Boolean Minimization: Are LLMs glorified pattern matchers?

Ashish Pandian
University of California, Berkeley
Berkeley, California, USA

Yongshi Zhan
University of California, Berkeley
Berkeley, California, USA

Alberto Hojel
University of California, Berkeley
Berkeley, California, USA

Abstract

Efficient hardware design critically relies on the minimization of Boolean functions, a task that can lead to significant reductions in logic gate counts, power consumption, and system latency. Recent advancements in Large Language Models (LLMs) have shown potential in complex reasoning and problem-solving tasks, yet their efficacy in multi-step deliberative tasks like Boolean minimization remains less explored. In this study, we assess the performance of LLMs, particularly focusing on combinational logic, to discern whether they act as mere pattern machines or possess the capacity for strategic forward-looking reasoning. Our methodology involves extensive prompt engineering, inspired by recent advancements in optimizing LLM outputs, and fine-tuning techniques on custom datasets. We evaluate prominent models including ChatGPT and Llama2 across various complexities of Boolean expressions. Our findings reveal that while LLMs like ChatGPT exhibit moderate success in zero-shot learning scenarios, their performance significantly improves with fine-tuning, as demonstrated by the Llama2 model, which shows a high accuracy in simplifying Boolean expressions across large and extreme-sized datasets. This research not only sheds light on the reasoning capabilities and limitations of LLMs but also suggests the utility of fine-tuning in aligning model outputs with deterministic logical tasks, marking a step towards the deployment of LLMs in specialized computational domains.

Keywords: Large Language Models, Boolean Expression, Hardware Design, State tracking, Reasoning,

ACM Reference Format:
Ashish Pandian, Yongshi Zhan, and Alberto Hojel. 2018. Boolean Minimization: Are LLMs glorified pattern matchers?. In . ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

, ,

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Boolean function minimization is an important task in efficient hardware design. Reducing the complexity of Boolean expression can directly translate to a reduction in logic gate counts, power consumption, and latency of hardware systems. Large Language models have demonstrated impressive performance on a range of reasoning and problem solving tasks. However, their ability to perform multi-step deliberative tasks like Boolean minimization is still been explored. In this work, we plan to conduct an assessment of LLMs' performance on the simplification of Boolean logic, specifically combinational logic. We demonstrate LLMs' capabilities to simplify Boolean expressions to evaluate if they are general pattern machines or if they have the ability to strategically look ahead and reason through complex tasks. Our results demonstrate their strengths and weaknesses in boolean expressions manipulation by modeling their truth tables to understand their state tracking abilities. Our analysis provides insight into the reasoning capabilities of LLMs and highlights areas of weakness. We aim to use boolean minimization as a way to demonstrate the underlying flaws and strengths of LLMs.

2 Background

Boolean function minimization is vital for efficient hardware design. By minimizing the complexity of Boolean expression, it can directly enable the reduction of logic gate counts, latency, and power consumption of hardware systems. Additionally, it can result in reduced costs and more energy efficient circuits. Boolean circuit complexity scales exponentially as they have multi outputs and hundreds of inputs. The Boolean minimization algorithm must be able to balance having fast time complexity while still maintaining close to perfect levels of accuracy. Additionally, we want to focus on algorithms that scale efficiently as real-world IC designs are getting more complex by the day. As we will see, most Boolean minimization algorithms have tradeoffs between these 4 characteristics: time complexity, accuracy, minimization, and scalability. We will cover some background before explaining our methodology.

2.1 History of Boolean expressions

The history of boolean expression minimization dates back to early techniques that focused on small scale expressions. Karnaugh Maps [3] and Quine-McCluskey algorithm [2] can

optimally solve small scale problems, but become intractable as the number of active variables increases.

Heuristic algorithms can handle larger input sizes. These include espresso algorithms that utilize heuristics to explore the space and logic expressions providing strong minimization within a reasonable time. SAT solvers are able to give simplifications of large boolean expressions, but due to the format these problems, which are NP-Hard, they thus struggle to scale with multiple outputs. They also lack optimal guarantees that were seen in the smaller scale algorithms.

Recently, more work has been done with machine learning approaches such as genetic algorithms. Genetic algorithms have a population of possible boolean expressions and by applying mutations and crossovers over generations they are able to get a accurate logical boolean expressions. These methods suffered from a lot of finetuning of extensive hyper parameters which may overfit certain formats of boolean expressions.

2.2 Boolformer

Boolformer [1] demonstrates the promise of transformers for boolean minimization. It is the first end-to-end transformer model that simplifies Boolean expressions. Boolformer was trained on 300,000 truth table and simplified function pairs dataset and was able to see close to 100 percent performance on tasks that were dealing with up to 6 variables. When pushing further, at 10 variables, the performance dropped to 80% accuracy. We explored some of the possible directions of their model as preliminary work to understand the field of boolean minimization.

