Report from Dagstuhl Seminar 24251

Teaching Support Systems for Formal Foundations of Computer Science

Tiffany Barnes*[1], Jan Vahrenhold*[2], Thomas Zeume*[3], and Florian Schmalstieg†[4]

1 **North Carolina State University – Raleigh, US.** `tiffany.barnes@gmail.com`
2 **Universität Münster, DE.** `jan.vahrenhold@uni-muenster.de`
3 **Ruhr-Universität Bochum, DE.** `thomas.zeume@rub.de`
4 **Ruhr-Universität Bochum, DE.** `florian.schmalstieg@rub.de`

Abstract

Introductory courses on formal foundations of computer science – including basic courses on theoretical computer science (regular and context-free languages, computability theory, and complexity theory) as well as on logic in computer science (propositional and first-order logic, modeling, and algorithms for evaluation and satisfaction of formulas) – are a cornerstone of computer science curricula, yet many students struggle with their often theoretical contents. The recent influx of students in computer science, as well as the shift towards the inclusion of more online-based teaching ask for advanced teaching support systems that aid both students and instructors.

This Dagstuhl Seminar focussed on fostering discussion between researchers in computing education, builders of systems for teaching formal foundations, as well as instructors of these foundations in order to facilitate more robust research and development of systems to support teaching and learning of the formal foundations of computer science.

Seminar June 16–21, 2024 – https://www.dagstuhl.de/24251
2012 ACM Subject Classification Social and professional topics → Computing education; Theory of computation; Applied computing → Education
Keywords and phrases artificial intelligence in education, computing education research, educational data mining, formal foundations of computer science, intelligent tutoring systems, user modeling and adaptive personalization, user studies
Digital Object Identifier 10.4230/DagRep.14.6.108

1 Executive Summary

Thomas Zeume (Ruhr-Universität Bochum, DE)
Tiffany Barnes (North Carolina State University – Raleigh, US)
Jan Vahrenhold (Universität Münster, DE)

The primary goal of this Dagstuhl Seminar was to determine how to enable communication between between researchers in computing education, builders of systems for teaching formal foundations, as well as instructors of these foundations. While these groups have very similar interests, they also have very different notions, foci, and methods. In particular, participants from the "formal foundations" community talk about the "hardness" of a problem in terms of

* Editor / Organizer
† Editorial Assistant / Collector

its computational complexity, participants from the "intelligent tutoring systems" community are concerned with whether or not a system can scale or how to best provide feedback to the learner, and participants from the "computing education research" community study the effectiveness of teaching methods for learning, e.g., the cognitive load, student learning, etc.

Within the first one-and-a-half days of the seminar, tutorials on Formal Foundations of CS, CS Education Research, and Intelligent Tutoring Systems given by experts of the respective domains set the stage for the rest of the seminar. The tutorial on CS Education Research was interspersed with breakout sessions for applying the theoretical content of the tutorial to projects of seminar participants, leading to intense discussions across the different communities and therefore being very effective also in bridging barriers between the communities. On the afternoon of the first day, tools and tutoring systems in the formal foundation domain were presented in teaser and poster sessions as well.

The rest of the seminar was centered around breakout sessions, whose research and discussion topics were proposed and voted on by participants. There were a few contributed research talks and occasional ad-hoc tutorial-like sessions as they became relevant for the breakout sessions.

Participants noted that the seminar had a very open atmosphere and that the different research communities were eager to learn from each other. This welcoming spirit was also reflected by a music event on one of the evenings where three of the participants gave a concert and a fare-well magician's show by one of the participants as part of the closing session.

In summary, it was a very fruitful seminar – both with respect to research collaborations and personal interactions. The goal of bringing together the communities and bridging the gaps between them was fully achieved. Several collaborative research projects were initiated during the seminar and are currently being followed-up on.

2 Table of Contents

3 Overview of Talks

3.1 Intelligent Tutoring Systems – An Introduction

Johan Jeuring (Utrecht University, NL)

I presented a brief introduction to Intelligent Tutoring Systems. In this abstract I will review the themes I discussed, and include pointers for further reading.

A good introduction to Tutoring Systems is VanLehn's the behavior of tutoring systems [17]. He distinguishes two components:

- the inner loop, in which a student works on a task, takes steps towards solving the task, and a tutoring system provides support in the form of feedback and hints;
- the outer loop, which helps a student with finding a path in the learning material, for example by suggesting next tasks to work on.

