What Can Computer Science Educators Learn From the Failures of Top-Down Pedagogy?

Sverrir Thorgeirsson
sverrir.thorgeirsson@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

Tracy Ewen
tracy.ewen@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

Zhendong Su
zhendong.su@inf.ethz.ch
ETH Zurich
Zurich, Switzerland

ABSTRACT

While educational researchers in various disciplines are grappling with how to develop policies and pedagogical approaches that address the use of generative artificial intelligence, the challenge is especially complex in computer science education where the new technology is changing the core of the field. In this paper, we take a look at the pedagogy of other subjects with a longer history than computer science and a more extensive body of educational research to collect insights on how this challenge can be met. We begin by drawing from recent neurological research to find domains that share cognitive commonalities with computer programming and then build upon comparisons that others have made to literacy and mathematics education. We then consider how the "reading wars" and "math wars" have shaped these fields, which we see as conflicts between less effective top-down pedagogy and more effective bottom-up pedagogy, and reflect on what would be comparable approaches in teaching computing. We find that approaches that make heavy use of large language models without teaching fundamentals can be compared to the top-down pedagogy of reading and mathematics and are likely to be ineffective on their own. Therefore, we advise against the exclusive use of such approaches with novices. However, we also acknowledge that the social science surrounding computer science education is complex and that effectiveness only tells a part of the story, with other factors such as engagement, motivation and social dynamics also being important.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

computer science education, literacy, reading, literacy, reading wars, math wars, phonics, whole language, top-down pedagogy, bottom-up pedagogy, large language models, generative artificial intelligence, position paper

1 MOTIVATION

Computing education research faces well-known challenges that are either unique or amplified when compared to other fields. First, as a formal academic discipline, computer science (CS) has a short history compared to long-established educational subjects, meaning that CS educators have a more shallow pedagogical well to draw from. Second, while rapid advancements in technology such as artificial intelligence (AI) are affecting the entire educational system, CS education is directly impacted, requiring curricula to constantly adapt in a way that is less urgent in more static disciplines. To make sense of these changes and to help us evaluate new CS-specific pedagogical innovations, CS educators would benefit greatly by borrowing insights from disciplines with hundreds of years of study and research, for example mathematics and language education.

As a starting point for any comparison, consider the neural mechanisms involved when solving tasks in CS and other domains. A 2014 study [35] using functional magnetic resonance imaging (fMRI) found that brain regions related to language processing are distinctly activated during programming tasks. The authors anticipated this finding would significantly impact programming pedagogy [34]. Some authors have used these results to support the claim that programming should be taught in the same way as natural languages [27, 30]. For example, in a 2018 *ACM Inroads* essay [30], Portnoff argues that the 2014 fMRI study supports the conclusion that computer programming should be taught in the same way as natural languages, arguing that "knowing fluent programmers process programming languages like natural languages, pedagogies can be crafted to attempt to facilitate this physiological outcome," and therefore, the principles of second language acquisition (SLA) pedagogy could be used to inform programming instruction. On the other end of the spectrum, two fMRI studies published in 2020 [12, 19] found that computer code comprehension shares more neural resources with the system responsible for formal logical inference [18]. If corroborated by further studies, one could argue that computer science education should therefore borrow more heavily from mathematics education and that existing efforts to do so (for example by adopting Hazzan's model of abstraction in algebra [9] to the CS pedagogical framework called PGK hierarchy [29]) are robust and well-supported by neuroscience research.

In this position paper, we will not take a stance on which subjects most resemble CS from a neuroeducational perspective. However, the research we mentioned does underscore that it is not a cognitive activity that exists in a vacuum and that there are certainly some shared neural processes involved when students learn to code, read, or solve algebra. Consequently, we find that CS educators may benefit from reviewing ongoing debates about how reading and mathematics should be taught. For one, the scale and scope of some empirical studies that have been conducted within these subjects, for example *Project Follow Through* [15, 37], vastly exceed those of any experiments conducted in computer science. This means that even though there are significant differences between language learning and mathematics on one hand and CS on the other, what makes them relevant for CS educators is the strength of the evidence and knowledge that informs their pedagogical practices.

