UDC 004.4`6:004.4`4
DOI https://doi.org/10.32689/maup.it.2024.1.7

Oleksandr DEINEHA
PhD student, Department of Theoretical and Applied Computer Science, School of Mathematics and Computer Sciences, V. N. Karazin Kharkiv National University, oleksandr.deineha@karazin.ua
ORCID: *0000-0001-8024-8812*

LAMBDA CALCULUS TERM REDUCTION: EVALUATING LLMS' PREDICTIVE CAPABILITIES

Abstract. *This study is part of a research series of optimizing compilers and interpreters of functional programming languages. Lambda Calculus was chosen as the most straightforward functional programming language, which can process any operation available to other functional programming languages but with the simplest syntax. Using machine learning methods allows for uncovering relations inside lambda terms, which might indicate which reduction strategy better suits their reduction. Finding those techniques for lambda terms allows optimizing not only lambda term reduction but also interpreters and compilers of functional programming languages.*

This research aims *to scrutinize LLMs' understanding of Lambda term reduction to predict reduction steps and evaluate prediction accuracy. Artificially generated Lambda terms were employed Utilizing OpenAI's GPT-4 and GPT-3.5 models. However, due to model constraints and cost considerations, experiments were limited to terms with specific token counts.*

Despite its larger size, results revealed that the GPT-4 model did not significantly outperform GPT-3.5 in understanding reduction procedures. Moreover, while the GPT-3.5 model exhibited improved accuracy with reduced token counts, its performance with more complex prompts was suboptimal. This underscores the LLMs' limitations in grasping Lambda terms and reduction strategies, especially with larger and more intricate terms.

Conclusions. *The research concludes that general-purpose LLMs like GPT-3.5 and GPT-4 are inadequate for accurately predicting Lambda term reductions and distinguishing between strategies, particularly with larger terms. While fine-tuning may enhance model performance, the current findings highlight the need for further exploration and alternative approaches to achieve a deeper understanding of lambda term reduction using LLMs.*

Key words: *Lambda Calculus, Large Language Model, reduction process, prompt engineering.*

Олександр ДЕЙНЕГА. РЕДУКЦІЯ ТЕРМІВ ЛЯМБДА-ЧИСЛЕННЯ: ОЦІНКА ПРОГНОЗИВНИХ ЗДАТНОСТЕЙ LLM

Анотація. *Це дослідження є частиною серії досліджень оптимізації компіляторів та інтерпретаторів функціональних мов програмування. Лямбда-числення було обрано як найпростішу мову функціонального програмування, яка може обробляти будь-які операції, доступні іншим мовам функціонального програмування, але з найпростішим синтаксисом. Використання методів машинного навчання дозволяє виявити зв'язки всередині лямбда-термів, які можуть вказати, яка стратегія редукції краще підходить для їх нормалізації. Пошук цих методів для лямбда-термів дозволяє оптимізувати не тільки редукцію лямбда-термів, але й інтерпретатори та компілятори функціональних мов програмування.*

Мета. *Це дослідження має на меті вивчити як LLM розуміє лямбда-терми, для цього передбачити кроки редукції та оцінити точність передбачень. Використовувалися штучно створені лямбда-терми з використанням моделей OpenAI GPT-3.5 і GPT-4. Однак через обмеження моделей та міркування щодо вартості експериментів були обмежені термами з певною кількістю токенів.*

Незважаючи на більший розмір, результати показали, що модель GPT-4 незначно перевершила GPT-3.5 у розумінні процесу редукції. Крім того, у той час як модель GPT-3.5 продемонструвала підвищену точність із зменшеною кількістю токенів, її продуктивність із більш складними термами була неоптимальною. Це підкреслює обмеження LLM у розумінні лямбда-термів і стратегій скорочення, особливо з більшими та складнішими термами.

