Specifying exact circuit algorithms in universal transformers

Takuya Ito*
T.J. Watson Research Center
IBM Research

Ruchir Puri
T.J. Watson Research Center
IBM Research

Parikshit Ram
T.J. Watson Research Center
IBM Research

Abstract

Understanding how transformers can execute specific algorithmic and symbolic computations remains a challenge in artificial intelligence. Prior work has demonstrated that standard transformers have trouble generalizing algorithmic problems of arbitrary length (i.e., length generalization), such as arithmetic problems. Here we present an interpretable and modular framework for specifying exact algorithmic computations with universal transformers that enable these models to perfectly solve algorithmic problems of arbitrary depth (length), *without any training*. In particular, by formulating algorithmic problems as computable circuits, we exactly map circuit computations onto components of the universal transformer architecture. We showcase this ability by specifying universal transformers that perfectly solve two fundamental algorithmic problems: modular arithmetic and Boolean logic. Notably, these two models demonstrate how transformers can generalize to problems of *any length* using interpretable architectural modifications. This framework can be naturally adopted for any algorithmic problem that can be formulated as a circuit, illustrating exactly how transformers can implement arbitrary circuit algorithms. More broadly, this framework provides an existence proof of transformer models capable of implementing exact algorithms, providing avenues of opportunity for exploring their learnability in future work.

1 Introduction

Recent advances in AI have led to remarkable improvements in models capable of complex reasoning tasks [Shao et al., 2024, DeepSeek-AI et al., 2025, Yang et al., 2025, Mishra et al., 2024, Bercovich et al., 2025]. These models, often powered by large transformer models trained on vast corpora, exhibit behaviors that resemble human-like inference and deduction. However, the underlying mechanisms by which these models reason, or the algorithms by which they reason, remain largely opaque. This is because the algorithms these models implicitly implement are neither explicitly encoded, understood, nor easily interpretable.

To address this, recent work has explored the use of circuits – structured, interpretable, computational graphs – to model and analyze the algorithmic reasoning capabilities of AI systems [Dziri et al., 2023, Ito et al., 2025, Ram et al., 2024]. Importantly, a circuit representation of a problem *exactly* encodes the algorithm required to solve that problem (follow the edges from the input gates). Moreover, circuit-based approaches to devising algorithmic problems are closely related to other problems

*taku.ito1@gmail.com

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MATH-AI.

widely studied throughout the generalization literature, including compositional generalization and length generalization problems [Hupkes et al., 2020, Jelassi et al., 2023, Zhou et al., 2024a, Lee et al., 2023]. These problems require models to generalize to novel, variable-length sequences (often of greater lengths than seen during training) [Deletang et al., 2022]. However, a number of empirical studies have demonstrated that transformers typically struggle with length generalization problems (see Sinha et al. [2024] for a review).

In this study, we introduce a universal transformer architecture capable of exactly computing algorithmic circuit problems of arbitrary depth. We focus on two simple yet widely important algorithmic problems: Boolean logic and modular arithmetic. Critically, provided a problem's circuit encoding, we implement universal transformer models that *exactly* solve these problems of arbitrary length *without any training*. We achieve this by ensuring the transformer's mechanisms – namely the attention mechanism and multilayer perceptron (MLP) – implement well-defined roles when solving the problem: syntactic parsing (attention) and semantic evaluation (MLP). We provide formal descriptions of this transformer formulation, as well as empirical experiments that exhibit perfect performance on Boolean logic and arithmetic problems of arbitrary circuit depths. Overall, our results provide an existence proof (in the form of a tangible, performant model) of a minimal universal transformer architecture capable of solving arbitrary algorithmic circuit problems.

2 Circuit Problems

Overview. We focus on computing Boolean circuit and arithmetic circuit problems, varying the size and depth of these formulas. These problems are highly related to compositional and length generalization problems (compositional problems can be reformulated as a circuit) [Hupkes et al., 2020, Jelassi et al., 2023]. However, circuit problems have the added benefit of having algorithmically meaningful descriptions; the circuit describes the algorithm to solve that problem (follow the edges from the input gates), and the circuit depth and size correspond to algorithmic time and space complexity, respectively. Figure 1A depicts an example of a depth-2 arithmetic circuit, and Figure 1B depicts an example of a depth-2 Boolean circuit.

Circuit sampling. For experiments, we constructed a dataset of circuits by constructing a Boltzmann sampling procedure over a simple logical grammar. For Boolean formulas, the grammar consisted of two terminals/leaves (`TRUE`, `FALSE`), two binary operators (`AND`, `OR`), and one unary operator (`NOT`). For arithmetic formulas, the grammar consisted of 10 terminals/leaves (integers `0` through `9`, and two binary operators ($+$ and $\times$). Boltzmann sampling provides a principled way to randomly generate these combinatorial objects (circuits). Specifically, by using the generating function

$$Z(x) = Lx + UxZ(x) + BxZ(x)^2 \tag{1}$$

where L, U, and B are the number of leaf, unary, and binary productions, respectively, we can encode the counts of formulas of each size. For Boolean circuits specified above, $L = 2$, $U = 1$, and $B = 2$. For arithmetic circuits specified above, $L = 10$, $U = 0$, and $B = 2$. This procedure and dataset was implemented in PyTorch (version 2.6.0).

