The Education Column

BY

Dennis Komm and Thomas Zeume

ETH Zurich, Switzerland and Ruhr University Bochum, Germany

dennis.komm@inf.ethz.ch and thomas.zeume@rub.de

Surveying Theory of Computing Education in the United States

Ryan E. Dougherty
Department of Electrical Engineering & Computer Science
United States Military Academy
ryan.dougherty@westpoint.edu

Tim Randolph
Department of Computer Science
Harvey Mudd College
trandolph@g.hmc.edu

Abstract

We summarize the results of a 2024 survey of theory of computing (ToC) courses that received responses from 166 institutions of higher education in the United States [8]. Although our survey is far from comprehensive, it provides new information on many aspects of ToC in U.S. bachelor's degree programs in computer science. We report on the most frequently covered topics in ToC courses as well as structural features such as typical prerequisites, section sizes, teaching modalities, and course staff.

In addition to summarizing survey results, we briefly discuss observed trends, suggest opportunities for future teaching and research, and conclude with a call for increased collaboration between American and European theoretical computer science educators.

1 Introduction

While attending the 2024 Technical Symposium on Computer Science Education (SIGCSE TS) in Portland, Oregon, a group of computer science educators convened for lunch. We discussed our shared passion for teaching *theory of computing* (ToC): roughly, the formal concepts and theoretical frameworks that unify computer science as an academic discipline, as presented to undergraduate students during their degree program. We also lamented the relative lack of computer science education literature on the subject or its pedagogy.

As the reader of this bulletin is aware, the umbrella term "theory of computing" covers a variety of related topics and its boundaries are not precisely defined. The group that met at SIGCSE TS 2024 broadly agreed on the inclusion of some topics, including formal languages, automata theory, computability, and basic concepts from complexity theory. However, we disagreed about the status of others. We taught at a wide range of institutions of higher education in the United States, including small teaching colleges, large public universities, and research-focused institutions; accordingly, our disagreements might have resulted from the institutions at which we worked. Alternatively, they might have arisen from regional, personal, or sub-disciplinary differences. Moreover, we recognized that the group was not necessarily representative of theoretical computer science educators in the U.S. Even if *we* agreed on what ToC was and how to teach it, we might not necessarily be able to speak for others.

This uncertainty presented a problem. We all shared the desire to improve our teaching, but we lacked objective information about the diversity of ToC teaching practices across the country. What topics were usually being taught in U.S. colleges and universities in a first course in ToC? How were these topics taught, by whom, and how were they organized into units, courses, and curricula? We found very little existing research on this question, so we conceived a broad survey to find out.

2 A Survey of Theory of Computing Courses

The survey project became a working group convened for the 2024 ACM Virtual Global Computing Education Conference (SIGCSE Virtual 2024); see [7] for the initial abstract. Although the group included some European researchers, the majority of members were based in the U.S. Accordingly, we narrowed the scope of our survey to the U.S. so that the project was feasible within the working group time constraints. In this respect, the mission that motivated our survey is still incomplete: we need a better understanding of how the content and rationale of ToC education differs between educational systems in the U.S., in European countries, and elsewhere.

Our first challenge was providing a definition of ToC that was simultaneously concrete enough for respondents to understand the aim of the survey, and flexible enough to capture topics and approaches to teaching we did not anticipate. We adopted a bottom-up approach by constructing a list of "ToC topics" based on the "Computational Models and Formal Languages" unit of the 2023 ACM Computer Science curricular guidelines [14]. We then grouped these topics under three broad headings and defined a theory course (for our purposes) to be any course with substantial coverage of one or more of the following:

- **Automata theory:** finite automata, regular expressions, regular languages,

the pumping lemma, context-free languages/grammars, etc.

- **Computability theory:** Turing machines, reductions, (un)decidability, etc.
- **Introductory complexity theory:** the complexity classes P and NP, NP-hardness, NP-completeness, polynomial-time reductions, etc.