First, we will discuss "the reading wars", a conflict in literacy education that contrasts *phonics* (a more effective, bottom-up, fundamentals-first approach) with *whole language* (a less effective, top-down, exposure-driven approach). We consider the related pedagogy and

what it means for computer science. We will also discuss the less-known "math wars", a conflict that also contrasts top-down and bottom-up pedagogical approaches but whose resolution has been more ambiguous. We will consider the implications of these debates when it comes to the use of large language models (LLMs) in teaching programming. We contend that approaches that make heavy use of LLMs without teaching fundamentals can be compared to the top-down pedagogy of teaching reading and mathematics, and drawing on extensive research from these fields, we suggest it is likely ineffective as a primary or stand-alone approach in teaching programming. Finally, we discuss the social science surrounding computer science education, highlighting that effectiveness only tells a part of the story, with engagement, motivation and social dynamics also being important factors for learning and introducing technologies into the classroom. We also offer some insight and further discussion on how LLMs might be effectively adopted in a more balanced and informed way into the classroom, with consideration of both bottom-up and top-down approaches.

2 PEDAGOGICAL CONFLICTS

In this section, we offer a brief overview of two debates in education, one in literacy and the other in mathematics, both which can be broadly understood as conflicts between top-down and bottom-up pedagogical approaches. We will then discuss their possible implications for computing education.

2.1 The Reading Wars

The "reading wars" refers to a debate over the most effective methods for teaching children how to read [1, 28]. Occurring primarily in English-speaking countries in the 1980s and 1990s, this conflict, which is between two binary oppositional models [36] called *phonics* and *whole language*, has influenced educational policies, teaching practices, and curricular design for a long time. Soler [36] explains that researchers and practitioners have linked the two approaches to bottom-up and top-down pedagogical approaches, respectively. Phonics, as a pedagogical approach, is characterized by a bottom-up methodology, beginning with the most elementary components of language, namely letters and their corresponding sounds. Students using this method learn to decode words by sounding them out, which involves blending the sounds of letters and letter combinations to form words. This approach is systematic and structured, following a sequential progression from simple to more complex sound patterns, thereby building a foundation that gradually enables students to understand unfamiliar words independently.

In contrast, whole language is a top-down approach under which students are engaged with complete texts from the very beginning. This method emphasizes what has been called "reading for meaning," trying to encourage students to use their prior knowledge of spoken language and contextual clues within the text to interpret and understand what they read, for example by recognizing words by their shape. Proponents of whole language believe that language learning is a natural process, mirroring the way children learn to speak, and that exposure to text can help them learn reading without as much structure or scaffolding as phonics makes use of. This approach often uses techniques like shared reading of simple, engaging texts like storybooks, aiming to make learning to read an enjoyable activity and emphasizing creativity.

Both approaches have had passionate adherents and some of the debate has transcended the subject matter; DeJulio et al. [4] claim the conflict "has been used as a proxy to further users' political agenda," which may reflect broader societal tensions about authority, control, and the purpose of education. Some view the phonics approach as aligned with traditional, authoritative educational practices that prioritize measurable outcomes and standardized testing, emphasizing rigor and discipline. In contrast, whole language is often associated with more progressive educational philosophies that value student-centered learning and holistic development.

Today, the broad scope of scientific evidence indicates that the conflict should be largely settled as far as effectiveness in concerned; in the words of Australian politician Sarah Mitchell, "the reading wars are over - and phonics has won" [21]. This view is supported by the wealth of empirical studies that indicate that whole language is ineffective at teaching reading [1, 20]. However, while the body of research is quite clear, there still remains ambiguity on how the reading wars are understood and applied in practice [4]. Although the ultimate goal of educational research is the improvement of educational practice through policies, historically, research has been regarded as only one among many information sources consulted in educational policy formation [28]. Pearson [28] attributes the downfall of whole language to seven factors, which culminated (in the US) in the reading first component of the No Child Left Behind Act of 2001 [24], including increasing politicization of reading research and questionable applications of the method, as well as pressure for educators to produce better and measurable results in schools. However, and without any doubt, the most pertinent reason is the number of empirical studies showing that whole language is ineffective at teaching reading [1, 22]. In other words, its central claim, namely that students could pick up reading simply through exposure much like children learn to speak, is not supported by scientific evidence. In a position paper from 2000, Moats [23] puts it concisely: "Learning to read is not a natural process. Most children must be taught to read through a structured and protracted process in which they are made aware of sounds and the symbols that represent them, and then learn to apply these skills automatically and attend to meaning."