3 Universal Transformer Architecture for Computing Circuits

At a high-level, we consider a 1-layer universal transformer. The goal of this universal transformer is to simulate a depth-1 circuit, capturing the minimal circuit computation. For Boolean formulas, this is equivalent to a finite truth table. This implies that the number of forward passes required for a universal transformer to simulate a depth-k circuit is exactly k. To architect this transformer, we assign specific functions – syntactic parsing and semantic evaluation – to two of the transformer's core components – attention and the MLP, respectively. Using the attention mechanism to impose a syntactic parse (via the circuit's adjacency matrix), we control the read/write access to residual token streams in the transformer to mimic the structure of the circuit. This can be conceptualized as deriving an attention mask determined by relative positional encodings, where the distance between tokens (circuit gates) is determined by the existence of an edge between circuit gates. This ensures that the MLPs only need to implement a finite look-up table, such that it evaluates a depth-1 circuit (e.g., $1 \vee 1 = 1$ or $5 \times 4 = 0$) We illustrate the high-level intuition for a single forward pass of this universal transformer in Fig. 1D and step-by-step computations in Algorithms 1 and 2 (Appendix). In the Appendix, we also provide additional details for how to construct the token embeddings (A.1), attention mechanism (A.2), and MLP (A.3).

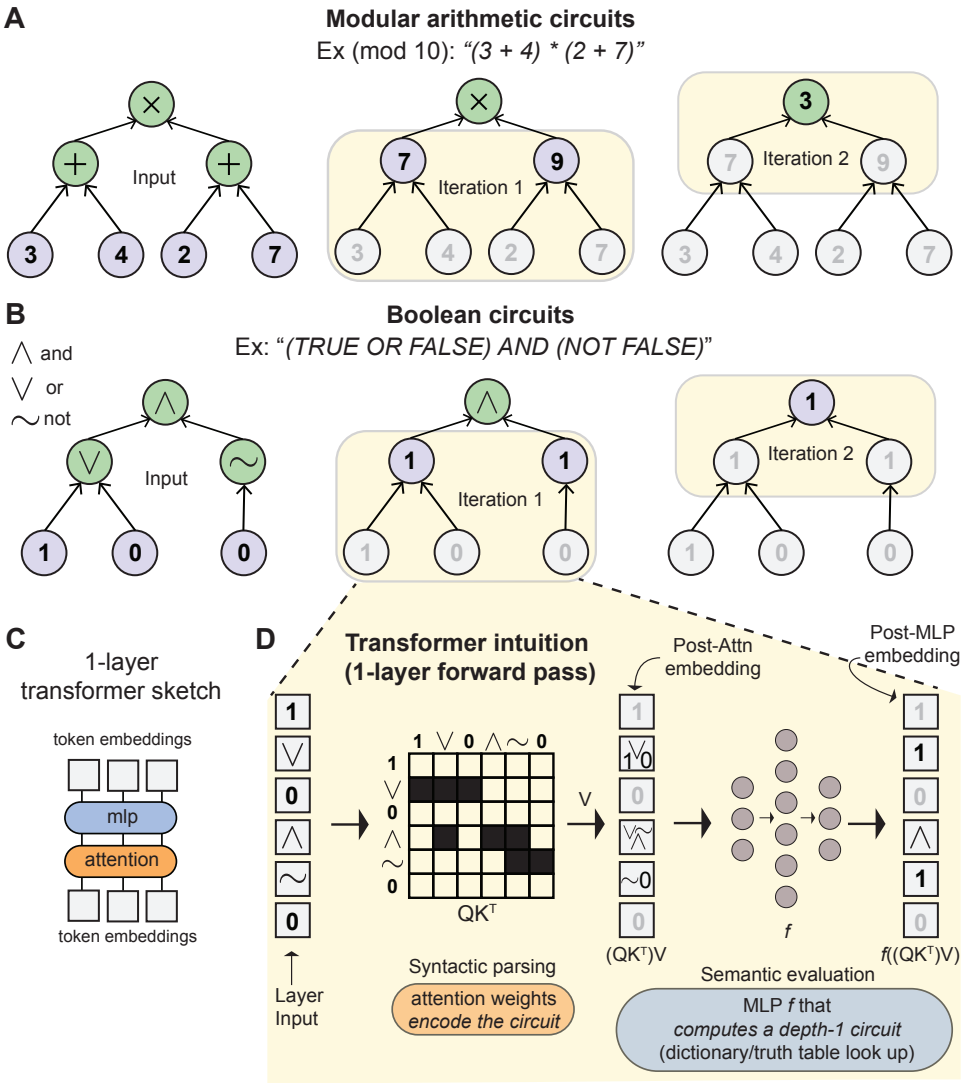

Figure 1: **Circuit problems and transformer architecture intuition. A)** An example of an arithmetic circuit (mod 10) of depth 2, and our approach to solving these circuits in transformers. Importantly, at each universal transformer iteration, we update the operator gate into the evaluated operator. **B)** An example of a depth-2 Boolean circuit with gates 1, 0, $\wedge$, $\vee$, $\sim$. **C)** The key architectural components of a 1-layer universal transformer. **D)** An intuition of how to compute *one iteration* of a circuit algorithm with the transformer. Given a sequence of tokens as the input string, the attention mechanism implements a syntactic parse of that string by using the circuit's adjacency matrix as the attention mechanism (i.e., the $QK^\top$). By setting the values matrix V as the identity matrix, the attention weights map the embeddings of input gates (which are one-hot encodings) to the target (operator) gate, yielding a superposition of token embeddings (multi-hot encoding). The token-wise MLP implements a depth-1 circuit evaluation (in this case, a truth table lookup) that maps a vector of token counts (equivalent to a superposition of token embeddings) to the correct token. When a counts vector cannot be evaluated (e.g., an invalid expression), a conditional residual connection is used to carry the layer's input embedding to the post-MLP embedding. When all operator gates are computed, the circuit computation is completed.