Our definition does not attempt to encompass the full breadth of theory courses in the U.S., let alone worldwide. In particular we chose to exclude some closely related subfields, including algorithms and logic, from our definition.[1] Although the list was informed by group consensus and prior assumptions about what a "theory course" might look like, we asked respondents to provide additional "missing" topics; their responses largely confirmed our expectations about what an introductory theory course might include.

We distributed copies of the survey to representatives of over 1,000 institutions of higher education in the U.S. This list included a majority of all U.S. institutions that offer bachelor's degrees in computer science or a closely related field, and all of those for which we could obtain relevant contact information. Where possible, we enlisted senior faculty members with ToC teaching experience to summarize ToC teaching at their institution, defaulting to department chairs or other senior faculty members where necessary. Our 166 survey responses spanned the wide variety of U.S. institutions of higher education, from "R1" universities with very high research spending to small liberal arts colleges (SLACs) devoted to undergraduate education. From these responses, we assembled a preliminary picture of ToC teaching.

3 The Mythical Median Theory Course

As an aid to the reader's imagination, consider the "mythical median theory course": a course that represents the majority or significant plurality of our responses in qualitative dimensions and the median in quantitative dimensions. This course is "mythical" in the sense that it may not perfectly describe any course in particular. In constructing it, we have ignored correlations between the various dimensions of our data. Furthermore, we caution that even if the plurality of *courses* have some feature in common, this feature is not necessarily experienced by the plurality of *students*: for instance, one course with a very large enrollment might contain more students than many small courses put together.

[1] Our data confirmed our working assumption that American students often encounter theory of computing and algorithms in separate courses. See Luu et al. for a survey of introductory undergraduate algorithms courses [16]. For the complete topic list and further discussion of the rationale behind it, see the full report [8].

In this section we provide only rough approximations of our data. This is intentional: we wish to give necessary context for our claims without implying any particular degree of statistical exactness. For complete data on each of the features listed below, we encourage the reader to refer to the full report [8].

3.1 Frequently Covered ToC Topics

We begin with the curriculum: from our long list of possibilities, what ToC topics are actually taught? Our imposed definition of ToC limits our results to topics associated with automata, computability, and complexity theory, but within these domains some topics are more popular than others. Perhaps surprisingly, we found general agreement on a set of core ToC topics that a majority of institutions covered somewhere in their computer science degree program. We identified the following trends:

1. **Almost all responding institutions include proof techniques as part of their CS curricula.** We asked survey respondents about only one topic that did not fall under the heading of automata, computability, or complexity: techniques for writing mathematical proofs. Proof techniques are a required part of the computer science degree program for about 90 % of respondents, and most of the remaining respondents specified that this skill was covered in a regularly offered elective course. This seems to illustrate a strong shared commitment to proof-writing among our sample. (Note that if proof techniques were part of a discrete mathematics class required for computer scientists, this would still be counted using our methodology.)

2. **Almost all responding institutions cover at least one topic in automata theory, computability, and complexity.** Approximately 90 % of institutions reported that they covered at least one ToC topic in automata theory in at least one course in their computer science curriculum. The same was true for computability and for complexity theory (each considered independently).

3. **Automata theory topic coverage.** Most responding institutions cover regular languages and (non-)deterministic finite automata in their computer science curriculum. Fewer, about half, cover context-free grammars and pushdown automata, and a minority cover other topics including other automata and language classes.

4. **Computability and complexity theory topic coverage.** Coverage of ToC topics in computability theory and complexity theory was somewhat less consistent than the core topics in automata theory. Most respondents reported covering Turing machines and basic computability and complexity-theoretic

definitions (decidability, recognizability, the complexity class P, the complexity class NP, etc.) in their computer science curricula. Slim majorities covered recursively enumerable languages, mapping (many-one) and polynomial-time reductions, and the Chomsky hierarchy. Only a minority covered topics such as space complexity, Rice's Theorem, the Cook-Levin Theorem, and other computational models equivalent in power to Turing machines, such as the lambda calculus.

