The Importance of Teaching Logic to Computer Scientists and Electrical Engineers

Paul Mayer, *Member, IEEE,* Rich Baraniuk, *Member, IEEE*

***Abstract*—It is argued that logic, and in particular mathematical logic, should play a key role in the undergraduate curriculum for students in the computing fields, which include electrical engineering (EE), computer engineering (CE), and computer science (CS). This is based on 1) the history of the field of computing and its close ties with logic, 2) empirical results showing that students with better logical thinking skills perform better in tasks such as programming and mathematics, and 3) the skills students are expected to have in the job market. Further, the authors believe teaching logic to students explicitly will improve student retention, especially involving underrepresented minorities. Though this work focuses specifically on the computing fields, these results demonstrate the importance of logic education to STEM (science, technology, engineering, and mathematics) as a whole.**

***Index Terms*—IEEE, logic, engineering education, computer science education, student attrition, retention,**

I. Introduction

LOGIC is a branch of philosophy concerned with drawing valid inferences, and in the West can be traced back to the work of Aristotle [1]. Aristotle's work was an important step towards formalizing logic, but as Frege and Leibniz complained, logical relations made in natural languages can be vague and ambiguous [2]. As a result, Frege and Leibniz suggested natural languages be replaced by formal languages, a goal which eventually led the development of what is now called mathematical logic. According to logician Jon Barwise, "Modern mathematical logic has its origins in the dream of Leibniz of a universal symbolic calculus which could encompass all mental activity of a logically rigorous nature, in particular, all of mathematics [3]." Significant progress towards Leibniz's dream was made in the 19th century by Augustus De Morgan [4] and George Boole [5], both of whom used mathematics to describe Aristotle's syllogisms [6]. This helped to formalize the field of mathematical logic.

About a century later, Claude Shannon showed the applicability of Boole's Laws of Thought [7] to designing electronics in his masters thesis titled "A symbolic analysis of relay and switching circuits" [8]. Mathematician Herman Goldstine considered Shannon's work "one of the most important master's theses ever written, [helping] change digital circuit design from an art to a science [9]." It was Shannon's knowledge of logic that allowed him to establish the fundamentals of digital circuit design over a decade before the invention of the transistor in 1947 [10].

Department of Electrical and Computer Engineering, Rice University, Houston Texas 77098 pmm3@rice.edu

Many of the pioneers in electrical engineering and computer science had strong foundations in logic, which is necessary to understand the history of these fields and their subsequent developments. As a result, the authors of this paper propose that teaching logic should form a necessary and foundational part of the undergraduate curriculum in these areas, which are subsequently called the computing fields. Unfortunately, there has been a curricular shift away from deductive logic, to a greater focus on induction, in large part due to the success of machine learning, which is primarily inductive [11]. If students are to understand and contribute to the computing fields, they need to understand topics from logic and mathematical logic; in particular, deductive reasoning, first-order logic, and set theory.

II. Related Work

A. Computational Thinking

The term "computational thinking" refers to an idea introduced by computer scientist Jeannette Wing, who argues in a 2006 paper that the underlying thought processes are more important for success in computer science than mere programming [12]. This is similar to an argument made by Dijkstra in 1988 [13], who suggested introductory programming courses should not allow students to test their programs "in order to drive home the message that this introductory programming course is primarily a course in formal mathematics." While Dijkstra's suggestion may be too radical for many (and there are likely additional goals for introductory programming courses than purely teaching formal mathematics [14]), he is correct to emphasize the role of valid thinking in writing programs.

The term computational thinking is not always well defined [15, 16], however it usually includes logical thinking and principles assumed to be useful to computer scientists such as abstraction [17]. The authors of this paper agree with Wing: education in the computing fields should emphasize the thought processes necessary for tasks such as programming over programming skill alone. The teaching of logic, and in particular mathematical logic including first order logic and set theory [3], can thus be seen as teaching explicit topics relevant to computational thinking.

B. Mathematical Thinking

1) What is Mathematical Thinking: Mathematical Thinking involves reasoning about mathematics using topics from logic, and in particular mathematical logic. Many of these topics are necessary for understanding engineering mathematics, such as deductive reasoning relevant to proof theory [18], and set theory relevant to linear algebra [19]. Given the importance of mathematics for the computing fields, in particular for specializations such as communication, statistics, signal processing, and machine learning, it is vital that students are to think mathematically. For instance, a 2010 paper by Lauri Hietalahti goes through several examples where mathematical reasoning is relevant to solving problems in power systems, concluding "mathematical thinking is a vital skill needed in electrical engineering [20]."

2) Mathematical Logic and Proofs: A common way of trying to teach and mathematical thinking involves assigning proofs. Mathematician Susanna Epp claims the reason computer science (and, by extension, engineering) students are tasked with doing proofs is to teach them to "operate in a mathematically sophisticated environment...[and enhance their] logical reasoning [21]." Furthermore, as Steve Reeves points out, "The activity of constructing and reasoning about programs in not all that different from the activity of constructing and reasoning about proofs [22]."