Through the years, many approaches to supporting a student when solving an exercise (the inner loop) have been developed:

- cognitive tutors, built upon theories such as ACT-R [7];
- constraint-based tutors [6];
- domain reasoners [5, 4, 3];
- data-driven tutors [2];
- LLM-based tutors [15];
- and more.

Some approaches use a student model to keep track of the learning progress of a student, and to adapt the kind of feedback given to a student. There exist many approaches to student modelling [1]; some well known examples are:

- ELO ratings
- Overlay models
- Knowledge space theory
- Constraint-based modelling
- Bayesian modelling
- Model tracing

Student models are also used to support the outer loop. The outer loop typically presents tasks to a student. The sequence in which tasks are offered is often fixed, sometimes determined by the student, and sometimes supported by a giving suggestions for a next task to work on. Recommendations can be based on a learner model and task attributes, on behavior from other students, on ratings on earlier items, and sometimes other components [8].

Many experiments have been performed to study the effectiveness of Tutoring Systems. VanLehn has shown that if you compare the effects of a tutor that supports stepwise exercises with the help of teaching assistants, there is little difference [16]. Quite a few other meta-reviews on the effectiveness of tutoring systems have been performed, but it is hard to use these reviews to make general statements: they regularly compare apples and pears [9, 10].

There is quite a lot of recent work on teaching-support systems for formal foundations of computer science [11, 12, 13, 14]. A complete overview would require a more systematic approach.

References

1 Konstantina Chrysafiadi and Maria Virvou: Student modeling approaches: A literature review for the last decade. Expert Syst. Appl. 40(11): 4715-4729 (2013)

2 Behrooz Mostafavi and Tiffany Barnes: Evolution of an Intelligent Deductive Logic Tutor Using Data-Driven Elements. Int. J. Artif. Intell. Educ. 27(1): 5-36 (2017)

3 Bastiaan Heeren and Johan Jeuring: Feedback services for stepwise exercises. Sci. Comput. Program. 88: 110-129 (2014)

4 Bastiaan Heeren, Johan Jeuring and Alex Gerdes: Specifying Rewrite Strategies for Interactive Exercises. Math. Comput. Sci. 3(3): 349-370 (2010)

5 Josje Lodder, Bastiaan Heeren and Johan Jeuring: A Domain Reasoner for Propositional Logic. J. Univers. Comput. Sci. 22(8): 1097-1122 (2016)

6 Antonija Mitrovic, Michael Mayo, Pramuditha Suraweera and Brent Martin: Constraint-Based Tutors: A Success Story. IEA/AIE 2001: 931-940

7 J.R. Anderson, A. T. Corbett, K.R. Koedinger and Ray Pelletier. (1995). Cognitive Tutors: Lessons Learned. Journal of the Learning Sciences, 4(2), 167–207. `https://doi.org/10.1207/s15327809jls0402_2`

8 M. Deschênes. Recommender systems to support learners' Agency in a Learning Context: a systematic review. Int J Educ Technol High Educ 17, 50 (2020). `https://doi.org/10.1186/s41239-020-00219-w`

9 J. A. Kulik and J. D. Fletcher (2016). Effectiveness of Intelligent Tutoring Systems: A Meta-Analytic Review. Review of Educational Research, 86(1), 42-78. `https://doi.org/10.3102/0034654315581420`

10 Alan C.K. Cheung and Robert E. Slavin. The effectiveness of educational technology applications for enhancing mathematics achievement in K-12 classrooms: A meta-analysis, Educational Research Review, Volume 9, 2013, Pages 88-113, ISSN 1747-938X, `https://doi.org/10.1016/j.edurev.2013.01.001`.

11 Marko Schmellenkamp, Alexandra Latys, Thomas Zeume: Discovering and Quantifying Misconceptions in Formal Methods Using Intelligent Tutoring Systems. SIGCSE (1) 2023: 465-471

12 Seth Poulsen, Mahesh Viswanathan, Geoffrey L. Herman and Matthew West: Proof Blocks: Autogradable Scaffolding Activities for Learning to Write Proofs. ITiCSE (1) 2022: 428-434

13 Matthew Farrugia-Roberts, Bryn Jeffries, Harald Søndergaard: Programming to Learn: Logic and Computation from a Programming Perspective. ITiCSE (1) 2022: 311-317

14 Preya Shabrina, Behrooz Mostafavi, Mark Abdelshiheed, Min Chi and Tiffany Barnes. Investigating the impact of backward strategy learning in a logic tutor: Aiding subgoal learning towards improved problem solving. *International Journal of Artificial Intelligence in Education*, pages 1–37, 2023.