The names of the courses in which ToC topics appear give us further clues into the way ToC is divided up across the curriculum. For example, almost half of the courses in our sample that cover at least one topic in automata theory include the phrase "theory of computing" or the phrase "theory of computation" in their titles. These same courses often covered computability theory and complexity theory topics as well. In other words, "ToC courses" that correspond approximately to our working definition do exist at many institutions.

Many other courses covering automata theory included more specific terms such as "formal language" or "automata theory," and a few included "programming languages" or "programming language theory." Courses covering at least one computability topic were more divided, with "theory of computing" commonly included in their titles along with the more specific term "computability" and, occasionally, "complexity." Finally, although many courses that covered at least one topic in basic complexity theory were the same as those covering automata and computability, about a third were covered in courses whose name contained the term "algorithm" (presumably, courses focused on algorithm design and analysis). Relatively few courses in which students encountered our complexity theory topics contained the word "complexity" in their title, perhaps an indication that students rarely first encounter complexity theory in a course devoted primarily to the topic.

3.2 Structural Features and Teaching Modality

With the curriculum of the mythical median theory course established, we proceed to the structural features, including prerequisites, enrollment, teaching modality, and teaching staff, that define the mythical median theory course.

1. **Prerequisite courses.** The median theory course requires a course in *discrete mathematics* (about 3/4 of the courses in our sample), and is likely to require an *introduction to programming* course and a course on *data structures* (each about 2/3 of the courses in our sample). It appears that the median theory course occurs midway through the degree program, after students have taken a standard set of introductory courses, but does not require other special preparation.

2. **Enrollment.** The median theory course enrolls between 10 and 29 students per section. This is likely influenced by the preponderance of small colleges in our sample, but there are too many courses that offer small sections of ToC for this to be explained by small colleges alone. Courses with enrollments in the range of 10–29 students per section account for a majority of all ToC courses reported (about 3/5 of the courses in our sample), and nearly 9 out of 10 courses in our sample had typical enrollments under 40 students per section. In other words, few sections of ToC courses are large lectures.

3. **Teaching modality.** The median theory course is likely to be taught via lecture, perhaps punctuated with sessions devoted to small-group problem-solving. It is slightly more likely to be taught from a whiteboard or blackboard than via slides, although both are common. The median theory course does not use ToC-specific digital tools, with the possible exception of the formal language software JFLAP [20] (about 1/4 of the courses in our sample). The median theory course does not use other digital tools such as clickers for live polling.

4. **Teaching staff.** The median theory course is typically taught by a permanent faculty member who is also a computer scientist; few courses are taught by faculty in other disciplines or adjunct faculty members.[2] A majority of responding institutions reported that their ToC instructors focus primarily on teaching, but a substantial minority reported that their ToC instructors focus primarily on research. Similar numbers of institutions reported that their ToC courses are taught by specialists in theory of computing / algorithms and by computer scientists with other specialties.

   Almost all sections of the median theory course have few (0–2) teaching assistants or student helpers (more than 9/10 of the courses in our sample), corresponding to the relatively small median section size.

In sum, the median theory course is relatively small and relatively traditional in the way it is taught. Most ToC courses in our sample are taught by computer scientists and require prerequisite computer science courses, implying that ToC courses in the U.S. are associated much more closely with computer science as a discipline than with other disciplines such as mathematics.

[2] Part-time instructional staff.

4 Future Opportunities for Teaching and Education Research in ToC

Our "mythical median theory course" takes place in a small lecture hall and has many features in common with traditional mathematics education, including board work, slides, and a limited reliance on digital tools. This may not be surprising, but it provides the context required to discuss teaching innovations in response to internal changes and external pressures on the field of computer science.

The suggestions we offer in this section are speculative and invite future experiments, both in the formal context of computer science education research and the informal context of one's own classroom.