However, if students are to gain any benefit from proof writing, they must first understand the underlying logical and mathematical principles that give rise to valid vs invalid proofs. Unfortunately, as Epp observed when teaching introductory proof writing courses for mathematics and computer science students, "very few have an intuitive understanding of [logical] reasoning principles [21]." She explains:

> [M]y students' difficulties were much more profound than I had imagined. Indeed, I was almost overwhelmed by the poor quality of their proof-writing attempts. Often their efforts consisted of little more than a few disconnected calculations and imprecisely or incorrectly used words and phrases that did not even advance the substance of their cases. My students seemed to live in a different logical and linguistic world from the one I inhabited, a world that made it very difficult for them to engage in the kind of abstract mathematical thinking I was trying to help them learn. [21]

Epp thus concluded that instruction in formal logical reasoning skills would help lay the required foundation for proof writing and understanding. The authors of this paper agree: teaching proof writing *can* be a promising way to further develop logical thinking once students understand the deductive principles necessary for doing proofs *and* the relevance of these proofs to their educational goals.

3) Mathematical Thinking and Problem Solving: Mathematical thinking is often linked to problem solving, a key skill for future engineers and designers [23]. In fact, mathematician Paul Halmos went so far as to claim the heart of mathematics is "problems and solutions [24]." Both the Accreditation Board for Engineering and Technology (ABET) and the Engineering Concil (EC) list problem solving as a criteria for accreditation [25]. Furthermore, the National Science Foundation (NSF) established various engineering education coalitions for addressing topics including open-end problem solving and the development of a curriculum based on first principles [26], topics related to developing mathematical thinking. If the goal of education in the computing fields is teaching students to solve problems, and mathematical thinking and thus mathematical logic can help students solve problems, then such education should include topics from mathematical logic that can improve their thinking and help them understand the mathematics required for their major.

III. Logic in the Computing Fields

A. Logic in the History of Computing

Many of the most influential figures in the field of computing were trained in and contributed to mathematical logic. For instance, Alonzo Church, an important figure in computer science, was a founding editor of the Journal of Symbolic Logic, publishing "A Bibliography of Symbolic Logic" in 1936 [27] and "An Introduction to Mathematical Logic" in 1956 [28]. His doctoral student, Alan Turing, developed ideas around algorithms and computation with what is now called Turing machines, getting his PhD in 1938 with a dissertation titled "Systems of Logic Based on Ordinals" [29]. Some believe Turing's 1936 work [30] was directly inspired by logician Kurt Godel's 1931 incompleteness theorems [31, 32]. Another of Church's doctoral students was Stephen Cole Kleene, who also studied mathematical logic. Kleene's thesis was titled "A Theory of Positive Integers in Formal Logic [33]". Kleene's pioneered the branch of mathematical logic called recursion theory, and helped to form basis of what are now regular expressions. Much of Kleene's syntax, such as the star, is still used by computer scientists when pattern matching [34].

John Von Neumann, before his work on computer architecture [35] also did work in mathematical logic. In 1923, he published a paper on the construction of the natural numbers [36] and his 1925 dissertation produced an axiomatization of set theory [37]. Fellow physicist and mathematician Freeman Dyson attributed Von Neuman's incredible success to his "unique gift as a mathematician was to transform problems in all areas of mathematics into problems of logic" [38].

B. Logic's Lasting Legacy

Many modern ideas in the computing fields are also inspired by ideas in logic. Edgar Codd based the relational model for database systems on set theory and propositional logic [39]. Ole-Johan Dahl, one of the fathers of object oriented programming (along with Kristen Nygaard), taught formal methods for 30 years, believing "computer science students should know the principles of program reasoning, and that this would make them better programmers even without performing detailed verification [40]." Stephen Cook, who formalized the notions of polynomial time and NP-completeness [41] (independently discovered by Leonid Levin [42]) also published in propositional calculus [43] and the logic of proofs [44].

One can see from their work that these individuals were able to make contributions to computing due, at least in part, to their understanding of logic. Furthermore, since logic was instrumental to the development of computing and is needed for understanding its historical and current concepts, the authors of this paper believe it should play a critical role in the computing curriculum.

IV. Empirical Support for Teaching Logic

Research suggests students with stronger logical and mathematical thinking skills perform better on tasks in the computing fields such as programming. One study, looking at high school students in Turkey, found “students’ mathematical and logical thinking skills predicted their computational thinking skills [45].” Another study, performed on college freshmen, found a significant correlation between logical thinking skills and programming achievement, concluding that “Learners with strong logical thinking skills can program more efficiently than those without [46].”

Despite these correlations, until recently it was less clear that logical thinking could be improved by explicitly studying logic. A 1936 study performed on adolescent boys found improved test scores for students who had an hour of logic instruction every week for three months, suggesting teaching logic can improve thinking. On the other hand, a 1986 study looking at students’ performance on the famous Wason selection task [47], concluded “deductive reasoning is not likely to be improved by training on standard logic [48].” Some researchers expressed doubt that the Wason selection task measures reasoning due to how much success rates depend on the relevance of the task in question [49]. If true, this suggests [48] is premature, if not outright wrong, to conclude deductive reasoning cannot improved with instruction in logic.