2.2 The Math Wars

Similar to the reading wars, the "math wars" of the late 20th century were a significant conflict in mathematics education that occurred primarily in the United States [32]. The "war" revolved around competing philosophies and methodologies for teaching mathematics, pitting traditional methods against the reform-based methods advocated by a professional association of schoolteachers [25]. Reformists, influenced by constructivist learning theories, believed that an earlier return to basics in mathematics was wrong-headed, finding that "an exclusive focus on basics leaves students without the understandings that enable them to use mathematics," and that instead, educators should focus on problem solving [32]. While reform-opponents did laud it for replacing rote learning with realistic applications that are relevant for students, they criticized

it for a lack of rigor, and that in neglecting basic skills and algorithmic proficiency, students would be ill-prepared for higher-level mathematics [39]. Meanwhile, supporters of the reform contended that traditional methods failed to develop true mathematical understanding and left many students disengaged from the subject [32].

Like the reading wars, a part of the conflict can be described as one between top-down and bottom-up pedagogy; traditionalists favored an approach that begins with the fundamentals, aiming to build proficiency in rote skills before moving up onto more advanced topics, while reformists emphasized the use of authentic, real-world applications from the start, teaching fundamentals along the way. However, unlike the reading wars, where a clear consensus emerged favoring phonics-based instruction, the math wars have not reached a definitive resolution. Instead, many educators and researchers advocate for a balanced approach that combines elements of both traditional and reform methods [14]. This perspective recognizes the importance of both procedural fluency and conceptual understanding, aiming to develop students who are computationally skilled and mathematically proficient problem-solvers.

2.3 Implications

If we think that learning to read prose and code are similar activities that can borrow the same pedagogy, the reading wars have some surface-level lessons for CS educators. For one, a part of the conflict has to do with the unnecessary complexities of the English language. It has been found that children learn reading more easily in languages with more transparent spelling (part of the "mechanics" of a language) [5], which has been conjectured to explain the success of Finnish students in reading [41]. Therefore, programming language designers may wish to consider adopting simple "orthographies," meaning straightforward syntax and semantics with minimal irregularities and exceptions in language rules. For example, perhaps it is undesirable that same symbols can represent multiple distinct concepts, such as how an asterisk in C can be used for both pointer declaration and multiplication, or how the plus symbol can be used in JavaScript for addition, concatenation, or type conversion.

However, what we find more important about both the reading wars and the math wars is what they can tell us broadly about the appropriate use of pedagogy. In a 2009 blog post, Mark Guzdial wondered what would be the equivalent of phonics in CS education [8], speculating that syntax, reading code and tinkering with code are the mechanics that we would teach under a phonics-equivalent model in computing education. In our view, the question has a clearer answer in 2024 than it did in 2009: the classical, bottoms-up pedagogy in computer science, with an emphasis on fundamentals like data structures and algorithms, can be compared to phonics. Here, we suggest Guzdial's "mechanics" of learning syntax might be closer to learning the alphabet as a first step of a bottom-up approach, followed by phonics, or learning the sounds and language patterns, which might be closer to learning data structures and algorithms. Regardless, we posit that approaches that rely heavily on large language models (LLMs) or "prompt programming" run the risk of being the computer science equivalent of whole language, focusing heavily on end results without understanding the underlying mechanics.

In a recent empirical study on GitHub Copilot [3], a software development tool powered by an LLM, the authors assessed how well Copilot solved fundamental algorithmic problems (including sorting, binary search trees, graph and greedy algorithms) and then compared the output to human solutions. They found that Copilot was able to generate correct code for most algorithms, however the quality of the solutions depended strongly on well defined and concise prompts given by an experienced programmer, and even then, in some cases the generated code needed to be modified (by the programmer) to generate correct solutions. Overall, the study concluded that while Copilot was of great help to expert users, "it can also become a liability if it is used by novices, who may not be familiar with the problem context and correct coding methods" [3], inadvertently encouraging a superficial grasp of coding principles. In this case, the term, "novices" refers to junior developers, so those who are already quite advanced in programming.