In preliminary work, we explored the extension of Boolformer for handling complex logic fates like NAND and XOR. These extensions achieved strong performance in generating simplified boolean expressions. Our results demonstrate that there was only a 7.37 percent decrease in the accuracy of simplified boolean expression compared to the benchmarked Boolformer without NAND/XOR. However, one concern is that there is a drastic drop in performance on the accuracy of boolean expressions for boolean expressions with more than the max of 8 active variables which their dataset tested. These results are concerning as it show that it is not able to generalize to larger boolean expression sizes due to their full attention model complexity that grows quadratically, which makes it hard to demonstrate it as a viable future for Boolean minimization. The scalability concerns, long training time, and out of distribution concerns were some issues we noticed with transformers when faced with the task of Boolean minimization.

One of the key contributions of Boolformer is their novel method for controlling diverse generations of boolean functions. They leverage the creation of a random binary tree, replacing leaf nodes with input variables and branch nodes with gates, to create the initial functions. By utilizing deep trees, we can generate functions with many nested components making them ripe for simplification. Afterwards, they leverage the boolean.py 4.0 package to perform simplification with DeMorgan and other boolean algebra laws. In the end, they created a pipeline for generating highly redundant and unsimplified functions through the use of the binary trees, and their simplified counterpart through the use of the python library. Their methodology provides a large dataset for us to train and evaluate models on the task of boolean expression minimization. In our work, we adopt this data generation process, but instead split their data into 2 sets. We performed 2 different part of experiments. The first are focused on starting with redundantly long boolean expression to simplified boolean expressions. The second part of the experiments are going from truth tables to any form of logically equivalent boolean expressions. We utilize the key concepts of this paper to create our dataset that relies on tree generation.

2.3 Large Language Models

Large language models are advanced artificial intelligence systems with notable ability to understand and generate human language. These models are built on deep learning architectures, specifically transformer architectures, being pretrained using supervised learning. Two notable examples of large language models are ChatGPT and Llama2. By representing boolean expressions using characters, it is possible for LLMs to tokenize its inputs and even capture some of the logic in boolean functions.

ChatGPT and Llama 2 are built on the transformer architecture and are trained on a diverse range of text to generate coherent and contextually relevant responses to user inputs. Pre-training exposes these model to vast amounts of data, allowing them to learn grammar, facts, reasoning abilities, and even some level of common sense. Developed by OpenAI, ChatGPT is part of the GPT (Generative Pre-trained Transformer) family of models. With an ability in zero/few shot learning, ChatGPT 3.5 can produce decent responses being given very few examples. Therefore, this project chose to use ChatGPT 3.5 Turbo to test the performance of LLMs without fine-tuning on the task of boolean expression minimization and assess the effects of few shot examples have on its performance. Llama 2 [7] is an open source large language model developed by Meta AI that has 3 model sizes (7B, 13B, 70B) and allows users to perform fine tuning. Due to the open source nature of Llama 2, we used it to demonstrate the effect of fine tuning on the ability of LLMs solving the problems of boolean logic minimization. This allows us to perform three different tests: small open-source model performance with original Llama2-7B, large API-based model performance with and without in-context learning, and finally fine-tuned small open-source model by training Llama2-7B on a dataset of boolean simplification.

2.4 Autoregressive Behaviors

Autoregressive language models, such as those utilized by OpenAI and Anthropic, generate text by modeling the probability distribution of a sequence, predicting each subsequent token based on the preceding context [5]. This capability enables them to perform reasoning tasks by sequentially building upon prior outputs. Investigations into the state-tracking ability of these models, as exemplified by studies on language-based chess play [6], demonstrate their potential for complex problem-solving. These studies typically focus on the efficacy of fine-tuning approaches in teaching models to accurately learn and execute specific tasks.

OpenAI's ChatGPT leverages Reinforcement Learning from Human Feedback (RLHF) [4], a technique that combines machine learning algorithms with human feedback to enhance the model's ability to engage in human-like reasoning. This approach is instrumental in guiding the model towards outputs that are more aligned with human expectations and norms. Anthropic's Claude model adopts a similar strategy, integrating Reinforcement Learning (RL) with human feedback to align the model's responses closely with human values and preferences.

The relevance of such techniques is particularly pronounced in tasks that demand not creative generation, but the execution of deterministic logic with singular correct outcomes. In the context of Boolean function minimization, the objective is clear-cut: to arrive at the simplest equivalent expression. Thus, the challenge lies not in generating a plurality of responses but in ensuring the accuracy and precision of the singular solution.

Recent work extends the application of AI feedback mechanisms beyond fine-tuning, incorporating them into the inference process of LLMs [9]. This development hints at the possibility of LLMs continuously refining their problem-solving processes, even post-fine-tuning. Such ongoing improvements could be instrumental in tasks like Boolean function minimization, where precision and the ability to correct course based on feedback are crucial.

These advancements in autoregressive modeling and feedback-informed learning represent a significant leap towards models that can effectively emulate human-like problem-solving abilities. They mark a transition from generative capacities to the execution of tasks governed by strict logical rules, aligning model outputs with the deterministic nature of such problems.