Still, Cheng’s findings are informative in that they suggest students may not initially see the relevance of learning logic to their desired educational goals without proper motivation. Educational research on task value suggests students’ educational outcomes are influenced by how important and relevant they perceive course material [50]. In other words, the effectiveness of teaching logic may depend on the extent students believe *learning* logic will help their academic or professional career. This is discussed this more in Section VI.

More recent work has supported the hypothesis that logical reasoning can be improved. A 2016 study by Attridge et. al found a significant increase in the abstract reasoning skills of students who had some previous experience in logic [51]. A 2022 study found that teaching students concepts from propositional logic, such as deductive arguments and Venn and Euler diagrams, improved their logical reasoning skills [52]. The authors conclude the explicit instruction of logic will improve students’ performance on tasks in the computing fields such as programming, algorithmic reasoning, and hardware design.

V. The Importance of Logic for Industry

A. Logic and Hardware Design

Understanding logic is not just essential for those wanting to become research scientists in academia - it is also important for students pursuing a career in industry. If the curriculum of computing fields is to mirror what is expected of recent graduates, particular attention must be paid to the role of electronic design automation (EDA) which performs many tasks previously done by hardware engineers. Gone are the days when computer engineers explicitly place and route transistors on a PCB (printed circuit board). Instead hardware designers routinely use hardware description languages (HDLs), such as Verilog or VHDL to describe the desired behavior or logic of the device in question [53, 54]. HDL is sent to a logic synthesizer [55], where a netlist is generated. In some cases, the netlist is placed and routed onto a Field Programmable Gate Array (FPGA), or it is turned into masks which are etched for Application Specific Integrated Circuits (ASICs). This increased design abstraction means strong logical thinking skills are more important than they ever have been [20, 56, 57], since computers can arrange and route the hardware using optimization-based approaches once the behavior is specified. As a result, being able to formally and symbolically describe the desired logical behavior of a device is a necessary skill for those who want a career in hardware design and computer architecture.

B. Logic and Software Design

1) Logic and Programming: Strong mathematical logic skills are also important for those seeking employment designing software. This is because a computer, in the end, is a device built to follow a set of explicit instructions. Programming thus involves conveying what these instructions are in a precise manner. Since all errors in software can be traced back to human error and poor thought, it is important for programmers to be able to think rigorously and formally enough to avoid costly, and in some cases deadly, errors [13]. For a particularly harrowing incident of a fatal software bug, see the infamous Therac-25 [58]. Such errors can be avoided when programmers write code that is provably and verifiably correct.

Aswith hardware, there is increasing abstraction in software, in part due to the proliferation and adoption of high level and object oriented programming languages such as Python, Javascript, and C# [59]. These higher level languages remove many of the underlying hardware concepts such as pointers and registers, allowing more development time to be spent *describing* the desired behavior of a program than *implementing* it. Furthermore, due to advances in compiler technology, less development time is needed to test different implementations of the same algorithm, meaning more time can be spent writing algorithms than writing code.

This is not to say discussions of the underlying hardware or software mechanisms should be removed or de-emphasized from the curriculum. Instead instructors should focus students’ efforts and energy learning skills that transfer well to their future careers. The authors of this paper believe understanding logic is key for reasoning about algorithms and programming, and will clearly transfer to what students will be expected to do. In essence, logical thinking is much more future proof than specific programming languages, hardware, or packages which could be irrelevant in a few years.

2) Programming, Logic, and the Role of AI: Artificial Intelligence based programming tools, including large language models (LLMs) such as ChatGPT [60] perform well on inductive tasks, such as translating code from one programming language to another. However these AI models often struggle with deductive reasoning and propositional logic [61]. As a result, the quality of code generated from these models varies and there are no guarantees that the results produced are correct [62]. While AI may can be used as a tool to help programmers with mundane or repetitive tasks, authors propose programmers with strong logic and deductive reasoning skills need not fear replacement by AI.

C. Additional Benefits of Learning Logic

There may be additional benefits to learning logic beyond improving technical skills. One such area is writing: while many engineering students avoid writing classes [63] (perhaps even choosing engineering due to a preference for numbers over words), one book suggests many professional engineers spend over 40% of their working time writing [64]. Teaching students to write by emphasizing valid arguments and avoiding fallacies offers a clear path for improving the structure and clarity of writing [65]. Further, because the same principles of valid argumentation are those that produce valid hardware and software, a parallel can be made between clean hardware, clean code, and clean writing. In fact, many of deficiencies found in engineering report writing, such as inappropriate narrative and poor organization [66], can be framed as logical fallacies such as non-sequiturs. Once engineers can identify these issues, they can work towards eliminating them, improving the quality of their writing.

VI. Teaching Logic

A. Emphasizing the Role of Logic

Currently, topics from logic may be introduced in discrete math courses, often required for computer science majors, and digital design courses, usually required for electrical engineering and computer engineering majors. For instance, some discrete mathematics textbooks introduce propositional logic and set theory in the context of proofs [67], and many digital design textbooks introduce logical connectives and truth tables in the form of logic gates such as NANDS and XORs [55, 68].

One potential pitfall of this method of teaching logic is students may fail to see the connection between the underlying logical principles in in different courses and contexts. This creates a challenge to learning transfer [69] because these topics are only taught in the context of a single course or field of study. For instance the disjunction is fundamental the computing fields, but the different names, notations, and courses it is presented in may prevent students from recognizing they can leverage skills and experience they have already developed.