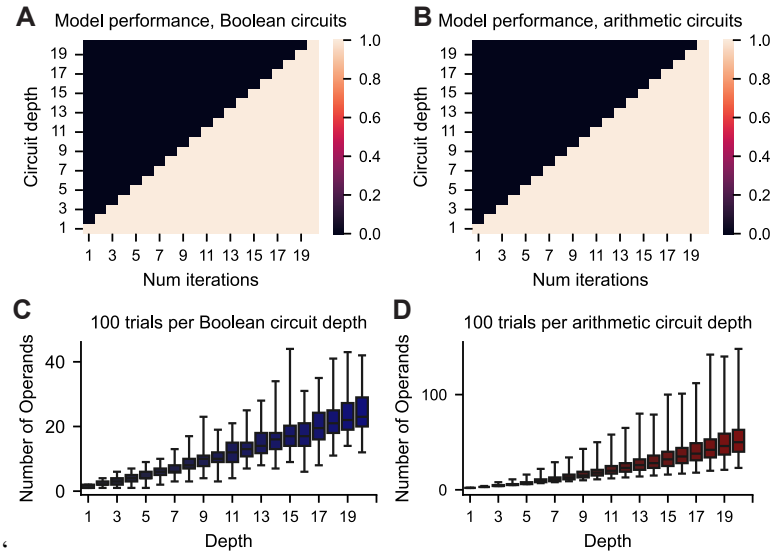

Figure 2: Empirical evaluation of our universal transformer on circuit problems of various depths. For each circuit depth, we sampled 100 random problems of **A)** Boolean and **B)** arithmetic expressions. Performance was perfect across all circuit depths when the universal transformer iterated at least k steps on problems of depth-k. **C, D)** We illustrate the number of operands across the 100 problems per circuit depth. Circuits of the same depth can have a variable number of operands. Most notably, however, as circuit depth increases, the number of operands exponentially increases, despite requiring only a linear increase in iterations. Boxplots represent the quartiles of the distribution, and the whiskers the full extent of distribution ($n = 100$).

4 Experiments

We tested our universal transformer on both Boolean and modular arithmetic problems of arbitrary length, randomly sampled from a Boltzmann distribution. For each circuit depth, we sampled 100 randomly generated circuits (formulas). Since model parameters were chosen (not learned), no data were used for training. Note that while we sampled circuits of a particular depth, each randomly sampled circuit may have a different number of operands, since circuits were not required to be balanced. To correctly evaluate the expression (i.e., flip the output gate to the correct response/operand), the universal transformer needed to iterate for at least k steps for a depth-k circuit. To verify our architecture, we sampled Boolean circuits from depth 1 to depth 20, and for each problem, iterated the universal transformer for 1 to 20 iterations. As expected, we found that when our universal transformer iterated for at least k iterations on a circuit problem of depth-k, our model achieved 100% performance (Fig. 2A,B). When our model iterated for $< k$ iterations, model performance dropped to 0%, despite chance performance being 10% for modular arithmetic, and 50% for Boolean logic. This is because the output gate, which is an operator, did not update to the correct operand.

We further provide statistics on the range of problems encompassed by the 100 randomly sampled problems when sampling from a depth-k circuit, counting the number of operands (Fig. 2C,D). This provides a meaningful comparison to related length generalization studies, which typically only evaluate generalization according to the number of operands, rather than depth. Interestingly, while the number of operands exponentially increases as a function of circuit depth, we illustrate with our universal transformer that the number of iterations to compute exponentially large problems (in operands) only scales linearly.