Висновки. *Дослідження показує, що LLM загального призначення, такі як GPT-3.5 і GPT-4, недостатні для точного прогнозування скорочень лямбда-термів і розрізнення стратегій, особливо з більшими термами. Хоча точне налаштування може підвищити продуктивність моделі, поточні результати підкреслюють необхідність подальшого дослідження та альтернативних підходів для досягнення глибшого розуміння редукції лямбда-терму за допомогою LLM.*

Ключові слова: *Лямбда-числення, Велика Мовна Модель, процес редукції, інженерія промпту.*

Introduction. Our research aimed at optimizing functional programming compilers and interpreters. For this purpose, we considered Lambda Calculus the most straightforward possible representation of functional programming languages [1]. Lambda Calculus allows the execution of its programs called terms as the expression reduction process. The lambda terms can be divided into Applications, Abstractions, and Variables. The term reduction is possible using redexes, special combinations of Abstract, and any other term inside an Application. Some terms may contain more than one redex, and choosing a specific redex by some rule defines a reduction strategy. The most famous reduction strategies are the normal order or the leftmost outermost (LO) strategy and the applicative order or the rightmost innermost (RI) strategy. An example of the Y term is shown in Figure 1. The example describes all lambda term elements and the RI and LO strategies.

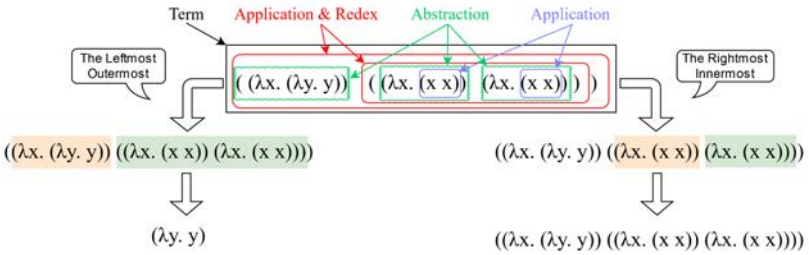

Fig. 1. Example of the Y term, applying the rightmost innermost and the leftmost outermost strategies and showing term types

Research in the Lambda Calculus reduction process may help optimize the interpretation and compilation of other function programming languages using similar optimization techniques [2]. Understanding the Lambda terms reduction process may help uncover essential features and methods of discovering strategy priorities.

Analysis of recent research and publications. Usually, reduction steps are estimated as equal [3], which allows comparison of reduction strategies via a simple number of reductions. In the research [4], we considered another approach to estimating reduction steps via their computational efficiency, which allows us to develop a greedy strategy that minimizes computational resources required for reduction. Although this approach enables estimating computational resources needed for single-step normalization, it does not allow us to understand the relation between specific terms and strategies that better suit its normalization.

The problem of measuring term complexity was considered in the studies [5, 6], where memory consumption. Also, cost models were proposed in the article [7] to solve the same problem. All works show that it is possible to define the complexity of reduction steps via different approaches.

The research [8] considers using the Transformer models for sequential analysis of lambda terms for predicting the type of term in Typed Lambda Calculus, which simply extends Lambda Calculus with defining types via specific terms without modifying the expressions lexicon. This usage highlights the idea that extracting particular term features that indicate its type is possible. That idea can be extended to the strategy priority, and in the research, [9] was considered to estimate the reduction steps number for the RI and the LO strategies. However, research [9], due to computational limitations, considers only simplified term representation, which does not count variable information. The study [9] results show that this approach allows accurate identification of reductions if the expected number is less than 10, but for bigger expected numbers, the accuracy drops.

Studies [2, 10] solved the problem of losing variable information with more complex pretrained machine learning models, namely Microsoft CodeBERT [11] and OpenAI [12] embedding models. Those studies used vector representations of lambda terms created with Large Language models (LLM) and uninformed Machine Learning techniques to find the relation between collected vectors and the most suitable reduction strategy. The main issue with the studies [2, 10] is using pretrained models on programming languages or for general problems. It does not provide information about LLMs' understanding of lambda terms.

Also, recent research has shown promising results with the implementation of LLMs for solving mathematical problems [13], code execution [14], and compilation optimization [15]. All such works show that it is possible to use LLMs as a discovery tool for code-related tasks, but no one has checked how good general tasks LLMs are for understanding lambda terms.

The research objective is to investigate LLM's understanding of the lambda term reduction process. Achieving this objective can be highlighted in the following tasks:

1. Prepare a lambda terms dataset containing the following reduction step term for the selected strategies.
2. Predict the following reduction step using the selected strategy, general LLM, and prepared terms.
3. Calculate the accuracy of such predictions and conclude that LLMs can be used to understand lambda term reduction and strategy differences.

Scientific novelty. For the first time, the ability of LLM to understand lambda calculus was investigated.

Research methodology. The research is considered one of the biggest publicly available general task LLM models, developed and trained by OpenAI: the GPT-4 and GPT-3.5 models [16]. Table 1 shows the considered models, their weights number, and price per million tokens [16].