As a result, the authors of this paper suggest introducing topics such as first order logic *simultaneously* with the different contexts it applies to when it is introduced. For example, while teaching about propositions and logical connectives, instructors can show how these connectives are used in proofs, circuit design, HDL, machine code, and high level programming languages. This allows students to see the wide applicability of logic while they are learning it, increasing task value [50]. This is important because a common reason students choose engineering or computer science is so they can design and build things. Introducing logic alongside different applications gives students practice seeing the relevance of theory to practical design work. The above suggestion is consistent with suggestions in the literature for mitigating the common complaint that engineering contains too much math and theory: [70], for instance, recommends adopting a problem-based learning approach and include practice *alongside* theory (not as a replacement for it).

B. What to Teach and When?

The authors of this paper suggest teaching logic by beginning with Boolean Algebra, including the Law of the Excluded Middle and the Law of Non-Contradiction [71]. A connection can made between Boolean Algebra and the behavior of semiconductor devices such as transistors. From here, logical connectives such as disjunctions and conjunctions can be introduced, teaching students to represent compound logical statements on paper, with circuit diagrams, and in programming languages. Here, students can be reminded that the voltage along a wire or a bit stored in memory can represent whether a given proposition is true, and can practice design work in hardware and software with these basic building blocks. From here, set theory can be introduced, allowing for quantification and completing first-order logic [3]. Further topics that can be included involve that of state and memory, allowing for sequential logic [55] and a discussion of finite state machines [67].

VII. Logic and Attrition

A concern for the computing fields is attrition, the phenomenon of students "dropping out" of engineering and computer science to another field [72]. Attrition rates in electrical engineering and computer engineering are typically higher than all other engineering fields, with often more than half dropping out [73, 74]. Computer science is similar; one study found a 38% attrition rate in computer science, with this number varying by institution from about 30% to 60% [75, 76] These numbers are particularly pronounced for minorities in STEM, such as Black and Hispanic students [77, 78].

One survey found that the number one factor students cited for not completing an engineering degree was that they were "not performing academically [79]." Another survey found that the top reason students gave for not completing an engineering degree was coursework related, which included comments of classes being too theoretical, too hard, or having too much math [70]. With this in mind, it is perhaps unsurprising that a common explanation for the high attrition rates in computer science is "poor math skills and problem solving abilities [80]." Given that the computing fields tend to be more math and theory intensive than other engineering disciplines, this helps to explain their higher attrition rates.

Teaching mathematical logic can develop students' analytical thinking skills, thus preparing them for the mathematical rigor of their future courses. Furthermore, the authors of this paper suggest teaching logic will particularly help first generation and minority students, who often enter college disproportionately unprepared. One study suggests that the lower average Math SAT scores of minority students [77] may reflect a lower level of quantitative and analytical knowledge [81]. If minority students are disproportionately unprepared for their engineering courses, and unpreparedness leads to attrition, one would expect to observe (as is observed empirically) higher attrition rates for minority students. This is compounded by the fact that many instructors may fail to distinguish between poor talent and poor preparation, prematurely discouraging minority students from continuing in the computing fields [72]. If electrical engineering and computer science departments are committed to improving retention, they should ensure all students have the prerequisite skills expected in their coursework. The authors of this paper argue that logic and mathematical logic will teach many of these required skills, and will be especially beneficial for students unaccustomed to the mathematical sophistication demanded by their major.

VIII. Future Work

Given the importance of logic to the computing fields, it is surprising that no studies have tested whether students who take a course in mathematical logic improves their outcomes in these fields. Future work should thus not only test this general hypothesis, but also focus on testing which topics from mathematical logic improves student success in different courses. Courses with clear relevance to mathematical logic are digital design courses, proof-based mathematics courses such as real analysis and linear algebra, and courses with material from discrete mathematics such as algorithms. Perhaps it is the case that set theory or modal logic is important for understanding probability while deductive logic is more important for building compilers.

One of the challenges to collecting data on this subject is that mathematical logic is often only offered by philosophy departments, and thus may not be listed or count towards a student's credits for graduation. As a result, students may only take such a class if and when their schedule allows, which may be their junior or senior years. Unfortunately, this means they may miss out on being able to use these skills when they could be the most helpful, such as in sophomore and junior-level mathematics and theory courses. Furthermore, relegating logic to a free elective means the students who are more likely to take such a course (or understand its relevance) may be those who are already better prepared, such as those who attended a high school that offered advanced placement (AP) courses to count for college credit. Failing to include logic into the curriculum explicitly may thus inadvertently worsen the gap between better and poorer prepared students.

IX. Conclusion

This paper has argued for fundamental importance of logic in electrical engineering and computer science curricula. Given the history of computing and the skills future engineers are expected to have, the authors believe the question is not *whether* logic should be taught, but rather *how* it should be taught to improve problem solving and design skills. Special attention should thus be paid to task value and the extent to which students see learning logic as relevant to their future goals. For students in the computing fields, demonstrating the relevance of logic to students is made easier given the clear parallels between logic and mathematics, digital design, programming, and computing history. Future work can perform experimental studies on the most effective ways to teach topics from mathematical logic, such as deductive reasoning and set theory, to engineering and computer science students.