5 Discussion

Relation to length generalization, compositional generalization, and reasoning. Length generalization and compositional generalization are central challenges in algorithmic reasoning in transformers. Prior work has shown that standard transformers often fail to generalize to longer or

more complex inputs than seen during training [Dziri et al., 2023, Jelassi et al., 2023, Zhou et al., 2024a, Ito et al., 2024, Shen et al., 2024, Zhou et al., 2024b]. Our framework directly addresses this limitation by leveraging 1) a circuit encoding of algorithmic problems, and 2) the iterative nature of universal transformers, providing an existence proof of an exact, universal transformer implementation capable of algorithmic reasoning on arbitrary length problems *without any training*. By forcing the attention weights to encode the circuit's edges as an adjacency matrix, we use the transformer's attention mechanism as a syntactic parser. This ensures that the MLP layer can act as a finite semantic evaluator, parallelizing the evaluation of depth-1 subcircuits. This design enables perfect generalization to arbitrarily deep circuit problems without training, providing a concrete architectural solution to compositional and length generalization problems.

Relation to transformer expressivity. Theoretical studies have established that transformers are Turing complete under certain conditions [Merrill and Sabharwal, 2024, Strobl et al., 2024, Chen et al., 2025, Yang et al., 2024]. However, practical demonstrations of this expressivity remain limited. Our work provides an existence proof of a performant universal transformer architecture for exact algorithmic circuit evaluation. As circuits are themselves used as models of algorithms – with associated algorithmic complexity measures, like time and space complexity [Ito et al., 2025] – this provides a framework for understanding algorithmic computation in neural models.

Relation to mechanistic interpretability. Recent efforts in mechanistic interpretability aim to understand how transformers implement specific computations [Elhage et al., 2021, Olsson et al., 2022, Sharkey et al., 2025]. Our framework contributes to this line of work by decomposing transformer operations into interpretable modules: attention for syntactic parsing and MLPs for semantic evaluation. Moreover, our use of hard-coded attention weights derived from circuit adjacency matrices provides a concrete instantiation of functional routing within transformer layers, and is common in the mechanistic interpretability literature [Elhage et al., 2021]. Our approach also shares conceptual similarities with RASP (Restricted Access Sequence Processing) [Weiss et al., 2021], which provides a programming language for expressing transformer computations using restricted attention and primitive operations. Like RASP, our model uses hard-coded attention patterns to simulate algorithmic behavior. However, RASP is limited in that only finite-depth transformers can be programmed, thereby limiting its expressivity. On the other hand, ALTA (A Language for Transformer Analysis) [Shaw et al., 2024] provides a formal language for specifying universal transformers that can similarly solve arbitrary depth problems. Our work complements ALTA by offering a concrete instantiation of algorithmic circuits within a universal transformer, without the need to specify a specific programming language, and instead demonstrating how such programs can be executed exactly through interpretable architectural design.

Limitations. While we provide a universal transformer implementation that computes exact circuit algorithms, there are several limitations to our study. First, our models rely on manually specified attention weights derived from circuit adjacency matrices. While this enables perfect performance without training, it bypasses the challenge of learning such attention patterns from data. Future work should explore whether these mechanisms can be learned end-to-end, and under what conditions. Second, our approach assumes noiseless input and idealized token embeddings (i.e., one-hot vectors). This simplifies the semantic evaluation to finite lookup tables, but may not generalize to real-world settings where inputs are noisy or ambiguous. Third, our experiments focus on two specific algorithmic domains: Boolean logic and modular arithmetic. While these are foundational tasks (e.g., any computable function can in principle be represented as a Boolean circuit), they do not encompass the full diversity of algorithmic tasks in practice. Generalizing this approach to broader classes of problems remains an open challenge.

Conclusion. We introduce a modular and interpretable framework for specifying exact algorithmic computations in universal transformers. By formulating algorithmic problems as circuits and mapping their structure onto transformer mechanisms, we demonstrate that transformers can solve Boolean and modular arithmetic problems of arbitrary depth without training. This approach provides a principled solution to compositional and length generalization problems, and offers a foundation for studying the learnability and interpretability of algorithmic reasoning in neural models. Our results suggest that structured architectural design can enhance the reliability and transparency of transformer-based systems, opening avenues for future research in scalable algorithmic reasoning and mechanistic understanding.

References

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Prasoon Varshney, Makesh Narsimhan, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Putterman, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Shaona Ghosh, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Chris Alexiuk, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben, Yonatan Geifman, and Eric Chung. Llama-Nemotron: Efficient Reasoning Models, June 2025. URL `http://arxiv.org/abs/2505.00949`. arXiv:2505.00949 [cs].

Thomas Chen, Tengyu Ma, and Zhiyuan Li. Non-Asymptotic Length Generalization, June 2025. URL `http://arxiv.org/abs/2506.03085`. arXiv:2506.03085 [cs].

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, January 2025. URL `http://arxiv.org/abs/2501.12948`. arXiv:2501.12948 [cs].

