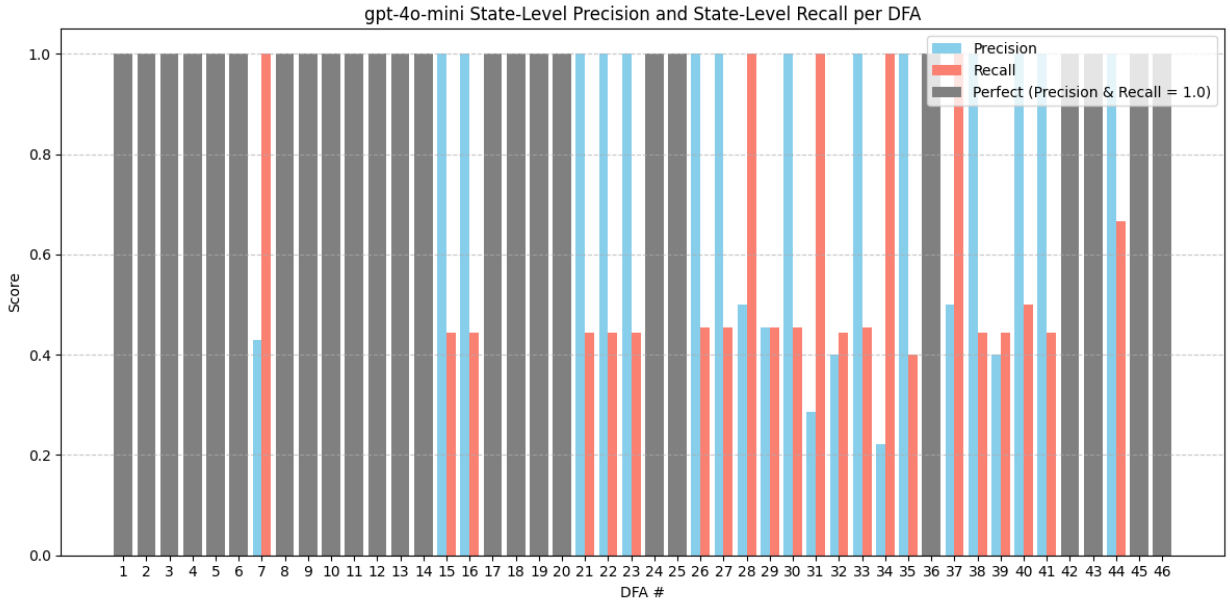
46 Total DFAs	**GPT 4o mini**	**GPT 4o**	**o1 mini**	**o1 preview**
Perfect DFAs	52% (24)	80% (37)	72% (33)	76% (35)
Avg. Precision	0.895	0.925	0.974	0.989
Avg. Recall	0.801	0.941	0.855	0.879
Avg. Imperfect Precision	0.781	0.617	0.908	0.955
Avg. Imperfect Recall	0.584	0.699	0.488	0.495

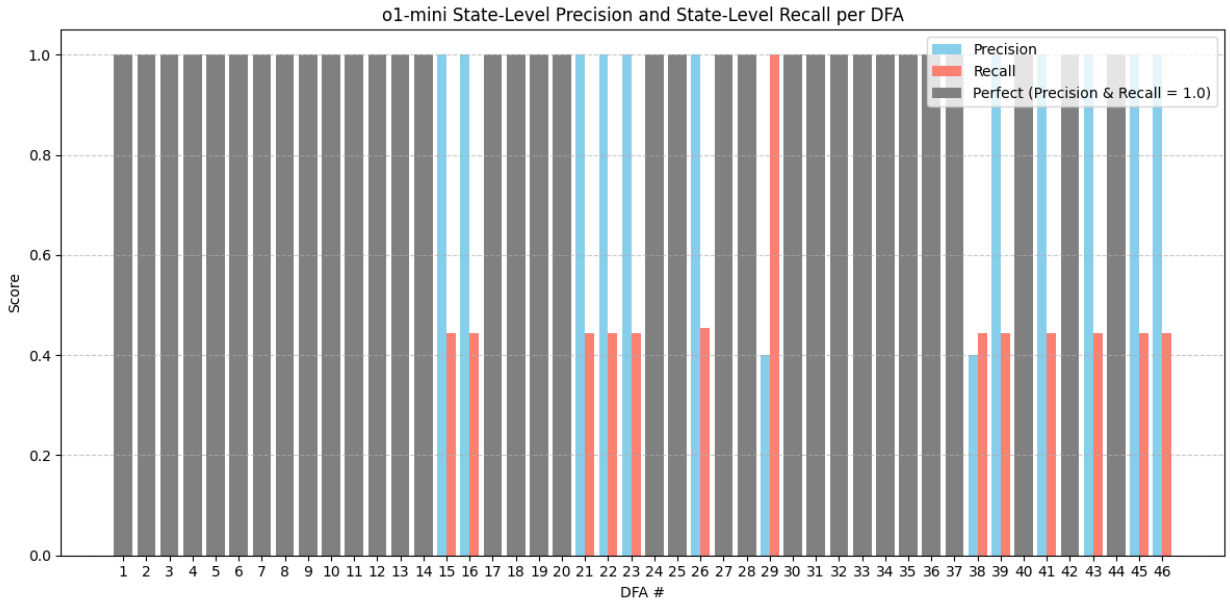
Perfect DFAs are those for which both precision and recall equal 1.0, meaning the predicted DFA exactly matches the original.