References

[1] Aristotle, *The Organon*. H. G. Bohn, 1853.

[2] S. Shapiro and T. Kouri Kissel, "Classical Logic," in *The Stanford Encyclopedia of Philosophy*, spring 2024 ed., E. N. Zalta and U. Nodelman, Eds. Metaphysics Research Lab, Stanford University, 2024. [Online]. Available: https://plato.stanford.edu/archives/spr2024/entries/logic-classical/

[3] J. Barwise, *Handbook of Mathematical Logic*. Elsevier, Mar. 1982.

[4] A. D. Morgan, *On the Syllogism: And Other Logical Writings*. Routledge, Sep. 2019.

[5] G. Boole, *The Mathematical Analysis of Logic*. CreateSpace Independent Publishing Platform, 1847.

[6] J. Corcoran, "Aristotle's Prior Analytics and Boole's Laws of Thought," *History and Philosophy of Logic*, vol. 24, no. 4, pp. 261–288, Dec. 2003, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01445340310001604707. [Online]. Available: https://doi.org/10.1080/01445340310001604707

[7] G. Boole, *The Laws of Thought (1854)*. Open Court Publishing Company, 1911.

[8] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Electrical Engineering*, vol. 57, no. 12, pp. 713–723, Dec. 1938, conference Name: Electrical Engineering. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6431064?casa_token=scCHdlbSI48AAAAA:0of06MtaJoP5LzSJixmOt4p5LWmAXFbUKrPDXcaWgSC0Rrsf6QZsJ64l-hnSGhRVqufJ678W

[9] H. H. Goldstine, *The Computer from Pascal to Von Neumann*. Princeton University Press, 1993.

[10] W. Brinkman, D. Haggan, and W. Troutman, "A history of the invention of the transistor and where it will lead us," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 12, pp. 1858–1865, Dec. 1997, conference Name: IEEE Journal of Solid-State Circuits. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/643644?casa_token=n5rDzECU4c8AAAAA:Iisq_a4VBMztoyl5lUcY-oLkJcldYbZ_nme8NExBYHjBkoy6o6wVJ3MHc7XaGycK9flTBcQH

[11] W. Pietsch, *On the Epistemology of Data Science: Conceptual Tools for a New Inductivism*. Springer Nature, Dec. 2021.

[12] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, 2006, publisher: ACM New York, NY, USA.

[13] E. W. Dijkstra, "On the cruelty of really teaching computing science," *Communications of the ACM*, vol. 32, no. 12, pp. 1398–1404, 1989.

[14] P. J. Denning, "A debate on teaching computing science," *Communications of the ACM*, vol. 32, no. 12, pp. 1397–1414, Dec. 1989. [Online]. Available: https://dl.acm.org/doi/10.1145/76380.76381

[15] C. Selby and J. Woollard, "Computational thinking: the developing definition," 2013, num Pages: 6 Publisher: University of Southampton. [Online]. Available: https://eprints.soton.ac.uk/356481/

[16] P. J. Denning and M. Tedre, *Computational Thinking*. MIT Press, May 2019.

[17] S. Sentance, E. Barendsen, and C. Schulte, *Computer Science Education: Perspectives on Teaching and Learning in School*. Bloomsbury Publishing, Mar. 2018.

[18] D. J. Velleman, *How to Prove It: A Structured Approach*, 3rd ed. Cambridge University Press, Jul. 2019.

[19] G. Strang, *Linear Algebra and Its Applications, Fourth Edition*, 4th ed. Thomson, Brooks/Cole, 2006, google-Books-ID: q9CaAAAACAAJ.

[20] L. Hietalahti, "Is mathematical logic needed in electrical engineering?" *Scientific Bulletin. Series, Mathematical Modeling in Civil Engineering (Nov 2010), p. 12-22*, 2010, publisher: Technical University of Civil Engineering.

[21] S. S. Epp, "The Role of Logic in Teaching Proof," *The American Mathematical Monthly*, vol. 110, no. 10, pp. 886–899, Dec. 2003, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00029890.2003.11920029. [Online]. Available: https://doi.org/10.1080/00029890.2003.11920029

[22] S. Reeves, *Logic for Computer Science*, 1990.

[23] K. Stacey, "What is mathematical thinking and why is it important," *Progress report of the APEC project: collaborative studies on innovations for teaching and learning mathematics in different cultures (II)—Lesson study focusing on mathematical thinking*, 2006.

[24] P. R. Halmos, "The Heart of Mathematics," *The American Mathematical Monthly*, Aug. 1980. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00029890.1980.11995081

[25] A. A. Anwar and D. J. Richards, "Comparison of EC and ABET accreditation criteria," *Journal of Professional Issues in Engineering Education and Practice*, vol. 144, no. 3, p. 06018001, 2018, publisher: American Society of Civil Engineers.