3 Methodology

3.1 Dataset Generation

By adopting the data generation methodology from Boolformer [1], we generated a dataset of unsimplified combinational logic functions with various input sizes, along with the minimized equivalent representations of the functions. Based on the size of the input dimensions D of each boolean function, the functions generated were divided into 4 different categories, each category having a different number of maximum binary gates Bmax. The number of binary gates B in a boolean expression were sampled uniformly between [D - 1, Bmax], each gate being selected from AND, OR with equal probability. We then generate a random binary tree with Bmax branching nodes. For each leaf node, we assign it an input variable identity sampling form the input dimension D. For each branching node, we assign it a gate sampled from AND, OR as described above. Finally, NOT gates are prepended to the gates with a probability of 0.5 independently for each branching node. The resulting boolean expressions were then simplified using combinational logic minimization rules.

The dataset of small-sized functions had an input dimension D of 2 with a Bmax of 3. The dataset of medium-sized functions had an input dimension D of 4 and a Bmax of 6. The dataset of large-sized functions had an input dimension D ranging from 6 to 8 and a Bmax of 10. The dataset of extreme-sized functions had an input dimension D ranging from 8 to 20 and a Bmax computed for each function with "round(1.5 * D)".

3.2 Prompting Methodology

The approach to prompting Large Language Models (LLMs) is a nuanced art that marries the complexity of language with the precision of logical tasks. Prompt engineering, the process of meticulously crafting the input provided to an LLM, is pivotal in guiding the model towards generating accurate and relevant outputs. Throughout this project, we have invested significant efforts in refining our prompting techniques to optimize the performance of LLMs on Boolean expression simplification tasks.

Initial experiments revealed that the accuracy of LLMs can vary dramatically based on the structure and content of the prompt. To address this, we adopted a systematic approach to prompt engineering, drawing on recent findings that suggest instructing the model to "take a deep breath" can induce a more thoughtful and measured response [8]. This metaphorical instruction is intended to prime the model for a step-by-step reasoning process, mirroring human cognitive strategies for dealing with complex problems.

> "(Here are number correct simplified boolean expressions: Examples of combinational logic minimization)
> Now, take a deep breath, think step by step, and simplify this boolean expression input: Boolean expression to the simplest form. It is IMPERATIVE that the simplified boolean expression be logically equivalent to the input, otherwise you are going into a timeout."

This prompt exemplifies the initial phase of our project, where the model is provided with a set number of correct simplifications before being challenged with a new expression to minimize.

In the subsequent phase, we further refined our prompting strategy:

> "Now, take a deep breath and flip the output on rows: 5, 6, 7. Think step by step on the new updated Boolean expression. Don't provide anything other than the answer and it is IMPERATIVE that the Boolean expression be logically equivalent to the following changes otherwise you are going into a timeout. Also, provide the truth table."

The imperative tone and the threat of a 'timeout' introduce an element of negative reinforcement to the prompt structure, a technique we explored to ascertain its impact on LLM performance. The concept of utilizing positive or negative incentives in prompting is an area of growing interest, with early indications that models may respond differently when presented with the prospect of a reward or penalty. Our empirical observations suggested an improvement in accuracy when incorporating a negative reward within our prompts, a finding that aligned with our objective to enhance the model's focus and precision.

While the broader implications of reward-based prompting are beyond the scope of this paper, our findings contribute to the ongoing discourse on the efficacy of different prompting strategies. The primary focus of our research remains on assessing the ability of LLMs to perform Boolean minimization, an endeavor that serves as a stringent test of the models' capacity for logical reasoning and state tracking.

3.3 Vanilla Llama

Our experimental setup involved the Vanilla Llama model, sourced from the Hugging Face platform. We utilized Google Colab's GPU resources to facilitate efficient model operation and data processing. The experimentation was structured around four distinct datasets, categorized based on the complexity and size of the Boolean functions: small, medium, large, and extreme.

For each dataset, Vanilla Llama was prompted to simplify Boolean expressions. The responses were then analyzed to evaluate the model's performance in terms of accuracy and adherence to Boolean logic. Due to the variability in the model's output formats, the extraction of results was primarily done using regular expressions. However, in cases where the outputs deviated significantly from expected formats, manual inspection was employed to discern and interpret the model's responses. This approach allowed us to systematically collect and assess the data necessary for our study, forming the basis of the results presented in the subsequent section.

3.4 ChatGPT 3.5

In this project, we decided to use ChatGPT 3.5 Turbo to show the few shots learning ability of LLMs on the task of boolean logic simplification. We connected to ChatGPT using OpenAI API key, sending prompts, receiving responses and extracting boolean function results using regex pattern matching. For tests in which ChatGPT was only given the unsimplified expressions and their fully simplified form, we tested the zero shot accuracy of each of small, medium and large functions dataset, and the 1, 3 and 10 shots accuracy for small function dataset. For tests in which ChatGPT was given examples of steps of simplification, we tested the 0, 1, 3, 10 shots accuracy of small function dataset. Each test was run on a dataset with a size of 1000. We used the percentage of equivalency between input and output boolean expressions as the main metric to evaluate the performance of ChatGPT, accompanied with percentage of input and output boolean expressions being exactly the same.