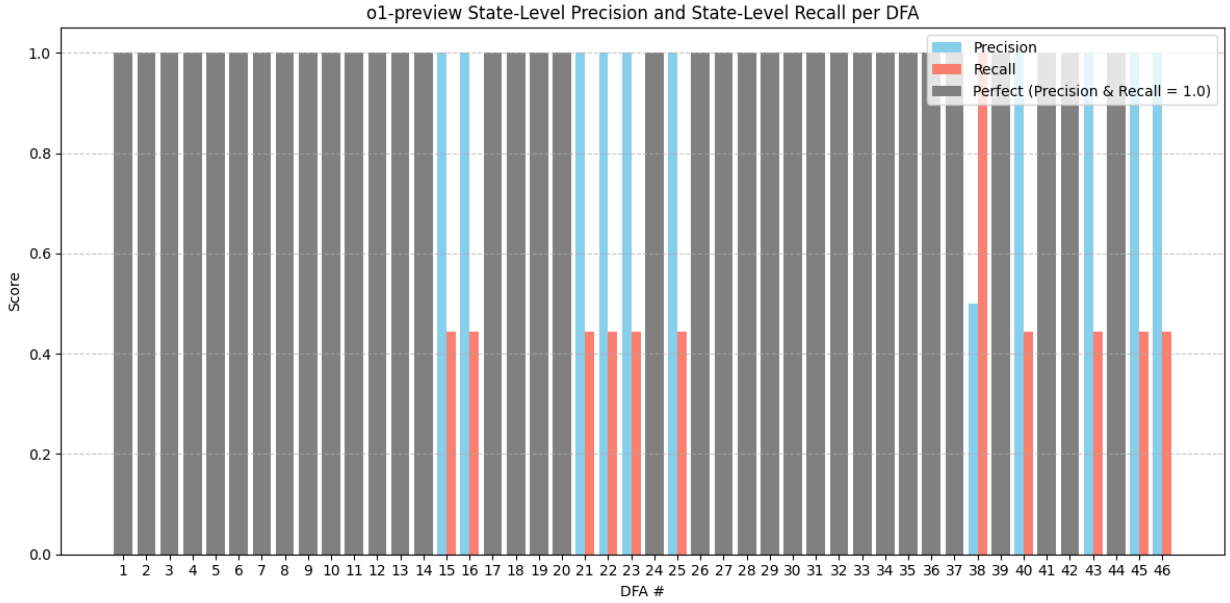
Average imperfect precision and recall only considers imperfect DFAs in the calculation (DFAs with one of state-level precision or recall less than 1.0).

Visualization

Bar graphs were used to visualize state-level precision and recall per DFA, highlighting trends across test cases.







Model Performance

GPT-4o produced the highest number of perfect DFAs, with high precision and recall across most CREs. GPT-4o-mini was less consistent but still competitive. o1-preview showed high precision but low recall, suggesting a tendency to overfit to the positive examples given. o1-mini exhibited similar behaviour to o1-preview, but with slightly lower precision.

It's fair to conclude that GPT-4o, o1-mini, and o1-preview all had similar performances, while GPT-4o-mini had a significantly weaker performance in these metrics.