[26] F. Berry, P. DiPiazza, and S. Sauer, "The future of electrical and computer engineering education," *IEEE Transactions on Education*, vol. 46, no. 4, pp. 467–476, Nov. 2003, conference Name: IEEE Transactions on Education. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1245170?casa_token=67px9pp2hgUAAAAA:N8UoYXWelbwxAaEBIxcdGevQtGiRzg7BGJGwe7C_jI62E9I4Attq-vqXq2fa45_ORWM7mbM4EQ

[27] A. Church, "A Bibliography of Symbolic Logic," *The Journal of Symbolic Logic*, vol. 1, no. 4, pp. 121–216, Dec. 1936. [Online]. Available: https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/abs/bibliography-of-symbolic-logic/05810ED72F2FE5CF9CA3CC3AF3C532AE

[28] ——, *Introduction to Mathematical Logic*, reprint edition ed. Princeton, NJ Chichester, West Sussex: Princeton University Press, Oct. 1996.

[29] A. Turing, "Systems of Logic Based on Ordinals (1938)," in *The Essential Turing*, B. J. Copeland, Ed. Oxford University Press, Sep. 2004, p. 0. [Online]. Available: https://doi.org/10.1093/oso/9780198250791.003.0007

[30] A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. s2-42, no. 1, pp. 230–265, 1937, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s2-42.1.230. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1112/plms/s2-42.1.230

[31] K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik*, vol. 38, no. 1, pp. 173–198, Dec. 1931. [Online]. Available: https://doi.org/10.1007/BF01700692

[32] B. J. Copeland and Z. Fan, "Did Turing Stand on Gödel's Shoulders?" *The Mathematical Intelligencer*, vol. 44, no. 4, pp. 308–319, Dec. 2022. [Online]. Available: https://doi.org/10.1007/s00283-022-10177-y

[33] S. C. Kleene, "A Theory of Positive Integers in Formal Logic. Part I," *American Journal of Mathematics*, vol. 57, no. 1, pp. 153–173, 1935, publisher: The Johns Hopkins University Press. [Online]. Available: https://www.jstor.org/stable/2372027

[34] ——, "Representation of Events in Nerve Nets and Finite Automata," in *Representation of Events in Nerve Nets and Finite Automata*. Princeton University Press, Mar. 2016, pp. 3–42. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/9781400882618-002/pdf?licenseType=restricted

[35] J. Von Neumann, "First draft of a report on the EDVAC," *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27–75, 1993, conference Name: IEEE Annals of the History of Computing. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/238389?casa_token=lYnkYwx3kBMAAAAA:5bushafZyuOTxkX-WE2_iHDa2s0DE8gpGDciz14JUR1Io9fWY-DTI2fdgyshmD5vOt5D9FrX

[36] ——, "Zur einführung der transfiniten zahlen," *Acta Litterarum ac Scientiarum Regiae Universitatis Hungaricae Francisco-Josephinae, sectio scientiarum mathematicarum*, vol. 1, pp. 199–208, 1923.

[37] ——, "Eine Axiomatisierung der Mengenlehre." vol. 1925, no. 154, pp. 219–240, Jan. 1925, publisher: De Gruyter Section: Journal für die reine und angewandte Mathematik. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/crll.1925.154.219/html

[38] F. Dyson, "A walk through Johnny von Neumann's garden," *Notices of the AMs*, vol. 60, no. 2, 2013.

[39] T. Connolly and C. Begg, *Database Systems: A Practical Approach to Design, Implementation, and Management*, 6th ed. Boston: Pearson, Jan. 2014.

[40] O. Owe, S. Krogdahl, and T. Lyche, "A Biography

of Ole-Johan Dahl," in *From Object-Orientation to Formal Methods: Essays in Memory of Ole-Johan Dahl*, O. Owe, S. Krogdahl, and T. Lyche, Eds. Berlin, Heidelberg: Springer, 2004, pp. 1–7. [Online]. Available: https://doi.org/10.1007/978-3-540-39993-3_1

[41] S. A. Cook, "The Complexity of Theorem-Proving Procedures (1971)," in *Ideas That Created the Future*, H. R. Lewis, Ed. The MIT Press, Feb. 2021, pp. 333–338. [Online]. Available: https://direct.mit.edu/books/book/5003/chapter/2657057/The-Complexity-of-Theorem-Proving-Procedures-1971

[42] L. A. Levin, "Universal sequential search problems," *Problemy peredachi informatsii*, vol. 9, no. 3, pp. 115–116, 1973, publisher: Russian Academy of Sciences, Branch of Informatics, Computer Equipment and ....

[43] S. A. Cook, "Feasibly constructive proofs and the propositional calculus (Preliminary Version)," in *Proceedings of the seventh annual ACM symposium on Theory of computing*, ser. STOC '75. New York, NY, USA: Association for Computing Machinery, May 1975, pp. 83–97. [Online]. Available: https://doi.org/10.1145/800116.803756

[44] S. Cook and P. Nguyen, *Logical Foundations of Proof Complexity*, 1st ed. Ithaca, NY : New York, NY: Cambridge University Press, Jan. 2010.