3.5 Fine-Tuning Llama2

We initiated the process by setting up our environment on Google Colab, which provided access to GPUs necessary for handling the computational demands of the Llama2 model. The Colab environment, equipped with a T4 GPU and 16 GB of VRAM, was pivotal in managing the large-scale nature of the Llama2 model, which comprises billions of parameters.

Given the VRAM constraints on consumer GPUs, we employed the QLoRA (Quantized Low-Rank Adaptation) technique for fine-tuning. This method allowed us to optimize VRAM usage and execute fine-tuning in 4-bit precision, crucial for accommodating the model's complexity on limited hardware. The QLoRA approach, which involves quantizing the pre-trained model to 4 bits while updating only the Low-Rank Adapter layers, ensured the model's overall structure remained intact.

The specific steps in our fine-tuning process were as follows:

1. Library Installation: We began by installing necessary libraries from the Hugging Face ecosystem, such as 'accelerate', 'peft', 'bitsandbytes', 'transformers', and 'trl'.

2. Model Configuration: Instead of using the official Meta Llama-2 model, we opted for "NousResearch/Llama-2-7b-chat-hf" from the Hugging Face hub. This selection was driven by easier accessibility and compatibility with our requirements.

3. Dataset Loading: Our custom dataset, designed to challenge the model with a range of Boolean expressions of varying complexity, was loaded from our local storage into the Colab environment.

4. 4-Bit Quantization Configuration: We configured the model for 4-bit quantization using the BitsAndBytes library, ensuring efficient memory usage during fine-tuning.

5. Model and Tokenizer Loading: The Llama2 model and its corresponding tokenizer were then loaded with the 4-bit precision configuration.

6. PEFT Parameters Setting: We configured Parameter-Efficient Fine-Tuning (PEFT) parameters using the LoraConfig from the ‘peft‘ library, focusing on a subset of model parameters for efficiency.

7. Training Parameters: Essential hyperparameters for the training process were set, including batch sizes, learning rates, gradient accumulation steps, and warmup ratios.

8. Model Fine-Tuning: We employed the SFTTrainer from the ‘trl‘ library for Supervised Fine-Tuning (SFT), providing it with the model, dataset, PEFT configuration, tokenizer, and training parameters. The fine-tuning process was carefully monitored to optimize performance and memory usage.

9. Model Evaluation: Post fine-tuning, the model's performance was evaluated using custom prompts to ensure its effectiveness in simplifying Boolean expressions.

The use of our custom dataset aimed to rigorously test and enhance the model's capabilities in this domain.

3.6 Modelling LLMs

The cognitive processes of humans have often been characterized as a dichotomy between rapid, intuitive thought (System 1) and slower, more deliberative reasoning (System 2). In attempting to model Large Language Models (LLMs) for tasks requiring logical reasoning, such as Boolean minimization, we draw parallels to these cognitive systems. LLMs, by their autoregressive nature, are adept at mimicking System 1 thinking—fast and reflexive—suitable for tasks with a heavy reliance on pattern recognition and recall [5].

However, Boolean function minimization often necessitates a more methodical approach analogous to System 2 thinking. Here, each step in the problem-solving process is deliberate and deterministic. To this end, we examine the capability of LLMs to simulate this type of cognition. For instance, by representing a truth table in a structured format, we can probe an LLM's understanding of Boolean expressions and its ability to predict outcomes based on logical operations [6].

Our experimental setup involves transforming Boolean expressions into tokenized sequences that an LLM can process, with the expectation that the model can track changes in the state of a truth table as it is manipulated. To evaluate this, we consider tasks such as flipping the outputs of specific rows in a truth table and assessing the LLM's response in generating the corresponding Boolean expression.

Furthermore, we aim to test the LLM's proficiency in handling a deterministic task: the alteration of a truth table's outputs through specified interventions. The input provided to the model includes the number of inputs, the initial truth table, the Boolean expression, and the rows designated for alteration. The expected output is an updated truth table and a revised Boolean expression. The evaluation criteria are twofold: for the Boolean expression, we assess the accuracy based on the number of incorrect logic gates added or omitted; for the truth table, we count the number of rows with incorrect output values.

In light of recent advancements in LLMs, such as the "Tree of Thoughts" framework [9], which extends beyond token-level decision-making to allow for exploration and strategic planning, we seek to push the boundaries of what these models can achieve. Although our work is initially inspired by the Chain of Thought prompting [10], we also intended to explore the potential of guiding LLMs through the initial steps of a complex task. Unfortunately, our search for a dataset providing the first few steps of Boolean simplification was unsuccessful, highlighting a gap in current resources that future work could aim to fill. However, we created 10 fully explained examples with detailed step by step reasoning so that LLMs could understand our thought process and evaluate the response of Chat-GPT on different numbers of shots.

This objective encapsulates our ambition to instruct LLMs not just to execute tasks, but to understand and carry forward partial solutions, a significant leap toward achieving human-like problem-solving abilities in artificial systems.