Similar research within CS education appears to have arrived at the same conclusion; a study exploring 21 novice programmers' use of generative AI tools showed that while these tools helped students with programming tasks, they also exacerbated or introduced new metacognitive difficulties, such as causing students to form wrong conceptual models about the problems and being unable to resolve them [31]. Critically, students who struggled often relied too heavily on the AI's suggestions, leading to an illusion of competence without a real understanding of the problem-solving process. We think that the results of this study bear a striking resemblance to the reading wars; just like whole language approaches have been critiqued for not only causing students to develop superficial reading skills, but also a flawed understanding of their own competence, and forming inaccurate mental models, reliance on AI for code generation without understanding the underlying principles could contribute not only to similarly superficial programming skills, but also give learners a false sense of mastery, with students struggling with problem solving, and figuring out the specific rules of the (programming) language on their own.

Another recent study [26], in an intermediate-level programming course with 158 participants, looked at how social perceptions influenced the decisions of students to use LLMs and impact on student self-perception and learning outcomes. In terms of learning outcomes, the study found that early use of LLMs by the students correlated with lower self-efficacy (and lower exam scores), while their perceived over-reliance on LLMs, correlated with decreased self-efficacy later in the course. Students generally had positive perceptions of their experiences with LLMs when using them to explain concepts or error messages (when they purposefully used the LLMs as a learning tool), however, when using it for learning new concepts and generating code, many students found using LLMs to be less intuitive or effective. Students who were more advanced in programming, were more self-aware of using LLMs as a tool for learning, and noted that their foundations (without LLMs) were helpful when they later used LLMs for their work.

Finally, to strengthen our argument, we note that apart from mathematics and language, similar discussions about top-down pedagogy have been found in other learning domains; for instance, *discovery* or *inquiry learning*, which has been called "a prominent strand to the top-down movement" [6], has been widely applied in mathematics and science education, and yet been strongly criticized

Figure 1: Bottom-up methods (left) contrasted against top-down methods (right) for teaching reading (top) and computer programming (bottom). The figure is inspired by Anne-Margaret Smith's figure from *Teaching Basic Literacy Skills* workshop [11]. Balanced approaches would involve using both top-down and bottom-up pedagogy simultaneously such that they complement each other.

for lack of empirical support [17, 38]. However, learning scientists are not all in agreement on this point, and some may criticize such arguments on the grounds that inquiry or problem-based learning strategies are distinct from minimally-guided constructivist instruction strategies [10].

3 THE SOCIAL SCIENCE IS LESS CLEAR

We conclude this paper by considering the societal aspects of education and how they may affect our argument. An overlooked aspect of pedagogical methods and interventions is that to be successful, they must appeal to teachers, students, parents, and even the wider community. While effectiveness is important, so is the motivation, engagement, and other nuanced factors, such as social dynamics and possibly even curriculum policies, of those involved. Teachers are more likely to embrace methods that they find enjoyable and manageable, which in turn enhances their commitment and effectiveness in the classroom. Similarly, students are more likely to continue with a subject when the learning process is engaging, resonates with their experiences and interests, and gives them the sense that they are making progress in their learning.

Consequentially, although traditional approaches may prove effective, they may not always be the best way to appeal to a large group of students and to keep them engaged. In the context of reading education, the pervasiveness of the whole language pedagogy, in spite of its demonstrated lack of effectiveness, has been attributed to that teachers simply do not enjoy doing phonics drills.

In the opinion piece "The social science of reading isn't so clear," Matthew Yglesias suggests that teachers may find them "tedious or even stultifying" [41]. Instead, attempting to read books as early as possible is much more interesting for both teachers and students, since even when students are unprepared, it is a much more meaningful activity than sounding out letters. Yglesias suggests that if the teachers are motivated, their performance will be better, even if the approach that they are following is not effective.