Gregoire Deletang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, and Pedro A. Ortega. Neural Networks and the Chomsky Hierarchy. *International Conference on Learning Representations*, September 2022. URL https://openreview.net/forum?id=WbxHAzkeQcn.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang (Lorraine) Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and Fate: Limits of Transformers on Compositionality. *Advances in Neural Information Processing Systems*, 36:70293–70332, December 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/deb3c28192f979302c157cb653c15e90-Abstract-Conference.html.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, and Tom Conerly. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. URL https://www.sciencedirect.com/science/article/pii/0893608089900208. Publisher: Elsevier.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality Decomposed: How do Neural Networks Generalise? *Journal of Artificial Intelligence Research*, 67:757–795, April 2020. ISSN 1076-9757. doi: 10.1613/jair.1.11674. URL https://www.jair.org/index.php/jair/article/view/11674.

Johan Håstad. *Computational limitations for small depth circuits*. PhD Thesis, Massachusetts Institute of Technology, 1986. URL https://dspace.mit.edu/bitstream/handle/1721.1/150504/15748273-MIT.pdf?sequence=1&isAllowed=y.

Takuya Ito, Soham Dan, Mattia Rigotti, James Kozloski, and Murray Campbell. On the generalization capacity of neural networks during generic multimodal reasoning. *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zyBJodMrn5&noteId=zyBJodMrn5.

Takuya Ito, Murray Campbell, Lior Horesh, Tim Klinger, and Parikshit Ram. Quantifying artificial intelligence through algorithmic generalization. *Nature Machine Intelligence*, 7(8):1195–1205, August 2025. ISSN 2522-5839. doi: 10.1038/s42256-025-01092-w. URL https://www.nature.com/articles/s42256-025-01092-w. Publisher: Nature Publishing Group.

Samy Jelassi, Stéphane d'Ascoli, Carles Domingo-Enrich, Yuhuai Wu, Yuanzhi Li, and François Charton. Length Generalization in Arithmetic Transformers, June 2023. URL http://arxiv.org/abs/2306.15400. arXiv:2306.15400 [cs].

Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching Arithmetic to Small Transformers. International Conference on Learning Representations, October 2023. URL https://openreview.net/forum?id=dsUB4bst9S.

William Merrill and Ashish Sabharwal. The Expressive Power of Transformers with Chain of Thought. International Conference on Learning Representations, 2024. URL https://openreview.net/forum?id=NjNGlPh8Wh.

Mayank Mishra, Matt Stallone, Gaoyuan Zhang, Yikang Shen, Aditya Prasad, Adriana Meza Soria, Michele Merler, Parameswaran Selvam, Saptha Surendran, Shivdeep Singh, Manish Sethi, Xuan-Hong Dang, Pengyuan Li, Kun-Lung Wu, Syed Zawad, Andrew Coleman, Matthew White, Mark Lewis, Raju Pavuluri, Yan Koyfman, Boris Lublinsky, Maximilien de Bayser, Ibrahim Abdelaziz, Kinjal Basu, Mayank Agarwal, Yi Zhou, Chris Johnson, Aanchal Goyal, Hima Patel, Yousaf Shah, Petros Zerfos, Heiko Ludwig, Asim Munawar, Maxwell Crouse, Pavan Kapanipathi, Shweta Salaria, Bob Calio, Sophia Wen, Seetharami Seelam, Brian Belgodere, Carlos Fonseca, Amith Singhee, Nirmit Desai, David D. Cox, Ruchir Puri, and Rameswar Panda. Granite Code Models: A Family of Open Foundation Models for Code Intelligence, May 2024. URL http://arxiv.org/abs/2405.04324. arXiv:2405.04324 [cs].

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context Learning and Induction Heads, September 2022. URL http://arxiv.org/abs/2209.11895. arXiv:2209.11895 [cs].

Parikshit Ram, Tim Klinger, and Alexander G. Gray. What makes Models Compositional? A Theoretical View: With Supplement, May 2024. URL http://arxiv.org/abs/2405.02350. arXiv:2405.02350 [cs].

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models, April 2024. URL http://arxiv.org/abs/2402.03300. arXiv:2402.03300 [cs].

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, Stella Biderman, Adria Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, Eric J. Michaud, Stephen Casper, Max Tegmark, William Saunders, David Bau, Eric Todd, Atticus Geiger, Mor Geva, Jesse Hoogland, Daniel Murfet, and Tom McGrath. Open Problems in Mechanistic Interpretability, January 2025. URL http://arxiv.org/abs/2501.16496. arXiv:2501.16496 [cs].

Peter Shaw, James Cohan, Jacob Eisenstein, Kenton Lee, Jonathan Berant, and Kristina Toutanova. ALTA: Compiler-Based Analysis of Transformers. *Transactions on Machine Learning Research*, November 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=h751wl9xiR.

Ruoqi Shen, Sebastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. Positional Description Matters for Transformers Arithmetic. *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZMuPAOY8Oz.

Sania Sinha, Tanawan Premsri, and Parisa Kordjamshidi. A Survey on Compositional Learning of AI Models: Theoretical and Experimental Practices, June 2024. URL http://arxiv.org/abs/2406.08787. arXiv:2406.08787 [cs].