15 Angelo Sifaleras. *Generative Intelligence and Intelligent Tutoring Systems: 20th International Conference, ITS 2024, Thessaloniki, Greece, June 10–13, 2024, Proceedings, Part I.* Springer Nature, 2024.

16 Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011.

17 Kurt VanLehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006.

3.2 Cognitive Science Concepts You Can Use

Shriram Krishnamurthi (Brown University – Providence, US), Rodrigo Duran (Federal Institute of Mato Grosso do Sul, BR), and R. Benjamin Shapiro (University of Washington – Seattle, US)

A tour of some of the central theories of cognitive science that are directly applicable in tools for teaching computing foundations (and also in generic pedagogy).

3.3 A Formal-Language-Based Framework for Computing Feedback Information Generically

Martin Lange (Universität Kassel, DE)

Joint work of Florian Bruse, Martin Lange
Main reference Florian Bruse, Martin Lange: "Computing All Minimal Ways to Reach a Context-Free Language", in Proc. of the Reachability Problems – 18th International Conference, RP 2024, Vienna, Austria, September 25-27, 2024, Proceedings, Lecture Notes in Computer Science, Vol. 15050, pp. 38–53, Springer, 2024.
URL https://doi.org/10.1007/978-3-031-72621-7_4

We present a theory of rewriting a word into a given target language. We show that the natural notion of equivalence between corrections as sequences of edit operations can be captured syntactically by means of a rather simple rewrite system. Completeness relies on a normal form for corrections that is then also used to develop a notion of minimality for corrections. This is not based on edit distance between words and languages but on a subsequence order on corrections, capturing the intuitive notion of doing a minimal number of rewriting steps. We show that the number of minimal corrections is always finite, and that they are computable for context-free languages.

The motivation for this theory and the more intricate notion of minimality is drawn from the study of digital classroom environments where language learning is required. Minimal corrections can be used to give individually targeted feedback and thus guide the learning process automatically.

3.4 Intelligent Tutoring Systems – Tools (What makes them intelligent?)

Martin Lange (Universität Kassel, DE), Tiffany Barnes (North Carolina State University – Raleigh, US), Felix Freiberger (Universität des Saarlandes – Saarbrücken, DE), Michael Goedicke (Universität Duisburg – Essen, DE), Norbert Hundeshagen (Universität Kassel, DE), Johan Jeuring (Utrecht University, NL), Alexandra Mendes (University of Porto, PT & INESC TEC – Porto, PT), Seth Poulsen (Utah State University, US), and Francois Schwarzentruber (IRISA – ENS Rennes, FR)

The original motivation for this breakout session was given by a simple question: how can intelligent tutoring systems (ITS) be made intelligent? Is the power of LLMs for example sufficient to guarantee a level of machine intelligence that is sufficient for generating feedback in ITS for guiding students through particular learning processes? But then, ITS have been designed and in use before LLMs came up, and therefore other techniques have been used to create specific forms of intelligence in these tools.

The aim of this breakout session was then to analyse and perhaps categorise and quantify such forms of intelligence in order to answer the question above in a way that ideally would tell the developers of ITS what technology to implement in their tools in order to achieve certain forms of intelligent feedback.

Not surprisingly, the concept of intelligence – even in the restricted setting of tutoring systems for formal foundations of computer science – is not easily categorised and quantified. So a large amount of time in this breakout session was initially spent on personal reports on what is used in particular tools. Examples of such methods include the following general methods.

- Comparing the way that a student constructs a solution to successful paths taken from previous attempt (of other students).
- Comparing a student's solution – either the final result or, in interactive tools, the construction path – to some master solution in the form of distance measures or, more generally, as inputs to some abstract problems, for instance comparing actions traces.
- Provide a set of rules that are allowed to be applied in order to construct a correct solution.

This has also sparked off a brief discussion on what should be judged as a correct solution: just the final answer, or the entire construction path. The latter perhaps needs a higher level of intelligence in an ITS. Another very much related question that has been discussed is: what technology can be used to intelligently create good exercises automatically?

The breakout session indentified some general technologies that can be used to create some form of intelligence or other, either in marking, feedback generation or creation of examples and exercise:

- LLMs
- SAT solvers and, more generally, SMT/CSP solvers,
- enumeration algorithms researched primarily in combinatorics and discrete math.