This point might be even more relevant in computer science education, where keeping students engaged appears to be particularly important in light of the subject's high dropout and failure rates. As the use of AI in software development becomes increasingly common, making it simpler than ever to implement creative ideas and make meaningful, authentic applications with little effort, it becomes more difficult to ask teachers and students to keep AI out of the classroom. Even if research studies indicate that novice students learn less when using LLMs or gain a distorted sense of their competence, if doing so keeps them motivated to continue, due to strong positive social motivation [26] or for other reasons, the point may be moot.

In literacy education, most approaches today do not focus exclusively on phonics; according to some, there is an increasingly common view that phonics and whole language have "a different but potentially complementary role to play in the effective teaching of reading" [2], and teachers often try to incorporate a blend of phonics instruction with exposure to rich and engaging texts. Similarly, in response to the "math wars," many educators have attempted to blend the two conflicting positions such that they complement each other, finding that "math instruction cannot be effective if it is based on extreme positions" and that focus on narrow approaches can explain poor educational outcomes in mathematics [14].

Figure 1 shows a diagram representing bottom-up (left) and top-down (right) pedagogical approaches for teaching literacy (top) and computer science (bottom). Each approach, represented as a triangle, can conceivably be taught in isolation or combined with each other. Similar to how reformists have proposed that mathematics education can begin with solving real-world, authentic tasks that are meaningful for students, top-down methodology for teaching computer science would also start with implementing real-world tasks before students have had a chance to learn computing fundamentals. Until the advent of powerful AI tools such LLMs, we believe that this pedagogical approach has been unfeasible or too difficult to implement in practice. Given the failures in other subjects of applying top-down pedagogical methodology in isolation, we think that CS educators should also avoid this. However, a balanced approach might involve combining foundational learning with the practical application of LLMs, thereby following two "triangles" at the same time. By integrating core programming concepts with opportunities to use the same state-of-the-art technology as professionals, students can develop a more robust understanding of computer science principles while also gaining practical experience that promotes higher engagement and student motivation.

In a 2024 paper, Kirova [16] also advocates for a holistic approach to incorporating LLMs and generative AI into course curricula, suggesting educators need to update pedagogical methods in order to make effective use of tools such as LLMs, emphasizing an adapting curriculum that should combine technical skills, ethical awareness, and adaptable and evolving learning strategies to stay up-to-date with the changing AI landscape. The authors use examples from their own course development and include suggestions for topic ideas, including design, the use of patterns, coding practices (e.g. writing clean code, code consistency, useful commenting). They add that time should be spent discussing the capabilities and limitations of LLMs and how to use them effectively, and how to interact and interface with them. Students should also have time to review, test and validate solutions in order to better understand incorrect solution or hallucinations from LLMs, thus also developing critical and reflective thinking skills. Although their suggestions and reflections come from their experience in an upper-level course, these are many important lessons that have already been explored in a teaching setting and can be incorporated into other levels.

A second point is that teachers will need to be ready and willing to embrace the use of LLMs in their classroom, and as we mention above, they must be motivated to do so. Yan et al. [40] discuss the practical and ethical challenges of LLMs in education and point out that in general, in order for new technologies to have actual benefits in supporting pedagogical goals, technologies should be convenient to access, support pedagogical beliefs, be able to adapt to curriculum, and be compatible with the level of knowledge and skills of the instructor. As previously discussed, although LLMs may provide a plausible set of solutions for a particular action, it may well also include unfeasible or incorrect code [3, 26, 31]. Grande et al. [7] discuss student perspectives on using LLMs (at the master's level), and conclude with some apt advice: "from an educators' perspective, there is a need for more instruction for students using LLMs: some students did not perceive the tools as such but rather as an authoritative knowledge base, which has implications for educators considering the use of LLMs as discussion partners or tools to practice critical thinking skills." For educators, even at the introductory level, it is thus important to instruct students appropriately, and carefully consider the implications within the curriculum design.