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What Formal Languages Can Transformers Express? A Survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, May 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00663. URL https://doi.org/10.1162/tacl_a_00663.

Gail Weiss, Yoav Goldberg, and Eran Yahav. Thinking Like Transformers. In *Proceedings of the 38th International Conference on Machine Learning*, pages 11080–11090. PMLR, July 2021. URL https://proceedings.mlr.press/v139/weiss21a.html. ISSN: 2640-3498.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report, May 2025. URL http://arxiv.org/abs/2505.09388. arXiv:2505.09388 [cs].

Andy Yang, David Chiang, and Dana Angluin. Masked Hard-Attention Transformers Recognize Exactly the Star-Free Languages, October 2024. URL http://arxiv.org/abs/2310.13897. arXiv:2310.13897 [cs].

Hattie Zhou, Arwen Bradley, Etai Littwin, Noam Razin, Omid Saremi, Joshua M. Susskind, Samy Bengio, and Preetum Nakkiran. What Algorithms can Transformers Learn? A Study in Length Generalization. International Conference on Learning Representations, 2024a. URL https://openreview.net/forum?id=AssIuHnmHX.

Yongchao Zhou, Uri Alon, Xinyun Chen, Xuezhi Wang, Rishabh Agarwal, and Denny Zhou. Transformers Can Achieve Length Generalization But Not Robustly. International Conference on Learning Representations, April 2024b. URL https://openreview.net/forum?id=DWkWIh3vFJ.

A Model Details

Notation. We denote scalars as lowercase italics (x), column vectors in lowercase boldface ($\boldsymbol{x}$), row vectors as its transpose ($\boldsymbol{x}^\top$), and matrices as uppercase italics (X).We denote $\boldsymbol{x}^i$ as the i^{th} element of vector $\boldsymbol{x}$, X^i as the row vector in the i^{th} row, and $X^{i,j}$ as the element the i^{th} row and j^{th} column. We denote $\mathbf{1}_d$ as a d-dimensional column vector, and $\mathbf{1}_{d\times d}$ as a matrix of ones, and $\boldsymbol{I}_d$ as a $d \times d$ identity matrix.

A.1 Token embedding

We use one-hot token embeddings, where $\boldsymbol{e_i}^\top \in \{0,1\}^{1\times d}$ is a basis vector with a 1 in the i^{th} position and 0s otherwise. Thus, the model's embedding dimensionality corresponds exactly to the size of the vocabulary. For Boolean formulas, $d = 5$, where the tokens in the vocabulary correspond to `0`, `1`, $\wedge$, $\vee$, $\sim$). For arithmetic formulas, $d = 12$, and the tokens correspond to integers 0-9, $+$, $\times$. A given formula of n tokens is then provided to the transformer as a concatenation of token embeddings, i.e., $X_0 = [\boldsymbol{e_i}, ..., \boldsymbol{e_j}]^\top \in \{0,1\}^{n\times d}$. X_0 is then used as the input in to the transformer's attention mechanism. Note that for simplicity, we do not include parentheses in the model vocabulary, as the circuit encoding provides the same information as parentheses. Pragmatically, we assume the parentheses are either encoded as the relative positional encoding / attention mask.

A.2 Attention and syntactic parsing

In standard transformers, given a query $W_q \in \mathbb{R}^{d\times d}$, key $W_k \in \mathbb{R}^{d\times d}$, and value $W_v \in \mathbb{R}^{d\times d}$ matrix, the attention mechanism in transformers is

$$\text{Attn}(X_i) = (X_i W_q)(X_i W_k)^\top (X_i W_v)$$

For simplicity, here we assume no explicit positional encoding, softmax, or scaling factor (e.g., $\frac{1}{\sqrt{d_k}}$). However, studies in mechanistic interpretability abstract this attention mechanism into two separate functional circuits: the attention matrix, $QK^\top = (X_i W_q)(X_i W_k)^\top \in \mathbb{R}^{n\times n}$ and the readout matrix $XV = X_i W_v \in \mathbb{R}^{n\times d}$ [Elhage et al., 2021]. In particular, the $QK^\top$ serves as the circuit that routes information from source tokens to target tokens, and XV determines how attending to a source token influences the embedding of the downstream target token. This mimics how in circuits, input gates map to operator gates.

In our universal transformer model, we leverage the abstractions introduced by Elhage et al. [2021] to specify attention at the level of $QK^\top$. First, we only use a single attention head. Second, we set $QK^\top \in \{0,1\}^{n\times n}$ as a binary matrix (no softmax) that is specified by the circuit's adjacency matrix. (Note that in our circuit's adjacency matrix, we additionally encode self-connections for operator gates.) In practice, in our model implementation, we use $QK^\top$ as an attention mask, and specify $W_q = W_k = \mathbf{1}_{d\times d}$. Note, that while we provide $QK^\top$ as a circuit-specific attention mask in our model implementation, in principle, obtaining the correct attention weights matrix $QK^\top$ can be achieved by incorporating a well-crafted relative positional encoding and leveraging a hardmax (rather than a softmax) on the attention matrix [Strobl et al., 2024]. This formulation of the attention mechanism can then be used to study the learnability conditions of this universal transformer in future work without the use of the attention mask we use here.