4 Experiments and Results

4.1 Vanilla Llama Results

In this section, we detail our experimental approach using the Vanilla Llama model, a variant of the Llama2 large language model. Our objective was to evaluate the model's ability to simplify Boolean expressions accurately.

Initially, we engaged in manual prompt testing, quickly discovering that Vanilla Llama's performance was suboptimal. The model frequently deviated from the correct simplification path, manifesting errors in several distinct ways:

1. **Introduction of Non-Existent Variables:** The model occasionally added new variables to the Boolean expressions that were not present in the original input, leading to incorrect and irrelevant solutions.
2. **Confidence in Incorrect Answers:** Often, Vanilla Llama confidently presented wrong answers, misleading the evaluation process.
3. **Incompatible Output Formats:** The model sometimes returned outputs in formats that were incompatible with the expected Boolean algebraic form, rendering these outputs unusable.
4. **Confusion between Algebra Types:** Vanilla Llama showed a tendency to confuse regular algebra and Boolean algebra, particularly in the representation of variables as numbers.
5. **Explanation Without Execution:** In some instances, the model provided explanations on how to simplify

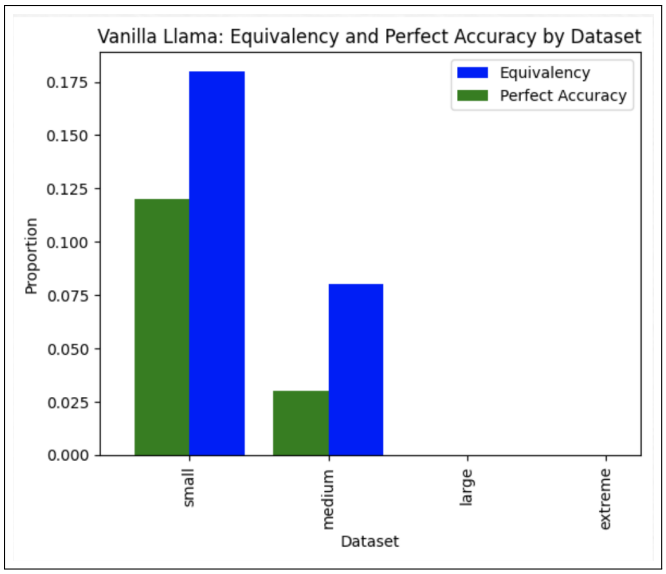

Figure 1. Vanilla Llama2-7B results

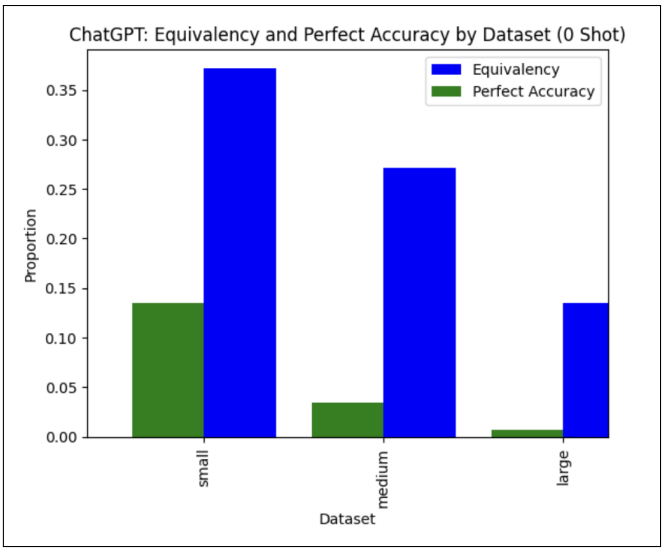

Figure 2. Bar plot indicating ChatGPT's equivalency and perfect accuracy by dataset size without shots.

the expressions rather than performing the actual simplification.

These challenges were evident in our dataset experiments, encompassing small, medium, large, and extreme-sized Boolean functions. The results are shown in the following figure 1.

These disappointing results, particularly the model's struggle with expressions as simple as those with only 2 variables and 3 gates, indicated the necessity for alternative approaches. The lack of accuracy even in smaller instances raised concerns about Vanilla Llama's applicability in Boolean minimization tasks.

Consequently, we considered strategies to enhance the model's performance. This included the exploration of few-shot prompting mechanisms, which involve providing the model with a small number of examples before the actual task to improve its understanding and accuracy. Additionally, the prospect of fine-tuning the model specifically for our requirements was contemplated as a potential solution to the observed limitations.

This exploration with Vanilla Llama provided valuable insights into the limitations of current large language models in handling complex, multi-step Boolean minimization tasks, underlining the need for specialized training and tailored prompting strategies to achieve satisfactory results in this domain.

4.2 ChatGPT Results

Before delving into the fine-tuned Llama results, we examined the performance of ChatGPT as a baseline. The following subsections provide a detailed assessment of ChatGPT's ability to simplify Boolean expressions.