When it comes to integrating new technology in the classroom, there is more to consider than only effectiveness. The social dynamics will also need to be considered such that a balanced approach can be implemented. The 2024 study we mentioned by Padiyath et al. [26] discusses the use of LLMs in programming education in terms of social and cultural dimensions of LLM adoption in the classroom. The authors suggest that the capabilities of the technology alone should not guide or determine the adoption of use since this would overlook important aspects such as the preferences and goals of both students and instructors within the broader educational context. These aspects, although nuanced, should help to shape decisions, as societal groups ultimately help shape the ways in which technologies are used and adopted, and in turn, technologies themselves can influence the behaviors of groups. In trying to understand how social perceptions influenced the decisions of students to use LLMs and impact on student self-perception and learning outcomes, their study found that students' use of LLMs was significantly influenced by their expectations of their future career as well as perceptions of their peer usage of LLMs. Early adoption of LLMs by the students correlated with lower self-efficacy (and lower mid-term exam scores), while their perceived over-reliance on LLMs correlated with decreased self-efficacy later in the course.

These results highlight the complex dynamics between technology and social factors, arguing that these factors need to be considered with the integration and use of such technologies in CS education. Such studies are needed to provide quantitative measures for social, and other more nuanced dimensions, which can be more readily used, or at least considered, in the planning and decision making process by CS educators.

Although we do not offer concrete recommendations for how LLMs should be implemented in CS education, we can draw from other fields where we find strong similarities and offer guidance based on their established pedagogy, combined with the emerging studies on LLM use in CS education. Drawing parallels from these fields, we strongly recommend that computer science educators avoid the recurrent adoption of top-down pedagogy as seen in other fields, even though we now have the tools to support this, and instead suggest a balanced approach. This may mean involving foundational learning combined with the practical application of AI tools like LLMs to develop more robust understanding of computer science principles, while also gaining practical experience that keeps students engaged and motivated, with the caution of introducing AI tools gradually, correctly, and at the appropriate stage.

For guidance on the introduction of LLMs into curriculum, we can draw from the learning theories on the development of reading. In moving from learning letters of the alphabet, to phonics and matching sounds, somewhere along the process the learner transitions into a skilled reader. An influential theory on this transition is the *self-teaching hypothesis* [13, 33], which has alphabetic decoding at its core and helps to model how children make an effective transition from learning bottom-up concepts to becoming a skilled practitioner. Through independent reading of engaging short stories and texts using homophones, children are able to use a combination of alphabetic decoding and repeated exposure to form substantial learning about the orthography and meaning of words. This model provides a powerful paradigm for understanding how children move from novice to expert, using a combination of fundamental concepts and tools, including phonics, along the path of learning to become a skilled reader. This is a lesson on when students should be using generative AI in their learning; if it is to be used as a tool to support and solidify their transition from novice to expert, it is important it is not introduced too early, and that foundational concepts and mechanics, such as syntax, data structures and algorithms are learned first. AI tools might then be introduced to complement and reinforce these concepts, and with a proper introduction by instructors for its correct and complementary use. It should be integrated into curriculum at the point which will help to solidify their learning, moving from novice programmers to experts, and set students up for success in critically using LLMs in their future work.

Lastly, we observe that more studies on the balanced use of AI tools would complement emerging and recent results of the effect of LLMs on novices. While studies published this year do support the conclusion that unguided use of LLM tools can be harmful or of no use to novices, studies on effectiveness of such tools when used in tandem with a traditional, bottom-up approach of teaching computing fundamentals are needed. Specifically, we are interested in how LLMs can be strategically employed alongside conventional teaching methods that emphasize basic coding principles and algorithm proficiency. The outcome of such research could be used to move our report up from observations and tentative advice into a complete educational framework.

4 ACKNOWLEDGMENTS

We wish to thank the anonymous peer reviewers for their thorough and helpful feedback on the paper.