3.5 Automated Proof by Induction Feedback

Seth Poulsen (Utah State University – Logan, US)

Joint work of Seth Poulsen, Chenyan Zhao, Yael Gertner, Benjamin Cosman, Shubhang Kulkarni, Mahesh Viswanathan, Hongxuan Chen, Geoffrey Herman, Matthew West
Main reference Chenyan Zhao, Mariana Silva, Seth Poulsen: "Autograding Mathematical Induction Proofs with Natural Language Processing", CoRR, Vol. abs/2406.10268, 2024.
URL https://doi.org/10.48550/ARXIV.2406.10268

This talk is about software tools that help students learn to write mathematical proofs. Research has shown that timely feedback can be very helpful to students learning new skills. First I introduce Proof Blocks, a tool which enables students to construct mathematical proofs by dragging and dropping prewritten proof lines into the correct order instead of needing to write them from scratch. The instructor specifies the dependency graph of the lines of the proof, so that any correct arrangement of the lines can receive full credit. We develop a novel algorithm which enables assigning students' partial credit on Proof Blocks problems based on the number of edits that their submission is from a correct solution.

For assessment, we provide statistical evidence that Proof Blocks are easier than written proofs, which are typically very difficult. We also show that Proof Blocks problems provide about as much information about student knowledge as written proofs. Survey results show that students believe that the Proof Blocks user interface is easy to use, and that the questions accurately represent their ability to write proofs.

Next, I present a set of training methods and models capable of autograding freeform mathematical proofs by leveraging existing large language models and other machine learning techniques. We recruit human graders to grade the same proofs as the training data, and find that the best grading model is also more accurate than most human graders.

With the development of these grading models, we create and deploy an autograder for proof by induction problems and perform a user study with students. Results from the study shows that students are able to make significant improvements to their proofs using the feedback from the autograder, but students still do not trust the AI autograders as much as they trust human graders. Future work can improve on the autograder feedback and figure out ways to help students trust AI autograders.

3.6 Theory & Methods in Computing Education Research

R. Benjamin Shapiro (University of Washington – Seattle, US) and Shriram Krishnamurthi (Brown University – Providence, US)

A tour through some of the foundational ideas in education research applicable in computing. This was followed by a description of good and poor research questions. Finally, we presented three "grammars" for structuring a study.

3.7 Formal Foundations of Computer Science: A personal perspective

Thomas Zeume (Ruhr-Universität Bochum, DE)

In this tutorial, I will outline typical topics and methods included in introductory courses on formal foundations of computer science. Material from introductory courses in logic and theoretical computer science at the Ruhr-Universität Bochum will be used as examples.

4 Demos

4.1 pseuCo Book

Felix Freiberger (Universität des Saarlandes – Saarbrücken, DE)

Main reference Felix Freiberger: "pseuCo Book: An Interactive Learning Experience", in Proc. of the ITiCSE 2022: Innovation and Technology in Computer Science Education, Dublin, Ireland, July 8 – 13, 2022, Volume 1, pp. 414–420, ACM, 2022.
URL https://doi.org/10.1145/3502718.3524801

In this demo, we present pseuCo Book, a truly interactive textbook experience designed to help teachers and students alike. In pseuCo Book, interactive demonstrations and exercises are interwoven with traditional textual elements. Its technical foundation, the Hybrid Document Framework, is a toolset that makes authoring interactive textbooks as easy as possible. PseuCo Book contains three chapters: The first one, covering Milner's Calculus of Communicating systems, is built around an interactive editor for CCS semantics derivations. The second chapter, teaching notions of equality for concurrent processes, features custom-built exercises covering proofs and algorithms around trace equality, bisimilarity, and observation congruence. The third chapter, which covers practical concurrent programming in a minimal, academic programming language called pseuCo, is built around a set of verification technologies that allow deep inspection of the concurrency-related features of pseuCo programs, enabling fast autograding of user-submitted solutions to programming tasks. PseuCo Book has been used extensively as part of the Concurrent Programming lecture at Saarland University. A comprehensive user study, run as part of the course, demonstrates that pseuCo Book is both well-received by students and has a measurable, positive impact on student performance.