4.2.1 Accuracy by Dataset Size without Shots.

Figure 2 showcases ChatGPT's performance across different dataset sizes without the benefit of examples (0-shot learning). It was observed that ChatGPT's accuracy was highest with small datasets and declined with increasing dataset size. This is consistent with expectations, as smaller problems are less complex and thus easier for the model to solve without prior examples.

4.2.2 Accuracy by Number of Shots.

As illustrated in Figure 3, ChatGPT's performance was evaluated based on the number of 'shots' or examples provided. By giving ChatGPT a few shots as part of the input, we can see an improvement in the performance of ChatGPT. The improvement was notable between 0 shot and 1 shot tests, and as The graph demonstrates a trend where the model's equivalency and perfect accuracy generally increased with the number of shots. This suggests that providing ChatGPT with more examples enhances its ability to generate accurate Boolean simplifications.

These findings from the ChatGPT analysis serve as an important baseline to contextualize the improvements seen with the fine-tuned Llama model, which will be discussed in the subsequent section.

4.3 Fine-tuned Llama Results

The fine-tuning process applied to the Llama model has resulted in a considerable enhancement of its ability to simplify Boolean expressions, as evidenced by the experimental results. The following subsections detail the performance improvements observed.

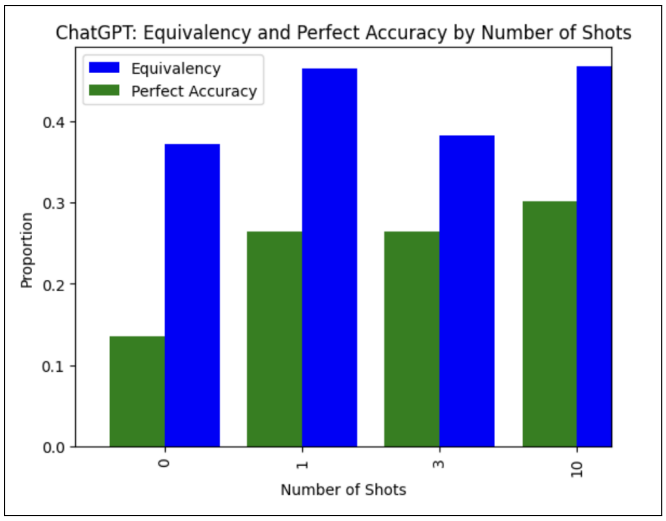

Figure 3. Bar plot showing the proportion of equivalency and perfect accuracy by the number of shots for ChatGPT.

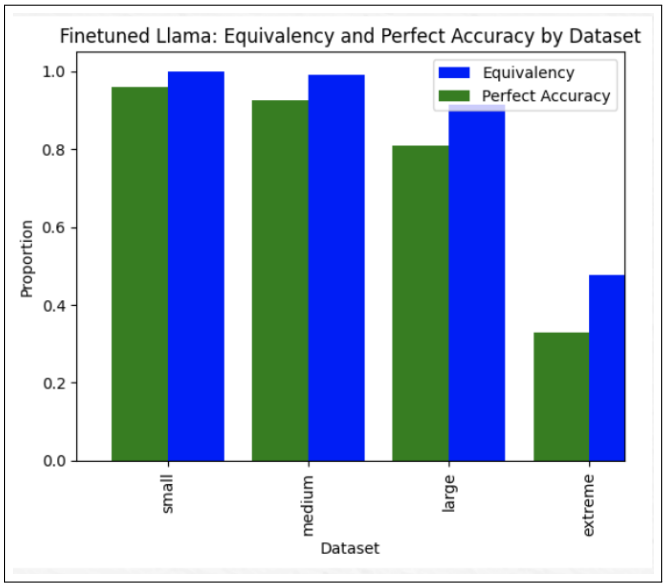

Figure 4. Bar plot showing the accuracy of the fine-tuned Llama model on different sizes of datasets.

4.3.1 Equivalency and Perfect Accuracy by Dataset Size.

The bar plot presented in the figure 4 illustrates the performance across varying sizes of datasets. Notably, the fine-tuned Llama achieved approximately 80% accuracy for large datasets and 50% for extreme datasets, a stark contrast to the 0% accuracy observed with the original Llama model. This indicates that the fine-tuning process significantly increases the model's capability to handle complex Boolean expressions.

4.3.2 Performance with Respect to Input Length.

The figure 5 demonstrates that the fine-tuned Llama sustains high accuracy levels even with input lengths surpassing the maximum encountered during training. This observation implies that the model has successfully generalized the simplification task beyond the training distribution.

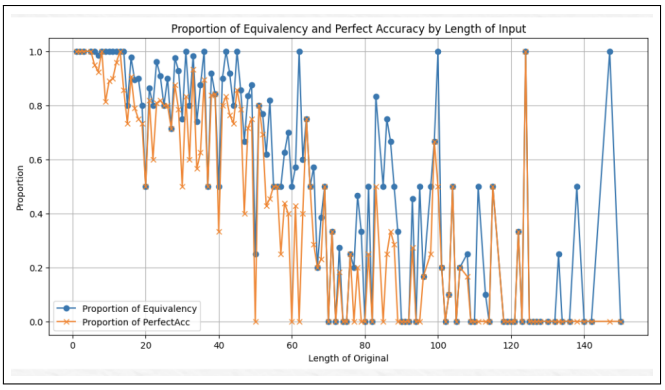

Figure 5. Scatter plot demonstrating the proportion of equivalency and perfect accuracy by length of input for the fine-tuned Llama model.