REFERENCES

[1] Anne Castles, Kathleen Rastle, and Kate Nation. 2018. Ending the reading wars: Reading acquisition from novice to expert. *Psychological science in the public interest* 19, 1 (2018), 5–51.
[2] Sharon Cromwell. 2016. Whole Language and Phonics: Can They Work Together? https://www.educationworld.com/a_curr/curr029.shtml
[3] Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, Michel C Desmarais, and Zhen Ming Jack Jiang. 2023. Github copilot ai pair programmer: Asset or liability? *Journal of Systems and Software* 203 (2023), 111734.
[4] Samuel DeJulio, Dixie D Massey, Norman Stahl, and James King. 2024. Terminologically Speaking, the Reading Wars. *Reading Psychology* (2024), 1–27.
[5] Nick C Ellis, Miwa Natsume, Katerina Stavropoulou, Lorenc Hoxhallari, Victor HP Van Daal, Nicoletta Polyzoe, Maria-Louisa Tsipa, and Michalis Petalas. 2004. The effects of orthographic depth on learning to read alphabetic, syllabic, and logographic scripts. *Reading research quarterly* 39, 4 (2004), 438–468.
[6] Robert M Gagne. 1982. Learning from the top down and the bottom up. *Florida Journal of Educational Research* 24, 1 (1982), 1–10.
[7] Virginia Grande, Natalie Kiesler, and María Andreína Francisco R. 2024. Student Perspectives on Using a Large Language Model (LLM) for an Assignment on Professional Ethics. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1* (Milan, Italy) *(ITiCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 478–484. https://doi.org/10.1145/3649217.3653624
[8] Mark Guzdial. 2009. The Phonics of Computing Education. https://computinged.wordpress.com/2009/09/22/the-phonics-of-computing-education/ Accessed: 2024-06-03.
[9] O. Hazzan. 1999. Reducing Abstraction Level When Learning Abstract Algebra Concepts. *Educational Studies in Mathematics* 40 (1999), 71–90. Issue 1.
[10] Cindy E. Hmelo-Silver, Ravit Golan Duncan, and Clark A. Chinn. 2007. Scaffolding and Achievement in Problem-Based and Inquiry Learning: A Response to Kirschner, Sweller, and Clark (2006). *Educational Psychologist* 42, 2 (2007), 99–107. https://doi.org/10.1080/00461520701263368
[11] Chloe Hynes. 2018. Teaching Basic Literacy – Quick and easy strategies for the toolkit. https://pdnorth.org.uk/2018/06/25/teaching-basic-literacy/ Accessed: 2024-07-10.
[12] Anna A Ivanova, Shashank Srikant, Yotaro Sueoka, Hope H Kean, Riva Dhamala, Una-May O'reilly, Marina U Bers, and Evelina Fedorenko. 2020. Comprehension of computer code relies primarily on domain-general executive brain regions. *elife* 9 (2020), e58906.
[13] Anthony F. Jorm and David L. Share. 1983. An invited article: Phonological recoding and reading acquisition. *Applied Psycholinguistics* 4, 2 (1983), 103–147. https://doi.org/10.1017/S0142716400004380
[14] Jeremy Kilpatrick and Jane Swafford. 2002. *Helping children learn mathematics*. National Academy Press.
[15] Thomas Kim and Saul Axelrod. 2005. Direct instruction: An educators' guide and a plea for action. *The Behavior Analyst Today* 6, 2 (2005), 111.
[16] Vassilka D. Kirova, Cyril S. Ku, Joseph R. Laracy, and Thomas J. Marlowe. 2024. Software Engineering Education Must Adapt and Evolve for an LLM Environment. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (Portland, OR, USA) *(SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 666–672. https://doi.org/10.1145/3626252.3630927
[17] Paul A Kirschner, John Sweller, and Richard E Clark. 2006. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist* 41, 2 (2006), 75–86.
[18] Ryan Krueger, Yu Huang, Xinyu Liu, Tyler Santander, Westley Weimer, and Kevin Leach. 2020. Neurological divide: an fMRI study of prose and code writing. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 678–690.
[19] Yun-Fei Liu, Judy Kim, Colin Wilson, and Marina Bedny. 2020. Computer code comprehension shares neural resources with formal logical inference in the fronto-parietal network. *Elife* 9 (2020), e59340.
[20] David Ludden. 2015. Whole Language or No Language? *Psychology Today* (July 2015). https://www.psychologytoday.com/intl/blog/talking-apes/201507/whole-

language-or-no-language Accessed: 2024-07-10.

[21] Sarah Mitchell. [n. d.]. The Reading Wars Are Over – and Phonics Has Won. https://www.smh.com.au/national/nsw/the-reading-wars-are-over-and-phonics-has-won-20201127-p56ioj.html

[22] Louisa Moats. 2007. Whole-Language High Jinks: How to Tell When" Scientifically-Based Reading Instruction" Isn't. *Thomas B. Fordham Institute* (2007).