Table 1

Comparison of the GPT-3.5 and GPT-4 models

Model name	Weights Number	Price of input per million tokens	Price of output per million tokens
GPT-3.5	~20 Billion	0.50$	1.50$
GPT-4	~220 Billion	30.00$	60.00$

In this research, artificially generated lambda terms were used. The procedure for generating those terms was described in the study [9]. Accordingly to the procedure, with some probability, choose the next term element (Application, Abstraction, or Variable from a set of available variables), which recursively builds a term. This procedure allows consider the maximum available terms in the selected bound of variables and probabilities of elements. This research uses the same terms dataset used in previous studies [2, 10]. However, considering price limitations, the number of terms used for testing decreased, considering the number of input and expected output tokens. The results of such data preparing shown in Table 2. Although the number of terms used in experiments has significantly shrunk, there are still enough terms to check the proposed LLM's ability to understand the reduction process with differing strategies. Also, shown LLMs require special text descriptions, called prompts, for problem statements, which a LLM must solve. The prompt size also increases the number of required input tokens for each term, and depending on the prompt, it can increase the number of expected output tokens

Table 2

Results of term dataset preparing

	Original dataset	**Cropped to 77 max tokens per term**	**Cropped to 40 max tokens per term**
Terms number	4282	1019	305
Input tokens	523k	52k	8k
Expected output LO tokens	503k	45k	6.4k
Expected output RI tokens	521k	45k	6.4k

Considering the analysis of available GPT models' price and weight numbers and the concluded size of datasets, it is possible to define the methodology of experiments:

1. Prepare prompts for predicting the LO / RI steps using the GPT-3.5 / GPT-4 models.
2. Using prepared prompts, predict the following reduction step using the selected model (use for prediction cropped dataset to 77 tokens per term with GPT-3.5 and cropped to 40 tokens – GPT-4 model).
3. Postprocces predictions to formulate actual term answers.
4. Using the Lambda Calculus interpreter, the expected following terms are compared with actual predictions.

Results of research. The first stage of the study requires preparing a prompt. There are a few prompt types [17]: some require detailed descriptions of solved tasks, some require examples of solving, and others require simply asking about the task. Due to the high price of using the GPT-4 model, the simplest approach was chosen. On another site, the GPT-3.5 model was considered a few approaches.

```
f"""Given the lambda term, apply the leftmost-outermost (LO) strategy to perform the next step of reduction.
The LO strategy, also known as normal order reduction, prioritizes the reduction of the leftmost-outermost redex first.
This means that if there's a choice between reducing an expression inside a lambda abstraction or an application outside,
the application takes precedence unless there's no other redex outside the abstraction.

Lambda Calculus Reduction Rules:

1. Alpha Conversion (α-conversion): Rename bound variables, ensuring no variable name conflicts.
This step is essential for avoiding collisions between variables.

2. Beta Reduction (β-reduction): Apply the function to its argument.
The formal rule is ((λx.M) N) → M[x:=N], where M[x:=N] denotes substituting N for x in M.

3. Eta Conversion (η-conversion): Simplify functions with unnecessary abstractions. The rule is λx.(M x) → M if x does not appear in M.

Prioritization in LO Strategy:

- Outermost First: Reduce the outermost redex before any inner redexes, even if the inner one is to the left of an outer one.
- Leftmost First: When faced with multiple outermost redexes, choose the leftmost one.

Examples:

- Given (λx.x x) ((λy.y) z), the LO strategy first reduces the outermost leftmost redex, resulting in (λx.x x) z.
- For ((λx.λy.x y) (λa.a)) b, the first step of reduction under LO strategy would yield (λy.(λa.a) y) b.

Procedure:

- Identify the leftmost-outermost redex in the term.
- Apply the appropriate reduction rule based on the structure of this redex.
- If multiple steps are available, choose the one that aligns with the LO strategy's prioritization.

Take your time to analyze the term <<<{str_term}>>. Consider each part of the term carefully and apply the reduction rules as described.
Remember to use α-conversion to avoid variable naming conflicts, especially when dealing with nested lambda expressions.
In the end provide the next reduction term in format: Result: next step term
"""
```