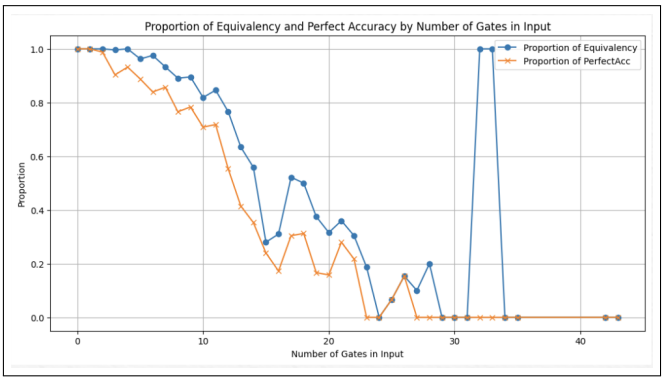

Figure 6. Scatter plot illustrating the proportion of equivalency and perfect accuracy by the number of gates in input for the fine-tuned Llama model.

4.3.3 Accuracy Based on Number of Gates in Input.

As depicted in figure 6, while there is a notable decrease in model accuracy when processing inputs with more than 12 gates, the fine-tuned Llama maintains non-zero accuracy. This is in sharp contrast to the Vanilla Llama's performance, suggesting that the fine-tuned model possesses a robust understanding of Boolean logic that extends beyond the complexity it was trained on.

4.3.4 Implications of the Findings.

The experimental results suggest several key implications for the deployment of large language models in logical reasoning and symbolic manipulation tasks:

- The fine-tuned Llama's generalization ability indicates it has captured the principles of Boolean simplification effectively.

- The model's performance in handling complex inputs suggests that fine-tuning can significantly enhance a model's reasoning capabilities.
- The efficiency of the fine-tuning process points to the possibility of achieving high performance without exhaustive training across all complexities.
- The robustness observed in the model's performance against more complex problems than those seen during training underscores the fine-tuning process's effectiveness in preventing overfitting.

In summary, the fine-tuned Llama exhibits a significant improvement in Boolean expression simplification, paving the way for further research and application of large language models in complex reasoning tasks.

4.4 Modeling LLMs

For this task, we decided to demonstrate the ability of LLMs to model the Truth table and utilize this to logically reason. We decide a deterministic task of flipping the output values of n rows and obtaining the new Boolean expression in CNF form will shine a light on the thinking process of LLMs. The boolean expression was given the number of active variables, a full truth table in matrix form, the original boolean expression, and the rows we would want the output value to be flipped.

Due to the complexity of this task and the large length of instruction text, we provide the LLMs, we use the prompt we described above but also decide to provide one example of doing this task so it fully understands what we are aiming for it to achieve.

Using the prompt from the prompting section in methodology, we got the outputted truth table and boolean expression. We evaluated the boolean expression on the number of incorrect logic gate clauses. This would include the number of logic gate clauses that were both incorrectly added to the final boolean expression and the logic gate clauses that were forgotten by the final boolean expression.

We evaluated the truth table in the same manner by calculating the number of rows that had an incorrect output value.

7

As seen through the figure, 7, the model struggles to obtain the correct boolean expression following the flipped row. We decided to further investigate this issue to understand the effects of the problem. When analyzing the truth table, we see that for the 1 row flip, it gets an average of 0.067 wrong output values on each truth table. This really low value shows that on our full validation set, it is able to properly understand the flips of our output values, but severely struggles when asked to produce the boolean expression. Digging further, we noticed that this can be due to the LLMs trying to obtain the updated boolean expression through

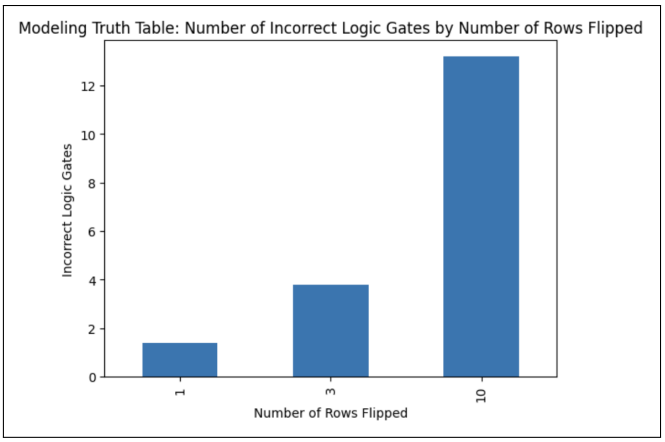

Figure 7. Bar plot illustrating the number of incorrect logic gates by the number of rows flipped.