[45] O. Kayhan, O. Korkmaz, and R. Cakir, "How Do Computational Thinking and Logical and Math Thinking Skills Predict Programming Self-Efficacy?" *Computers in the Schools*, vol. 0, no. 0, pp. 1–23, 2023, publisher: Routledge _eprint: https://doi.org/10.1080/07380569.2023.2220696. [Online]. Available: https://doi.org/10.1080/07380569.2023.2220696

[46] K. Daungcharone and K. Thongkoo, "The Effects of Thinking Process Model Technique on Logical Thinking Skills Influencing Programming Achievement," in *2022 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*, Jan. 2022, pp. 134–138, iSSN: 2768-4644. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9720360?casa_token=FIKxhs4jbzUAAAAA:1qOIgXe-9vDGVaW_Kstg5zUKBMk5weSJQHyyfp9d7HQbQ4iHnLzl8cBmSUjYDc5TbigiHbYQsg

[47] P. C. Wason, "Reasoning about a Rule," *Quarterly Journal of Experimental Psychology*, vol. 20, no. 3, pp. 273–281, Aug. 1968, publisher: SAGE Publications. [Online]. Available: https://doi.org/10.1080/14640746808400161

[48] P. W. Cheng, K. J. Holyoak, R. E. Nisbett, and L. M. Oliver, "Pragmatic versus syntactic approaches to training deductive reasoning," *Cognitive Psychology*, vol. 18, no. 3, pp. 293–328, Jul. 1986. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0010028586900022

[49] D. Sperber, F. Cara, and V. Girotto, "Relevance theory explains the selection task," *Cognition*, vol. 57, no. 1, pp. 31–95, Oct. 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/001002779500666M

[50] M. Bong, "Role of Self-Efficacy and Task-Value in Predicting College Students' Course Performance and Future Enrollment Intentions," *Contemporary Educational Psychology*, vol. 26, no. 4, pp. 553–570, Oct. 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0361476X00910488

[51] N. Attridge, A. Aberdein, and M. Inglis, "Does studying logic improve logical reasoning?" Jan. 2016, publisher: Loughborough University. [Online]. Available: https://repository.lboro.ac.uk/articles/conference_contribution/Does_studying_logic_improve_logical_reasoning_/9373463/1

[52] H. Bronkhorst, G. Roorda, C. Suhre, and M. Goedhart, "Students' use of formalisations for improved logical reasoning," *Research in Mathematics Education*, vol. 24, no. 3, pp. 291–323, Sep. 2022, publisher: Routledge _eprint: https://doi.org/10.1080/14794802.2021.1991463. [Online]. Available: https://doi.org/10.1080/14794802.2021.1991463

[53] N. Alaraje and A. Sergeyev, "Developing a Digital Logic Curriculum in Electrical Engineering Technology: Bridging the Gap Between Industry and Academia," *TRY ACAMEDICS!*, p. 47, 2010.

[54] N. Calazans and F. Moraes, "Integrating the teaching of computer organization and architecture with digital hardware design early in undergraduate courses," *IEEE Transactions on Education*, vol. 44, no. 2, pp. 109–119, May 2001, conference Name: IEEE Transactions on Education. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/925805?casa_token=Q1jkK91XTEgAAAAA:WNrXqpSE1VzIJ1HtoxyNBfQiE7JuSxY9NM8YDBGXgf9npf9sSCMENIWALwE1sXxHlGn07B-JOg

[55] D. Harris and S. Harris, *Digital Design and Computer Architecture*. Elsevier, Aug. 2012, google-Books-ID: ZyrjlPlpoFEC.

[56] R. R. Karfhage, "Logic for the computer sciences," *Communications of the ACM*, vol. 7, no. 4, pp. 216–218, 1964, publisher: ACM New York, NY, USA.

[57] L. Carley, P. Khosla, and R. Unetich, "Teaching "Introduction to electrical and computer engineering" in context," *Proceedings of the IEEE*, vol. 88, no. 1, pp. 8–22, Jan. 2000, conference Name: Proceedings of the IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/811595

[58] S. Baase, *A Gift of Fire: Social, Legal, and Ethical Issues for Computing Technology*, 4th ed. Upper Saddle River, NJ: Pearson, Aug. 2012.

[59] L. S. Vailshery, "Most used languages among software developers globally 2023," Apr. 2024. [Online]. Available: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

[60] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *OpenAI*, 2018, publisher: OpenAI.

[61] W. H. Holliday, M. Mandelkern, and C. E. Zhang,

"Conditional and Modal Reasoning in Large Language Models," Jul. 2024, arXiv:2401.17169 [cs]. [Online]. Available: http://arxiv.org/abs/2401.17169

[62] Z. Liu, Y. Tang, X. Luo, Y. Zhou, and L. F. Zhang, "No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT," *IEEE Transactions on Software Engineering*, vol. 50, no. 6, pp. 1548–1584, Jun. 2024, conference Name: IEEE Transactions on Software Engineering. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10507163?casa_token=lGb7_-wZHk4AAAAA:DWMe2YkObthvd3ziKz2Wa6MIneDJqujrha9xwRo1iG4Tbo4cXcQLdSAIsW_wvn02Bi7HRQ0k

[63] A. Freedman and N. Artemeva, "Learning to Teach Writing to Engineers," *Discourse and Writing/Rédactologie*, vol. 14, no. 1, pp. 20–Jan, Apr. 1998, number: 1. [Online]. Available: https://journals.sfu.ca/dwr/index.php/dwr/article/view/420

[64] D. F. Beer and D. A. McMurrey, *A Guide to Writing as an Engineer*. John Wiley & Sons, Apr. 2019.

