In search of simplicity: a case study in ML and LLM assisted applications of ATP in mathematics

Alexei Lisitsa

University of Liverpool,
A.Lisitsa@liverpool.ac.uk

Introduction

The Andrews-Curtis Conjecture (ACC) [1], well-known in combinatorial group theory and topology, asserts that any balanced presentation of the trivial group can be reduced to a trivial presentation using a sequence of basic transformations. However, the conjecture is widely believed to be false, with infinitely many potential counterexamples for which no simplifications are known. This leads to a computationally challenging algorithmic problem: given a balanced presentation of the trivial group, determine a sequence of transformations that simplifies it. Computational approaches to this problem include generic search algorithms [4, 12], genetic algorithms [10, 13], techniques from computational group theory [3, 11], and, more recently, reinforcement learning [12]. In [5, 6, 7], we introduced an approach that utilizes automated deduction in first-order logic to search for AC trivializations. This method has proven to be highly competitive: it successfully simplified all known presentations that could be simplified by any other method (achieving a form of ”relative completeness”); it discovered new simplifications previously unknown [6, 8]; and it demonstrated high efficiency, producing simplifications of thousands steps [8, 9].

Automated theorem proving for ACC and Machine Learning

Recently, machine learning—specifically reinforcement learning—has been successfully applied to the search for AC simplifications [12]. We believe that our automated reasoning approach to AC simplification search presents new opportunities for integration with machine learning (ML) methods, including, but not limited to, Large Language Models (LLMs).

Balanced presentations of trivial groups as a playground for learning guided theorem proving. There are several infinite parametric families of balanced group presentations representing trivial groups. Notably these include the Akbulut-Kirby family $AK_n = \langle a, b \mid a^n b^{-(n+1)}, abab^{-1}a^{-1}b^{-1} \rangle$, $n \geq 2$ the Miller-Schupp family $MS_n(w) = \langle a, b \mid a^{-1}b^n ab^{-(n+1)}, a^{-1}w \rangle$, where $n \geq 1$; and w is a word which has exponent sum 0 on a; and Gordon family $G_{n,m,p,q} = \langle a, b \mid a^{-1}[a^n, b^m], b^{-1}[b^p, a^q] \rangle$, where $n, m, p, q \in Z$ and $[x, y] = xyx^{-1}y^{-1}$. While AC-simplifiability for some values of parameters is known in each family, the simplifiability of all presentations in any of the AK, MS and G families remains largely open. Thus, one can generate numerous automated theorem proving tasks for a large set of presentations within any given family, apply automated theorem proving to search for simplifications, and then use both successful and unsuccessful attempts to train a Learning-Guided Automated Reasoning System, such as those reviewed in [2].

LLMs assisted implementations for data processing tasks The automated reasoning approach for searching AC simplifications leverages the power of existing automated theorem provers, eliminating the need for a custom search procedure. However, it does require data preparation—formulating the problem as a theorem-proving task—and post-processing—extracting simplification sequences from successful proofs. We have conducted initial experiments using Large Language Models (LLMs) to implement the data processing tasks mentioned above. Specifically, we used LLM [1] to generate Perl scripts [2] for creating datasets formatted in Prover9 syntax, enabling theorem-proving tasks aimed at finding AC simplifications for the AK and MS families. The scripts take specifications of a presentation or a subset of presentations by defining a range of parameters and then generate a corresponding dataset of theorem-proving tasks, each stored in a separate file. Admittedly, generating such data is a relatively simple task that could be implemented directly without the help of LLMs. However, we would like to highlight several key advantages of this approach. First, the implementation required minimal effort, as tasks were specified through free-form dialogue in semi-technical natural language. Moreover, LLMs demonstrated an exceptional ability to automatically handle conversation context and adjust the level of abstraction when discussing design and implementation. For example, when specifying the required properties of group presentations, multiple definitions and representations of relators were used seamlessly: as words in the alphabet a, b, a', b'; as shorthand for terms in group theory, omitting explicit multiplication operations and structured brackets; and as symbolic notation using exponentiation (e.g., `a^{3}b^{-4}`). All naturally occurring and assumed equivalences, as well as case-based explanations and generalizations (e.g., "... and similarly for any n"), were handled correctly. LLMs assissted implementations of generatiom for G and for simplification sequences extraction from proofs are ongoing and will be made available upon completion.