- **Rethinking assessment methods.** Generative Artificial Intelligence (GenAI) threatens traditional assessment methods across many academic disciplines, although it has naturally attracted special attention among computer science education researchers [21]. Students may attempt to cheat using GenAI in any class where written work is used as a proxy for learning, and this includes the many ToC classes that rely on problem sets.[3] Moreover, they may succeed: recent research has determined that GenAI models can effectively solve standard undergraduate ToC problems [5, 9, 12, 15]. Moreover, assessments that tempt students to use GenAI have additional consequences beyond unfair grading and failure to learn the material: some CS education research suggests that overreliance on GenAI can hurt students' grades and their programming ability [13].

  In response, we suggest that ToC educators experiment with assessment methods that prioritize critical analysis and effective communication of technical material, especially when these evaluations can be conducted in person by peers or instructors. Limited existing research suggests that these methods may be effective for teaching ToC [6, 22], although more study is needed.

  A drawback to interactive and in-person assessment models is that they are difficult to implement in large courses and with less experienced instructional staff. However, our survey indicates that many ToC classes occur in small sections taught by permanent faculty, so assessments of this type may be feasible. A small ToC class might even serve as a laboratory in which to test alternative assessment methods before they are adapted to larger classes in a computer science degree program.

[3] Unfortunately, our survey does not ask about the prevalence of various assessment methods in ToC classes. We view this as an important open question for future work.

- **Increasing student agency and motivation.** Our survey results indicate that for many students, their first class in ToC is also the first class in their degree program in which they apply formal mathematical methods to computer science. This may be connected to the perception among some students that ToC is particularly difficult (perhaps even "dry") [22, 23].

  This provides a second motivation for experimentation in ToC teaching. As one potential solution, we suggest trying existing pedagogical strategies that increase student engagement by focusing on agency and motivation. For example, some early research has shown effectiveness of active learning in ToC courses [11].

  A group of related teaching strategies, referred to collectively as "alternative grading", addresses motivation to learn by challenging traditional points-based grading systems [10, 18]. A significant amount of work considers the application of alternative grading methods to computer science, but relatively little considers theoretical computer science specifically. Exceptions include Weber's implementation of mastery grading in an algorithms course [24] and Randolph's experience report on participatory governance in ToC [19].

- **Multimodal learning.** Our survey found that digital tools are rarely used in ToC classrooms. Of course, this could be due to incompatibility between existing tools and the material: perhaps ToC is simply poorly suited for multimodal learning. However, we don't believe this is the case.

  Moreover, many other visualization tools exist for ToC instruction, such as JFLAP [20], TheoryViz [2], FAdo/GUItar [1], Gidayu [3], and FSM [17]; see [4] for a general overview. A substantial minority of classes indicated using JFLAP for teaching automata theory indicates an appetite for more visual learning. Given the relative abundance and availability of visualization tools, our finding that they are rarely used in ToC courses is surprising. However, the relative abundance of visualization tools contrasts with their very limited adoption; One possibility is that many ToC educators are unaware of the tools that do exist.

  Additionally, multimodal content can easily transcend the classroom in the form of online demos and videos. Online ToC courses exist, but videos are typically limited to lecture recordings; perhaps there is a niche for scripted video content.

Our suggestions—pivot assessment away from graded problem sets towards oral evaluation, focus on student motivation, and provide more visual, intuitive, and interactive learning models—combine our limited knowledge of the present landscape of ToC teaching with recommendations familiar from similar disciplines

and learning settings. However, a more daring approach might set aside what the median ToC course *is* and ask instead, "What should a ToC course *be*?"

The ToC topics covered by the median theory course are well-established and stable, especially by the fast-moving standards of computer science. In most cases, this is for good reason: concepts such as regular languages and decidability are essential to a theoretical view of computer science. However, if the curriculum becomes static, we risk fossilizing all the auxiliary concepts that come along with our preferred abstractions. For example, is spending a week wrangling over pumping lemma proofs the most effective use of student-instructor contact time in an introductory ToC class? Instructors may disagree, and the right answer may be classroom-specific. However, we believe that continually interrogating the ToC curriculum is essential for a healthy discourse around teaching practice, which in turn is useful to educators of all types.