[23] Louisa Cook Moats. 2000. *Whole language lives on: The illusion of" balanced" reading instruction.* DIANE Publishing.

[24] U.S. Department of Education. 2001. No Child Left Behind Act of 2001. https://www2.ed.gov/nclb/landing.jhtml Accessed: 2024-07-22.

[25] National Council of Teachers of Mathematics. Commission on Standards for School Mathematics. 1989. *Curriculum and evaluation standards for school mathematics.* National Council of Teachers of Mathematics.

[26] Aadarsh Padiyath, Xinying Hou, Amy Pang, Diego Vargas, Xingjian Gu, Tamara Nelson-Fromm, Zihan Wu, and Mark Guzdial. 2024. Insights from Social Shaping Theory: The Appropriation of Large Language Models in an Undergraduate Programming Course. In *the ACM Conference on International Computing Education Research 2024.*

[27] Nick B Pandža. 2016. Computer programming as a second language. In *Advances in Human Factors in Cybersecurity: Proceedings of the AHFE 2016 International Conference on Human Factors in Cybersecurity, July 27-31, 2016, Walt Disney World®, Florida, USA.* Springer, 439–445.

[28] P David Pearson. 2004. The reading wars. *Educational policy* 18, 1 (2004), 216–252.

[29] J. Perrenet and E. Kaasenbrood. 2006. Levels of abstraction in students' understanding of the concept of algorithm: the qualitative perspective. *SIGCSE Bulletin* 38 (2006), 270–274. Issue 3.

[30] Scott R Portnoff. 2018. The introductory computer programming course is first and foremost a language course. *ACM Inroads* 9, 2 (2018), 34–52.

[31] James Prather, Brent N Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S Randrianasolo, Brett A. Becker, Bailey Kimmel, Jared Wright, and Ben Briggs. 2024. The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. In *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1* (Melbourne, VIC, Australia) *(ICER '24)*. Association for Computing Machinery, New York, NY, USA, 469–486. https://doi.org/10.1145/3632620.3671116

[32] Alan H Schoenfeld. 2004. The math wars. *Educational policy* 18, 1 (2004), 253–286. https://doi.org/10.1177/0895904803260042

[33] David L. Share. 2004. Orthographic learning at a glance: On the time course and developmental onset of self-teaching. *Journal of Experimental Child Psychology* 87, 4 (2004), 267–298. https://doi.org/10.1016/j.jecp.2004.01.001

[34] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, and André Brechmann. 2014. Understanding Programmers' Brains with fMRI. In *Frontiers in Neuroinformatics. Conference Abstract: Neuroinformatics 2014.* https://doi.org/10.3389/conf.fninf.2014.18.00040

[35] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. 2014. Understanding understanding source code with functional magnetic resonance imaging. In *Proceedings of the 36th international conference on software engineering.* 378–389.

[36] Janet Soler. 2016. The politics of the teaching of reading. *Prospects* 46 (2016), 423–433. https://doi.org/10.1007/s11125-017-9415-8

[37] Linda B Stebbins. 1977. *Education as experimentation: A planned variation model.* Vol. 4. University Press of America.

[38] John Sweller. 2021. Why inquiry-based approaches harm students' learning. *The Centre for Independent Studies Analysis Paper* 24 (2021), 1–10.

[39] H. Wu. 1997. The Mathematics Education Reform: Why You Should be Concerned and What You Can Do. *The American Mathematical Monthly* 104, 10 (1997), 946–954. http://www.jstor.org/stable/2974477

[40] Lixiang Yan, Lele Sha, Linxuan Zhao, Yuheng Li, Roberto Martinez-Maldonado, Guanliang Chen, Xinyu Li, Yueqiao Jin, and Dragan Gašević. 2024. Practical and ethical challenges of large language models in education: A systematic scoping review. *British Journal of Educational Technology* 55, 1 (2024), 90–112. https://doi.org/10.1111/bjet.13370

[41] Matthew Yglesias. 2023. The social science of reading isn't so clear. Blog post on Slow Boring. https://www.slowboring.com/p/the-social-science-of-reading-isnt Accessed: 2024-05-20.