Finally, we set $W_v = \boldsymbol{I}_d$. Since our token embeddings are one-hot encodings, by setting W_v as the identity matrix, our attention mechanism effectively routes one-hot embeddings (in the transformer layer's inputs) to produce multi-hot encodings (i.e., a vector of token counts) in the post-attention embedding layer. This produces a superposition of token embeddings (e.g., see Fig. 1D). Moreover, because token embeddings are routed via the attention weights (exactly corresponding to the circuit's edges), downstream token embeddings are at most a count vector of a depth-1 circuit. Note, however, that operator gates are sometimes mapped to other internal operator gates (for circuits with greater than depth 1; e.g., see Fig. 1D). This results in a vector of counts that cannot be evaluated by a finite table. We specify how this will be handled by the MLP in the subsequent section.

A.3 MLP and semantic evaluation

The post-attention embedding layer provides at most a count vector of a depth-1 circuit, as each target token has at most fan-in 3 (2 child nodes and itself). Therefore, the MLP's required functionality

essentially reduces to that of a finite lookup table (corresponding to a Boolean truth table or a 2-operand modular arithmetic table). However, in some cases, the attention matrix routes information that produces a multi-hot encoding that cannot be evaluated. This occurs when an intermediate operator gate has not yet been computed, yet that operator is routed to its parent gate (which itself is an operator). This produces situations in which the post-attention count vector contains the incorrect number of operands (and/or operators). In these situations (when there is no valid expression to be evaluated), the MLP routes a residual connection from the pre-attention embedding layer (see f in Algorithms 1 and 2; Appendix). In the present implementation, we use tensor indexing to specify the function of f rather than an MLP. This was done for the ease of "hard-coding" parameters as a tensor, rather than an MLP. However, given the finite nature of the MLP's semantic evaluation, it is straightforward to train the MLP to compute a depth-1 circuit for either Boolean or arithmetic formulas, since MLPs are universal function approximators capable of learning Boolean truth tables [Hornik et al., 1989, Håstad, 1986].

A.4 Integrating components into a universal transformer

We have provided specifications for the token embeddings, attention mechanism, and MLP, the three ingredients required for our model. By putting together each of these components, we compute a depth-1 circuit in every transformer layer. In essence, at each transformer iteration, if an operator gate receives two operands, then the operator token is updated to the correct operand (via the MLP) (Fig. 1A,B). If an operator token cannot be updated because its count vector does not correspond to a valid depth-1 circuit (e.g., its inputs are themselves operator gates), then it copies the token embedding to the next layer via residual connection. To retrieve the correct answer for a circuit of depth-k, we inspect the output gate after k iterations. The output gate is defined as the gate with an out-degree of 0 in the circuit's adjacency matrix. The algorithm corresponding to each transformer forward pass is detailed in the Appendix (Algorithm 1 for Boolean circuits, and Algorithm 2 for arithmetic circuits).

Algorithm 1 Universal transformer for Boolean circuit evaluation

Definition: Let $\mathcal{E}_{operands} = \{\boldsymbol{e_1}^\top, \boldsymbol{e_2}^\top\}$ denote input gates `1` and `0`, respectively, where $\boldsymbol{e_i}^\top \in \{0,1\}^{1\times d}$ is a basis vector with a 1 in the i^{th} position, and 0s otherwise (i.e., a one-hot vector). Here, $d = 5$.

Definition: Let $\mathcal{E}_{operators} = \{\boldsymbol{e_3}^\top, \boldsymbol{e_4}^\top, \boldsymbol{e_5}^\top\}$ denote operator gates $\wedge$ (`AND`), $\vee$ (`OR`), $\sim$ (`NOT`), respectively.

Definition: Let $X_i \in \mathbb{N}_0^{n\times d}$ be the token embedding of the i^{th} transformer layer (each token $\boldsymbol{e}_i^\top$ is a row vector), where n refers to the number of tokens in the context window.

Definition: Let $QK^\top \in \{0,1\}^{n\times n}$ be the attention weights, which exactly encode the circuit's binary adjacency matrix. Operator gates additionally have self-connections. We restrict circuits to be fan-in 2.

Definition: Let X^o denote the token containing the output gate. o can be determined by identifying the index of $QK^\top$ with out-degree 0.

Definition: Let $V = X_i \cdot W_V$ be the values matrix, where $W_V = \boldsymbol{I}_d$.

Definition: Let f be a token-wise multilayer perceptron that exactly computes a depth-1 circuit (Boolean truth table).