5 An Invitation to Collaborate

Our survey of ToC in U.S. institutions was never intended as an abstract data-gathering exercise. Rather, it was a product of our desire to become better educators. We decided that an important step towards this goal was to get a better understanding of the way people teach theory of computing now; this, we hope, will allow us to make future teaching decisions with a better understanding of the way our peers teach and design research programs that are relevant to a broader audience of ToC educators.

These goals are naturally collaborative, and we believe we will continue to benefit from a greater understanding of ToC teaching outside the United States. Therefore, we conclude this article with an invitation to computer science researchers and educators: let us collaborate and learn from each other!

Acknowledgements

First and foremost, we are deeply grateful to all of our survey co-authors: Tzu-Yi Chen, Jeff Erickson, Matthew Ferland, Dennis Komm, Jonathan Liu, Timothy Ng, Ana Smaranda Sandu, Michael Shindler, Edward Talmage, and Thomas Zeume. We also thank SIGCSE Virtual and the ACM for providing the forum for this work, working group chairs Jacqueline Whalley and Juho Leinonen for their guidance throughout the development of the project, and several anonymous referees for extensive and thoughtful feedback.

The opinions in this work are solely of the authors, and do not necessarily reflect those of the U.S. Army, U.S. Army Research Labs, the U.S. Military Academy, or the Department of Defense.

References

[1] André Almeida, Marco Almeida, José Alves, Nelma Moreira, and Rogério Reis. Fado and guitar: Tools for automata manipulation and visualization. In Sebastian Maneth, editor, *Implementation and Application of Automata*, pages 65–74, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Lillian Baker, Sierra Zoe Bennett-Manke, Sebastian Neumann, Ian Njuguna, and Ryan E. Dougherty. TheoryViz: A visualizer tool for theory of computing concepts. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2*, SIGCSE TS 2025, pages 1373–1374, New York, NY, USA, February 2025. Association for Computing Machinery.

[3] Tiago Cogumbreiro and Gregory Blike. Gidayu: Visualizing automaton and their computations. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, ITiCSE 2022, pages 110–116, New York, NY, USA, 2022. Association for Computing Machinery.

[4] Ian Campbell, Anthony Notaro, and Ryan E. Dougherty. Visualization tools for cs theory: an initial literature review. (accepted to SIGCSE TS 2026).

[5] Ryan E. Dougherty, Matei A. Golesteanu, and Garrett B. Vowinkel. Large language models with reasoning on theory course exams. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 2*, ITiCSE 2025, page 785, New York, NY, USA, 2025. Association for Computing Machinery.

[6] Ryan E. Dougherty. Experiences with scaffolding research projects in theory of computing courses. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2025, pages 180–186, New York, NY, USA, 2025. Association for Computing Machinery.

[7] Ryan E. Dougherty, Tim Randolph, Tzu-Yi Chen, Jeff Erickson, Matthew Ferland, Dennis Komm, Jonathan Liu, Timothy Ng, Seth Poulsen, Smaranda Sandu, Michael Shindler, Edward Talmage, and Thomas Zeume. A survey of undergraduate theory of computing curricula. In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 2*, SIGCSE Virtual 2024, pages 281–282, New York, NY, USA, 2024. Association for Computing Machinery.

[8] Ryan E. Dougherty, Tim Randolph, Tzu-Yi Chen, Jeff Erickson, Matthew Ferland, Dennis Komm, Jonathan Liu, Timothy Ng, Ana Smaranda Sandu, Michael Shindler, Edward Talmage, and Thomas Zeume. A survey of undergraduate theory of computation curricula in the United States. In *2024 Working Group Reports on 1st ACM Virtual Global Computing Education Conference*, SIGCSE Virtual–WGR 2024, pages 1–14, New York, NY, USA, 2025. Association for Computing Machinery.