We can add state-level precision and recall scores together to get the average performance of each model:

GPT-4o-mini	1.696
GPT-4o	1.866
o1-mini	1.829
o1-preview	1.868

Across multiple test cases, GPT-4o stood out as the most consistent and accurate model, and generally had the best performance. It had a language equivalence success rate of 80%, meaning it only got 20% of the examples wrong. The reasoning models, o1-mini and o1-preview, had language equivalence success rates of 72% and 76% respectively. GPT-4o-mini had a poor language equivalence success rate of 52%.

When we compare the bar graphs of GPT-4o to o1-mini and o1-preview, we can see that GPT-4o gets the first 27 DFAs correct, but has weaker performance with the latter half of the CREs. The reasoning models have trouble with some of the earlier, but have better performances with the later CREs. Since the later CREs are more complex, the reasoning models do better with the complex languages but worse with the simpler compared to GPT-4o.

However, the study also revealed two major failure modes: overgeneralization, where the inferred DFA accepted strings far beyond the intended language, and overfitting, where the model reproduced only the provided examples without generalizing to the full language.

These failure patterns are consistent with those found in recent work [3, 4]. Vazquez-Chanlatte et al. (2024) observed similar overgeneralization behaviors in their L*LM framework when models were not sufficiently guided by demonstrations. Similarly, Chen et al. (2024) showed that even with structured membership queries, LLMs may hallucinate answers or fail to generalize without reinforcement through feedback. While their methods rely on interactive querying and language scaffolding, this project reveals that such failure modes also emerge in passive, one-shot inference setups.

Both of these issues reflect the lack of constraint in the model's inference process, an expected consequence of working exclusively with positive data. The structural metrics (state-level precision and recall) introduced in this project provided an effective way to quantify these behaviors and offered a more nuanced alternative to simple binary equivalence checks.

Conclusions

This research demonstrates that LLMs, specifically OpenAI's GPT models, show promising capabilities in performing automata inference using only positive examples. Given a small set of strings generated from simple CREs, the models were able to construct DFAs that, in many cases, closely approximated the intended language. These findings complement recent research on LLM-assisted automata learning. While prior work such as L*LM and pMAT demonstrates the benefit of prompt engineering and interactive refinement, this project isolates the capabilities of LLMs when no such support is provided. In doing so, it highlights both the promise and the current limits of LLM-based grammar inference. In summary, while GPT is not yet a reliable method for full automata induction, it has demonstrated emergent competence in identifying structural patterns within formal languages. This project illustrates a meaningful step toward integrating LLMs into more formal domains and offers tools for evaluating such models.

Future Directions

Expanding the Language Class

The current test suite was limited to simple CREs. Future work could extend this to more complex expressions, including nested alternations, optional symbols, and larger alphabets. Exploring inference on more expressive language classes, such as context-free languages, could also test the limits of GPT's formal reasoning.

Improving Prompts and Model Guidance

Prompt design played a key role in how effectively the model inferred DFAs. Further work could focus on engineering prompts that scaffold the model's reasoning, for example, by

breaking down the example set into transitions or asking for a transition table directly. Fine-tuning LLMs on automata-specific tasks may also yield performance improvements.

Robustness

GPT's outputs can vary between queries even when given the same input. Exploring how consistent its DFA inference is across multiple generations could help identify instability or biases in the model's behavior. Establishing benchmarks and test suites for automata inference tasks would also support replication and comparison across models.

Formal Verification

Integrating formal tools that can verify language equivalence between the inferred and original DFAs would provide stronger guarantees of correctness. Although this project used structural metrics, future studies could explore bounded or symbolic equivalence checking for a deeper analysis.

References

[1] https://platform.openai.com/docs/models

[2] https://platform.openai.com/docs/guides/reasoning

[3] Vazquez-Chanlatte, Marcell, et al. *L∗LM: Learning Automata from Examples Using Natural Language Oracles*. 2024, arXiv:2402.07051. arXiv, https://arxiv.org/abs/2402.07051.

[4] Chen, Lekai, et al. *LLMs as Probabilistic Minimally Adequate Teachers for DFA Learning*. University of Colorado, Boulder, 6 Aug. 2024. arXiv, https://arxiv.org/abs/2408.02999.