4.2 FLACI – Formal Languages, Automata, Compilers and Interpreters

Michael Hielscher (Pädagogische Hochschule Schwyz, CH)

Main reference Michael Hielscher, Christian Wagenknecht: "FLACI – Eine Lernumgebung für theoretische Informatik, Informatik für alle, 18. GI-Fachtagung Informatik und Schule", INFOS 2019: Page 211-220
URL https://flaci.com
Main reference Christian Wagenknecht, Michael Hielscher: "Formale Sprachen, abstrakte Automaten und Compiler: Lehr- und Arbeitsbuch mit FLACI für Grundstudium und Fortbildung", Springer Nature, 2022.
URL https://doi.org/10.1007/978-3-658-36853-1

I gave a brief demo session on FLACI, a web-based system designed for working with formal languages, context-free grammars, and automata. FLACI simplifies the application of these theories to compiler construction, making it accessible even at the high school level. The system allows users to visually construct and simulate automata, derivations, and compiler processes. The goal is to help students learn formal foundations while they work towards translating their own simple language into visual or acoustic output using a compiler they generate from a formal definition.

4.3 TeachingBook

Norbert Hundeshagen (Universität Kassel, DE)

Joint work of Norbert Hundeshagen, Maurice Herwig, John Hundhausen

The TeachingBook (TB) is a prototype of a web-based platform to create interactive learning materials in a cell-based Jupyter-Notebook-like environment. Its main feature is the flexibility in creating content for lectures such as interactive scripts or exercise sheets by simply arranging cells of different types (see attached screenshot). Currently, several cell-types are supported. Besides markdown cells to provide LateX content and quizzes, also cells are available to foster the learning of topics in the realm of formal foundations in computer science (regular languages, reductions, ...). Furthermore, TB is designed to ease the integration of existing tools either as iFrames or natively as front-end components. Several extensions of the TeachingBook are currently under development in student projects and a preliminary usability study has been conducted in a lecture on computability theory. It is planned that a publication will be ready for next year.

4.4 DiMo

Martin Lange (Universität Kassel, DE) and Norbert Hundeshagen (Universität Kassel, DE)

Joint work of Maurice Herwig, Norbert Hundeshagen, John Hundhausen, Stefan Kablowski, Martin Lange, Georg Siebert
Main reference Maurice Herwig, Norbert Hundeshagen, John Hundhausen, Stefan Kablowski, Martin Lange: "Problem-Specific Visual Feedback in Discrete Modelling", in Proc. of the DELFI 2024 – Die 22. Fachtagung Bildungstechnologien der Gesellschaft für Informatik e.V., DELFI 2024, Fulda, Germany, September 9-11, 2024, LNI, Vol. P-356, Gesellschaft für Informatik e.V., 2024.
URL https://doi.org/10.18420/DELFI2024_08

DiMo, short for *Discrete Modelling*, is a tool that supports learning of skills to use propositional logic as a backbone for general problem solving. Typical exercises in this area ask for the construction of propositional formulas depending on problem instance, A good example is: write a propositional formula Φ_n for $n \geq 1$ that is satisfiable iff the n-queens problem has a solution, i.e. it is possible to place n queens on a chessboard with no two of them sharing a row, a column or a diagonal line. Another example is: write a formula Φ_G for any undirected graph G that is satisfiable iff G is 3-colourable.

DiMo provides a language that is reminiscent of simple imperative programming languages in order to specify formulas. For example, $\bigvee_{i=0}^{n-1} D(i,0) \wedge \bigwedge_{\substack{j=0 \\ j\neq i}}^{n-1} \neg D(j,0)$ would be written as

```
FORSOME i: {0,..,n-1}. D(i,0) & FORALL j: {0,..,n-1} \ {i}. -D(j,0)
```

DiMo translates formulas in this formal language automatically into the mathematical form above so that students see the connection to the way formulas are presented in lectures etc.

DiMo's crown feature is a programming language with for-loops and propositions as data types. It is supposed to be used by teachers in order to write programs that turn propositional evaluations into any graphical form, for instance in HTML. DiMO then executes these programs after satisfiability checks on the students' formulas. This way, it is possible to check correctness of the formulas graphically, for instance by depicting the placement of queens on a chessboard.

DiMo is publically available to try out via a webinterface, located at `https://dumbarton.tifm.cs.uni-kassel.de/`. Further and more detailed information can be found in two papers on the technical aspects of DiMo [2] and on the graphical feedback interface [1].

References

1 M. Herwig, N. Hundeshagen, J. Hundhausen, S. Kablowski, and M. Lange. Problem-specific visual feedback in discrete modelling. In *Proc. 21. Fachtagung Bildungstechnologien der GI e.V., Delfi'24*, LNI, 2024. To appear.