Fig. 2. Using the description prompt, the GPT-3.5 model generates the following term according to the LO strategy

```
f"""Example of performing task #1:
Given term: (λx.((λy.((λz.z) x)) (λa.a))). Provide the next step of term reduction

Your output:
1. Identify the leftmost-outermost redex in the given term: ((λy.((λz.z) x)) (λa.a))
1.1. Where object of the redex is (λy.((λz.z) x)) (λa.a)
1.2. And subject of the redex is
2. Apply β-reduction: ((λy.((λz.z) x)) (λa.a)) [x:= (λa.a)]
3. Result: (λy.((λz.z) (λa.a)))

Example of performing task #2:
Given term: (((λx.x) (λy.(y (λz.z)))) (λa.a)). Provide the next step of term reduction.

Your output:
1. Identify the leftmost-outermost redex in the given term: (((λx.x) (λy.(y (λz.z)))) (λa.a))
2. Apply β-reduction: ((λy.(y (λz.z))) (λa.a)) [x:= (λy.(y (λz.z)))]
3. Result: ((λy.(y (λz.z))) (λa.a))

Given term: {str_term} Provide the next step of term reduction using example.

Your output:
"""
```

Fig. 3. Using the detailed step prompt, the GPT-3.5 model generates the following term according to the LO strategy

```
f"""
Please generate the next step of reduction a Lambda Calculus term. Provide only term expression.

Lambda term: '''{str_term}'''
"""
```

Fig. 4. Using the simplest command prompt, the GPT-4 model generates the following term according to the LO strategy

Figure 2 shows an example of the description prompt used for the GPT-3.5 model, Figure 3 shows an example of the detailed step prompt used for the GPT-3.5 model, and Figure 4 shows an example of the simplest command prompt used for the GPT-4 model. All shown prompts are used for the LO strategy, but the prompt for the RI strategy differs only in the description of the RI strategy, but the logic is kept the same.

Table 3

Accuracy of the following step predictions with the GPT-3.5 and GPT-4 models

	GPT-3.5 to LO (77 tokens)	GPT-3.5 to LO (40 tokens)	GPT-3.5 to RI (77 tokens)	GPT-3.5 to RI (40 tokens)	GPT-4 to LO (40 tokens)	GPT-4 to RI (40 tokens)
Description prompt	9.5%	23.6%	6.47%	17.04%	–	–
Detailed step prompt	4.0%	10.82%	3.0%	9.83%	–	–
Simplest command prompt	10.0%	27.21%	3.23%	9.83%	41.3%	36.39%

Table 3 shows all the experiments. Due to the high price of GPT-4 model generation, only the simplest command prompt and terms with a maximum of 40 tokens were considered, which showed the best results with the GPT-3.5 model. Experiments with the GPT-3.5 model were considered terms with a maximum of 77 tokens; 40 token results were extracted from the collected results.

Discussion. Low accuracy on more complex prompts (description and detailed step) might indicate overloading the model with redundant details. Also, increasing accuracy with a decreasing maximum number of tokens shows that the GPT-3.5 model cannot profoundly analyze and understand lambda terms. Real programs can contain hundreds of variables, which is a big problem to analyze. A drop in accuracy of 5-7% on prediction following terms for the RI strategy compared to the LO can be explained by the fact that GPT-3.5 and GPT-4 models do not wholly understand the redex and reduction strategy concept. However, the most significant accuracy for predicting the following RI term was achieved using the description prompt, indicating that LLMs can improve their redex understanding, but it depends on prompt construction.

The 10% difference between the best results achieved on GPT-3.5 and GPT-4 indicates that increasing the number of model weights doesn't significantly improve understanding of the reduction procedure. However, the GPT-4 model was the closest to accurate results.

This research benefits from showing that general task LLMs are unsuitable for predicting the following term step. Fine-tuning techniques can improve such models but with more affordable ones.

The research disadvantages are not uncovering all possible experiments on the GPT-4 model with different prompts due to the high generation price, using a limited number of tokens in experiment terms, and considering only OpenAI models. Considering these disadvantages, the following research step could fine-tune some LLM for more accurate results.

Conclusions. The results of the research were solved in the following tasks:

1. A lambda terms dataset has been prepared considering the limitations of selected LLM models. The prepared dataset allowed to check how selected models understand lambda calculus reduction and the difference in strategies by selecting two reduction strategies (LO and RI).

2. The following reduction steps were predicted using GPT-3.5 and GPT-4 models. The predictions were cleaned to check their reliability.