[9] Ming Ding, Federico Soldà, Weixuan Yuan, and Rasmus Kyng. Assessing GPT performance in a proof-based university-level course under blind grading. *Bulletin of the EATCS*, 146, 2025.

[10] Joe Feldman. *Grading for Equity: What It Is, Why it Matters, and How It Can Transform Schools and Classrooms*. Corwin Press, 2023.

[11] Michael T. Grinder, Seong B. Kim, Teresa L. Lutey, Rockford J. Ross, and Kathleen F. Walsh. Loving to learn theory: active learning modules for the theory of computing. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, SIGCSE 2002, pages 371–375, New York, NY, USA, 2002. Association for Computing Machinery.

[12] Matei A. Golesteanu, Garrett B. Vowinkel, and Ryan E. Dougherty. Can ChatGPT pass a theory of computing course? In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 1*, SIGCSE Virtual 2024, pages 33–38, New York, NY, USA, 2024. Association for Computing Machinery.

[13] Gregor Jošt, Viktor Taneski, and Sašo Karakatič. The impact of large language models on programming education and student learning outcomes. *Applied Sciences*, 14(10):4115, 2024.

[14] Amruth N. Kumar, Rajendra K. Raj, Sherif G. Aly, Monica D. Anderson, Brett A. Becker, Richard L. Blumenthal, Eric Eaton, Susan L. Epstein, Michael Goldweber, Pankaj Jalote, Douglas Lea, Michael Oudshoorn, Marcelo Pias, Susan Reiser, Christian Servin, Rahul Simha, Titus Winters, and Qiao Xiang. *Computer Science Curricula 2023*. Association for Computing Machinery, New York, NY, USA, 2024.

[15] Nero Li, Shahar Broner, Yubin Kim, Katrina Mizuo, Elijah Sauder, Claire To, Albert Wang, Ofek Gila, and Michael Shindler. Investigating the capabilities of generative AI in solving data structures, algorithms, and computability problems. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE TS 2025, pages 659–665, New York, NY, USA, 2025. Association for Computing Machinery.

[16] Michael Luu, Matthew Ferland, Varun Nagaraj Rao, Arushi Arora, Randy Huynh, Frederick Reiber, Jennifer Wong-Ma, and Michael Shindler. What is an algorithms course? Survey results of introductory undergraduate algorithms courses in the US. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE TS 2023, pages 284–290, 2023.

[17] Marco T. Morazan and Tijana Minic. Nondeterministic to deterministic finite-state machine visualization: Implementation and evaluation. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2024, pages 262–268, New York, NY, USA, 2024. Association for Computing Machinery.

[18] Linda B Nilson, David Clark, and Robert Talbert. *Grading For Growth: A Guide to Alternative Grading Practices That Promote Authentic Learning and Student Engagement In Higher Education*. Routledge, 2023.

[19] Tim Randolph. Participatory governance in the computer science theory classroom. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE TS 2024, pages 1091–1097, New York, NY, USA, 2024. Association for Computing Machinery. event-place: Portland, OR, USA.

[20] Susan H. Rodger and Thomas W. Finley. *JFLAP: An Interactive Formal Languages and Automata Package*. Jones & Bartlett Learning, 2006.

[21] Nishat Raihan, Mohammed Latif Siddiq, Joanna CS Santos, and Marcos Zampieri. Large language models in computer science education: A systematic literature review. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, pages 938–944, 2025.

[22] Scott Sigman. Engaging students in formal language theory and theory of computation. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE 2007, pages 450–453, New York, NY, USA, 2007. Association for Computing Machinery.

[23] Florian Schmalstieg, Marko Schmellenkamp, Jakob Schwerter, and Thomas Zeume. Difficulty generating factors for context-free language construction assignments. In *Proceedings of the 2025 ACM Conference on International Computing Education Research V. 1*, ICER 2025, pages 196–209, 2025.

[24] Robbie Weber. Using alternative grading in a non-major algorithms course. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE TS 2023, pages 638–644, 2023.