[65] R. Boyd, "Teaching Writing with Logic," *College Teaching*, Apr. 1995. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/87567555.1995.9925514

[66] B. McKenna, "How Engineers Write: An Empirical Study of Engineering Report Writing," *Applied Linguistics*, vol. 18, no. 2, pp. 189–211, Jun. 1997. [Online]. Available: https://doi.org/10.1093/applin/18.2.189

[67] K. Rosen, *Discrete Mathematics and Its Applications Seventh Edition*, 7th ed. New York: McGraw Hill, Jun. 2011.

[68] Y. N. Patt and S. J. Patel, *Introduction to Computing Systems: From bits & gates to C & beyond*, 2nd ed. Boston: McGraw Hill, Aug. 2003.

[69] S. M. Barnett and S. J. Ceci, "When and where do we apply what we learn?: A taxonomy for far transfer." *Psychological Bulletin*, vol. 128, no. 4, pp. 612–637, 2002. [Online]. Available: https://doi.apa.org/doi/10.1037/0033-2909.128.4.612

[70] C. Baillie and G. Fitzgerald, "Motivation and attrition in engineering students," *European Journal of Engineering Education*, vol. 25, no. 2, pp. 145–155, Jun. 2000, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/030437900308544. [Online]. Available: https://doi.org/10.1080/030437900308544

[71] L. R. Horn, "Contradiction," in *The Stanford Encyclopedia of Philosophy*, winter 2018 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2018. [Online]. Available: https://plato.stanford.edu/archives/win2018/entries/contradiction/

[72] R. Tapia, *Losing the Precious Few: How America Fails to Educate its Minorities in Science and Engineering*. Arte Publico Press, Apr. 2022.

[73] S. M. Lord, R. A. Layton, and M. W. Ohland, "A disciplinary comparison of trajectories of U.S.A. engineering students," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, Oct. 2014, pp. 1–4, iSSN: 2377-634X. [Online]. Available: https://ieeexplore.ieee.org/document/7044414

[74] S. M. Lord, R. A. Layton, M. W. Ohland, and M. K. Orr, "Student demographics and outcomes in Electrical and Mechanical Engineering," in *2013 IEEE Frontiers in Education Conference (FIE)*, Oct. 2013, pp. 57–63, iSSN: 2377-634X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6684788?casa_token=8UbJGaq6bKwAAAAA:KioLKfmw5xPsZ71TD2yo8z50E6oSR5vRfmEfRwNM7GZxcnRfs6tK7m8kC85n1ZSUHIaVLjVx

[75] M. W. Ohland, S. D. Sheppard, G. Lichtenstein, O. Eris, D. Chachra, and R. A. Layton, "Persistence, Engagement, and Migration in Engineering Programs," *Journal of Engineering Education*, vol. 97, no. 3, pp. 259–278, 2008, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2168-9830.2008.tb00978.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2168-9830.2008.tb00978.x

[76] M. N. Giannakos, I. O. Pappas, L. Jaccheri, and D. G. Sampson, "Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness," *Education and Information Technologies*, vol. 22, no. 5, pp. 2365–2382, Sep. 2017. [Online]. Available: https://doi.org/10.1007/s10639-016-9538-1

[77] M. Baird, M. Buchinsky, and V. Sovero, "Decomposing the Racial Gap in STEM Major Attrition: A Course-Level Investigation," Rochester, NY, Oct. 2016. [Online]. Available: https://papers.ssrn.com/abstract=2870049

[78] K. M. Whitcomb and C. Singh, "Underrepresented minority students receive lower grades and have higher rates of attrition across STEM disciplines: A sign of inequity?" *International Journal of Science Education*, vol. 43, no. 7, pp. 1054–1089, May 2021, publisher: Routledge _eprint: https://doi.org/10.1080/09500693.2021.1900623. [Online]. Available: https://doi.org/10.1080/09500693.2021.1900623

[79] E. Godfrey, T. Aubrey, and R. King, "Who leaves and who stays? Retention and attrition in engineering education," *Engineering Education*, vol. 5, no. 2, pp. 26–40, Dec. 2010, publisher: Routledge _eprint: https://doi.org/10.11120/ened.2010.05020026. [Online]. Available: https://doi.org/10.11120/ened.2010.05020026

[80] T. Beaubouef and J. Mason, "Why the high attrition rate for computer science students: some thoughts and observations," *SIGCSE Bull.*, vol. 37, no. 2, pp. 103–106, Jun. 2005. [Online]. Available: https://doi.org/10.1145/1083431.1083474

[81] C. P. Veenstra, E. L. Dey, and G. D. Herrin, "A Model for Freshman Engineering Retention," *Advances in Engineering Education*, vol. 1, no. 3, 2009, publisher: American Society for Engineering Education ERIC Number: EJ1076050. [Online]. Available: https://eric.ed.gov/?id=EJ1076050

Paul Mayer Paul Mayer is a PhD student in Electrical and Computer Engineering at Rice University. He got is B.S.E.E. from Rice University in 2018. His research focuses on applying signal processing knowledge to machine learning tasks. In addition to research, Paul enjoys DJing and producing music.