$i \leftarrow 0$
while $\exists X_i^j \in \mathcal{E}_{operators}, \forall j$ **do** ▷ Each iteration is a single transformer layer
  $X_{i,a} \leftarrow X_i$ ▷ Store pre-attention token embeddings
  $X_{i,b} \leftarrow X_{i,a} \cdot QK^\top \cdot V$ ▷ Map source gates to target gates via attention
  $X_{i+1} \leftarrow f(X_{i,b})$ ▷ Compute depth-1 circuit
  $i \leftarrow i + 1$
end while
Output $\leftarrow X_i^o$

More specifically, for an embedding X_i^j for token j, the function f is defined as:

$$f(X_{i,b}^j) = \begin{cases} \boldsymbol{e_1}^\top & \text{if } \texttt{1}\text{, i.e.,} \\ & X_{i,b}^j = \boldsymbol{e_1}^\top + \boldsymbol{e_1}^\top + \boldsymbol{e_3}^\top\text{, corresponding to } 1 \wedge 1 \\ & X_{i,b}^j = \boldsymbol{e_1}^\top + \boldsymbol{e_1}^\top + \boldsymbol{e_4}^\top\text{, corresponding to } 1 \vee 1 \\ & X_{i,b}^j = \boldsymbol{e_1}^\top + \boldsymbol{e_2}^\top + \boldsymbol{e_4}^\top\text{, corresponding to } 1 \vee 0 \\ & X_{i,b}^j = \boldsymbol{e_2}^\top + \boldsymbol{e_5}^\top\text{, corresponding to } \sim 0 \\ \boldsymbol{e_2}^\top & \text{if } \texttt{0}\text{, i.e.,} \\ & X_{i,b}^j = \boldsymbol{e_1}^\top + \boldsymbol{e_2}^\top + \boldsymbol{e_3}^\top\text{, corresponding to } 1 \wedge 0 \\ & X_{i,b}^j = \boldsymbol{e_2}^\top + \boldsymbol{e_2}^\top + \boldsymbol{e_3}^\top\text{, corresponding to } 0 \wedge 0 \\ & X_{i,b}^j = \boldsymbol{e_2}^\top + \boldsymbol{e_2}^\top + \boldsymbol{e_4}^\top\text{, corresponding to } 0 \vee 0 \\ & X_{i,b}^j = \boldsymbol{e_1}^\top + \boldsymbol{e_5}^\top\text{, corresponding to } \sim 1 \\ X_{i,a}^j & \text{otherwise, i.e., not a valid Boolean expression} \end{cases}$$

Algorithm 2 Universal transformer for modular arithmetic evaluation

Definition: Let $\mathcal{E}_{operands} = \{\boldsymbol{e_1}^\top, \boldsymbol{e_2}^\top ..., \boldsymbol{e_{10}}^\top\}$ denote input gates 1, 2, ..., 0, respectively, where $\boldsymbol{e_i}^\top \in \{0,1\}^{1\times d}$ is a basis vector with a 1 in the i^{th} position, and 0s otherwise (i.e., a one-hot vector). Here, $d = 12$.

Definition: Let $\mathcal{E}_{operators} = \{\boldsymbol{e_{11}}^\top, \boldsymbol{e_{12}}^\top\}$ denote operator gates $+$ and $\times$ (for modular addition and multiplication), respectively.

Definition: Let $X_i \in \mathbb{N}_0^{n\times d}$ be the token embedding of the i^{th} transformer layer (each token $\boldsymbol{e}_i^\top$ is a row vector), where n refers to the number of tokens in the context window.

Definition: Let $QK^\top \in \{0,1\}^{n\times n}$ be the attention weights, which exactly encode the circuit's binary adjacency matrix. Operator gates additionally have self-connections. We restrict circuits to be fan-in 2.

Definition: Let X^o denote the token containing the output gate. o can be determined by identifying the index of $QK^\top$ with out-degree 0.

Definition: Let $V = X_i \cdot W_V$ be the values matrix, where $W_V = \boldsymbol{I}_d$.

Definition: Let f be a token-wise multilayer perceptron that exactly computes a depth-1 circuit (2-operand modular arithmetic).

$i \leftarrow 0$
while $\exists X_i^j \in \mathcal{E}_{operators}, \forall j$ **do** ▷ Each iteration is a single transformer layer
  $X_{i,a} \leftarrow X_i$ ▷ Store pre-attention token embeddings
  $X_{i,b} \leftarrow X_{i,a} \cdot QK^\top \cdot V$ ▷ Map source gates to target gates via attention
  $X_{i+1} \leftarrow f(X_{i,b})$ ▷ Compute depth-1 circuit
  $i \leftarrow i + 1$
end while
Output $\leftarrow X_i^o$

More specifically, for an embedding X_i^j for token j, the function f is defined as:

$$f(X_{i,b}^j) = \begin{cases} \boldsymbol{e_i}^\top \in \mathcal{E}_{operands} & \text{compute two-variable modular arithmetic} \\ X_{i,a}^j & \text{otherwise, i.e., not a valid two-variable arithmetic expression} \end{cases}$$