LLMs for simplification sequences understanding Understanding the behavior of AC-simplification sequences obtained through Automated Theorem Proving (ATP) or other methods remains a significant challenge. Beyond the goal of (dis)proving the Andrews-Curtis Conjecture (ACC), the ultimate objective is to develop arguments that support human proofs of simplifiability for infinite (sub)families of group presentations. While we have not yet succeeded in this endeavor, we would like to acknowledge the support of LLMs in generating LaTeX source code for a simple diagram visualizing the so-called merging behavior of simplification sequences $MS_n(w_*)$ for $n = 3 \dots 9$ in [9].

[1] Chat GPT 4o

[2] available by request

Bibliography

[1] J. Andrews and M.L. Curtis. Free groups and handlebodies. *Proc. Amer. Math. Soc.*, 16:192–195, 1965.

[2] Lasse Blaauwbroek, David Cerna, Thibault Gauthier, Jan Jakubův, Cezary Kaliszyk, Martin Suda, and Josef Urban. Learning guided automated reasoning: A brief survey. arxiv: 2403.04017, 2024.

[3] George Havas and Colin Ramsay. Andrews-Curtis and Todd-Coxeter proof words. Technical report, in Oxford. Vol. I, London Math. Soc. Lecture Note Ser, 2001.

[4] George Havas and Colin Ramsay. Breadth-first search and the Andrews-Curtis conjecture. *International Journal of Algebra and Computation*, 13(01):61–68, 2003.

[5] Alexei Lisitsa. First-order theorem proving in the exploration of Andrews-Curtis conjecture. *TinyToCS*, 2, 2013.

[6] Alexei Lisitsa. The Andrews-Curtis Conjecture, Term Rewriting and First-Order Proofs. In *Mathematical Software - ICMS 2018 - 6th International Conference, South Bend, IN, USA, July 24-27, 2018, Proceedings*, pages 343–351, 2018.

[7] Alexei Lisitsa. Automated reasoning for the Andrews-Curtis conjecture. In *AITP 2019, Fourth Conference on Artificial Intelligence and Theorem Proving, Abstracts of the Talks April 7–12, 2019, Obergurgl, Austria*, pages 82–83, 2019.

[8] Alexei Lisitsa. New Andrews–Curtis trivializations for Miller–Schupp group presentations. *Examples and Counterexamples*, 6:100168, 2024.

[9] Alexei Lisitsa. Automated theorem proving reveals a lengthy Andrews-Curtis trivialization for a Miller-Shupp schupp trivial group presentation. pre-print available at SSRN: https://ssrn.com/abstract=5100345 or http://dx.doi.org/10.2139/ssrn.5100345, 2025.

[10] Alexei D. Miasnikov. Genetic algorithms and the Andrews-Curtis conjecture. *International Journal of Algebra and Computation*, 09(06):671–686, 1999.

[11] Dmitry Panteleev and Alexander Ushakov. Conjugacy search problem and the Andrews-Curtis conjecture. *ArXiv e-prints*, page arXiv:1609.00325, September 2016.

[12] Ali Shehper, Anibal M. Medina-Mardones, Bartłomiej Lewandowski, Angus Gruen, Piotr Kucharski, and Sergei Gukov. What makes math problems hard for reinforcement learning: a case study. arxiv:2408.15332, 2024.

[13] Jerry Swan, Gabriela Ochoa, Graham Kendall, and Martin Edjvet. Fitness Landscapes and the Andrews-Curtis Conjecture. *IJAC*, 22(2), 2012.