3. Using the Lambda Calculus interpreter, the predictions' results were compared to expected terms, which allowed the predictions to be calculated accurately. The applicability of GPT-3.5 and GPT-4 was examined, and it was concluded that selected LLMs are insufficient to understand lambda term reduction and strategy differences, especially on larger terms.

Bibliography:

1. Cummins, C., Seeker, V., Grubisic, D., Elhoushi, M., Liang, Y., Roziere, B., Gehring, J., Gloeckle, F., Hazelwood, K., Synnaeve, G., Leather, H. Large Language Models for Compiler Optimization. *ArXiv*, 2023. URL: https://arxiv.org/abs/2309.07062.

2. Dal Lago, U., and Martini, S. On Constructor Rewrite Systems and the Lambda Calculus. *Logical Methods in Computer Science*, 2012, Volume 8. DOI: 10.2168/LMCS-8(3:12)2012.

3. Deineha, O. Supervised data extraction from Transformer representation of lambda-terms. *Radioelectronic And Computer Systems*, 2024. In press.

4. Deineha, O., Donets, V., & Zholtkevych, G. Deep Learning Models for Estimating Number of Lambda-Term Reduction Steps. *ProfIT AI 2023: 3rd International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2023)*, 2023, vol. 3624, pp. 147-156. URL: https://ceur-ws.org/Vol-3641/paper12.pdf.

5. Deineha, O., Donets, V., & Zholtkevych, G. Estimating Lambda-Term Reduction Complexity with Regression Methods. *International Conference "Information Technology and Interactions"*, 2023, no. 3624, pp. 147–156. URL: https://ceur-ws.org/Vol-3624/Paper_13.pdf.

6. Deineha, O., Donets, V., & Zholtkevych, G. Unsupervised Data Extraction from Transformer Representation of Lambda-Terms. *Eastern European Journal of Enterprise Technology*, 2024. In press.

7. Deliyannis, E.P., Paul, N., Patel, P.U., & Papanikolaou, M. A comparative performance analysis of Chat GPT3.5, Chat GTP4.0 and Bard in answering common patient questions on melanoma. *Clinical and experimental dermatology*, 2023. DOI: 10.1093/ced%2Fllad409.

8. Dezani-Ciancaglini, M., Ronchi Della Rocca, S., and Saitta, L. Complexity of lambda-term reductions. *RAIRO Theor. Informatics*, 1979, Appl. 13: 257-287. DOI: 10.1051/ita/1979130302571.

9. Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp 1536–1547. DOI: 10.18653/v1/2020.findings-emnlp.139.

10. Grabmayer, C. Linear Depth Increase of Lambda Terms along Leftmost-Outermost Beta-Reduction. *ArXiv,* 2019. URL: https://doi.org/10.48550/arXiv.1604.07030.

11. Liu1, C., Lu, S., Chen, W., Jiang, D., Svyatkovskiy, A., Fu, S., Sundaresan, N., Duan, N. Code Execution with Pre-trained Language Models. *Accepted to the Findings of ACL 2023*, 2023. URL: https://arxiv.org/abs/2305.05383.

12. Miranda, Brando, Shinnar, Avi, Pestun, Vasily, Trager, Barry. Transformer Models for Type Inference in the Simply Typed Lambda Calculus: A Case Study in Deep Learning for Code. *Computer Science*, 2023. URL: https://arxiv.org/abs/2304.10500.

13. Pollak, D., Layka, V., & Sacco, A. Functional Programming. Beginning Scala 3. 2020. DOI:10.1007/978-1-4842-7422-4_4.

14. Qi, Xiaochu. Reduction Strategies in Lambda Term Normalization and their Effects on Heap Usage. *ArXiv*, 2004. URL: https://arxiv.org/abs/cs/0405075.

15. White, J., Hays, S., Fu, Q., Spencer-Smith, J., & Schmidt, D.C. ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design. *ArXiv*, 2023. DOI: 10.48550/arXiv.2303.07839.

16. Yang, Zhen, Ding, Ming, Lv, Qingsong, Jiang, Zhihuan, He, Zehai, Guo, Yuyi, Bai, Jinfeng, Tang, Jie. GPT Can Solve Mathematical Problems Without a Calculator. Machine Learning. *ArXiv*, 2023. URL: https://arxiv.org/abs/2309.03241.

17. New embedding models and API updates. *Blog OpenAI*, 2024. URL: https://openai.com/blog/new-embedding-models-and-api-updates (accessed 24.04.2024).