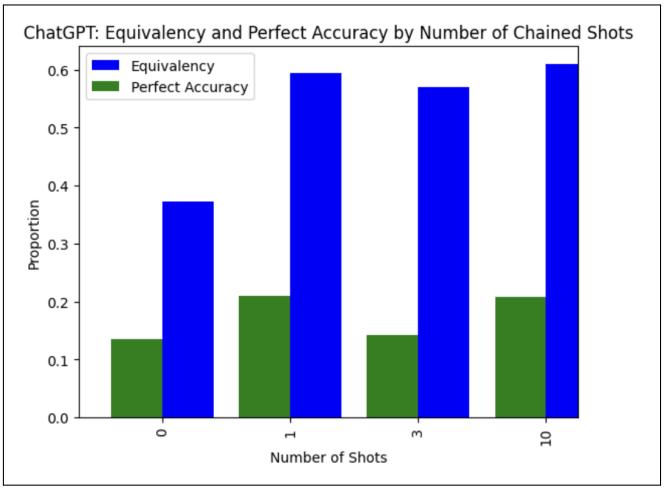

Figure 8. Bar plot illustrating the equivalency and perfect accuracy by the number of chained shots for ChatGPT.

boolean expression rules instead of just obtaining the updated boolean expression. While this could just boil down to bad prompting, it shows that LLMs aren't able to properly utilize their modeling unless it was properly fine-tuned on the dataset like other works. [6]

As we see in figure 8, we see a general trend towards higher equivalency and simplified boolean expressions given more fully worked out examples that show the step by step information using boolean expression rules improves the logical reasoning ability of the LLMs. We wish to pursue further work that would entail a larger and more diverse set of step by step instructions to the LLMs so that they would be able to congregate all the data and understand how to properly reason through our task. We believe RL with AI feedback approaches like a beam search could verify the LLM is taking a valid boolean expression step which will result in the model being more accurate in the responses.

5 Conclusion and Discussion

The experiments conducted on the Vanilla Llama, ChatGPT, and the fine-tuned Llama models have yielded insightful observations regarding the capabilities of large language models in the domain of Boolean function simplification. This study's aim was to not only evaluate the performance of these models on a complex, multi-step logic-based task but also to investigate the impact of fine-tuning on model proficiency.

5.1 Conclusion

The fine-tuned Llama model demonstrated a significant improvement in simplifying Boolean expressions, especially when dealing with large and extreme-sized datasets. The fine-tuning process has proven to be a powerful approach to enhancing the model's understanding of Boolean algebra, as evidenced by its ability to generalize well beyond the scope of the training data and maintain accuracy in the face of increasing input complexity.

In comparison, the baseline models, Vanilla Llama and ChatGPT, exhibited limitations in their ability to accurately simplify Boolean expressions, particularly as the complexity of the tasks increased. The introduction of few-shot learning techniques provided some improvement in ChatGPT's performance, indicating the potential benefits of example-driven learning in large language models.

5.2 Discussion

This research highlights the potential and challenges associated with employing large language models for tasks that require abstract reasoning and symbolic manipulation. The ability of the fine-tuned Llama to generalize and handle complex tasks suggests that large language models can indeed be tailored to specific computational domains through strategic fine-tuning.

However, the study also underscores the necessity of developing more nuanced training and prompting strategies to ensure that these models can provide reliable and accurate outputs when deployed in real-world scenarios. The limitations observed in the Vanilla Llama and ChatGPT models emphasize the need for a deeper understanding of model behaviors and the development of methodologies to mitigate errors such as the introduction of non-existent variables and confusion between different types of algebra.

Future work may explore the scalability of fine-tuning for other specialized tasks and investigate the underlying mechanisms that enable large language models to generalize from their training data. Additionally, further research could assess the cost-effectiveness of fine-tuning versus training new models from scratch for specific applications.

In conclusion, our study provides strong evidence that fine-tuning is an effective method for enhancing the capabilities of large language models in Boolean function simplification. It opens up avenues for more efficient and effective applications of these models in various fields requiring complex problem-solving abilities.

References

[1] Stéphane d'Ascoli, Samy Bengio, Josh Susskind, and Emmanuel Abbé. Boolformer: Symbolic regression of logic functions with transformers. 2023.

[2] Tarun Kumar Jain, Dharmender Singh Kushwaha, and Arun Kumar Misra. Optimization of the quine-mccluskey method for the minimization of the boolean expressions. In *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*, pages 165–168. IEEE, 2008.

[3] Maurice Karnaugh. The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, 72(5):593–599, 1953.

[4] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[5] Gabriel Recchia. Teaching autoregressive language models complex tasks by demonstration. *arXiv preprint arXiv:2109.02102*, 2021.

[6] Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. Chess as a testbed for language model state tracking. *arXiv preprint arXiv:2102.13249*, 2021.

[7] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. 2023.

[8] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[9] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[10] Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, et al. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. *arXiv preprint arXiv:2311.11797*, 2023.

6 Acknowledgments

I'd like to give a big thanks to Ph.D. student Josh Kang and Professor John Wawrzynek for their endless support and guidance throughout this project.

Received 13 December 2012