2 N. Hundeshagen, M. Lange, and G. Siebert. Dimo – discrete modelling using propositional logic. In *Proc. 24th Int. Conf. on Theory and Applications of Satisfiability Testing, SAT'21*, number 12831 in LNCS, pages 242–250. Springer, 2021.

4.5 Proof Blocks

Seth Poulsen (Utah State University, US)

Joint work of Mahesh Viswanathan, Geoffrey L. Herman, Matthew West
Main reference Seth Poulsen, Mahesh Viswanathan, Geoffrey L. Herman, Matthew West: "Proof Blocks: Autogradable Scaffolding Activities for Learning to Write Proofs", in Proc. of the ITiCSE 2022: Innovation and Technology in Computer Science Education, Dublin, Ireland, July 8 – 13, 2022, Volume 1, pp. 428–434, ACM, 2022.
URL https://doi.org/10.1145/3502718.3524774

In this software tool paper we present Proof Blocks, a tool which enables students to construct mathematical proofs by dragging and dropping prewritten proof lines into the correct order. We present both implementation details of the tool, as well as a rich reflection on our experiences using the tool in courses with hundreds of students. Proof Blocks problems can be graded completely automatically, enabling students to receive rapid feedback. When writing a problem, the instructor specifies the dependency graph of the lines of the proof, so that any correct arrangement of the lines can receive full credit. This innovation can improve assessment tools by increasing the types of questions we can ask students about proofs, and can give greater access to proof knowledge by increasing the amount that students can learn on their own with the help of a computer.

4.6 JFLAP (Java Formal Language and Automata Package)

Susan Rodger (Duke University – Durham, US)

Main reference Susan H. Rodger, Thomas W. Finley: "JFLAP: An Interactive Formal Languages and Automata Package". Jones and Bartlett Publishers, Inc., Sudbury, Massachussetts, USA, 2006.
URL https://www.jflap.org

We have been designing the JFLAP tool now for about thirty years. JFLAP allows one to experiment with finite state automata, pushdown automata, Turing machines, all kinds of grammars, and L-systems. In addition one can experiment with algorithms and proofs such as converting a nondeterministic finite automaton (NFA) to a deterministic finite automaton (DFA), a DFA to a regular expression, a context-free grammar (CFG) to an nondeterministic pushdown automaton (NPDA), or explore examples with the Pumping Lemma. We presented a demo on the JFLAP tool to show how to take a CFG to build an LR(1) parse table, and then parse a string from that CFG using the table. First, we loaded the following CFG: S -> aSb, S -> aBb, B -> cB, B -> b. We then showed how to convert that CFG to an equivalent NPDA that corresponds to the LR(1) parsing algorithm. We then traced the string "aacbbb" showing how the NPDA is nondeterministic and that the string is accepted. Next, we started with the same grammar and showed how to use JFLAP to construct the LR(1) parse table from the CFG. First we calculated FIRST and FOLLOW sets. Then we built a DFA that models how the LR(1) parsing stack works. Each state in the DFA has marked rules associated with it, indicating how much of the rule has been processed. Using the DFA, we constructed the equivalent LR(1) parse table. The table showed no conflicts. Finally, we parsed the same string "aacbbb", seeing which entry is being executed in the table and which symbols are on the stack, showing the string is accepted.

We have included two figures from this example. One of the figures shows the grammar on the left and on the right shows the corresponding FIRST set, FOLLOW set, DFA with marked rules associated with each state, and the corresponding LR(1) parse table. The other figure shows one step in the parsing of the string "aacbbb$" (we add $ to the right end, an end of string marker). The top left shows the LR(1) parse table, highlighting row 0 (state 0 in the DFA) and column S with an entry of 1, meaning the S and 1 were just pushed onto the parsing stack on top of the 0. The top right shows the input remaining, only "$", and the current stack contents of 1S0 (with 1 the top of the stack). The grammar is shown in the bottom left, highlighting the rule that is currently being reduced. The parse tree being built is shown in the bottom right, and is now complete.

We like to show applications with the theory! `www.jflap.org`.

References

1 Susan H. Rodger and Thomas W. Finley. *JFLAP: An Interactive Formal Languages and Automata Package*. Jones and Bartlett Publishers, Inc., Sudbury, Massachussetts, USA, 2006.

2 Susan H. Rodger, Eric N. Wiebe, Kyung Min Lee, Chris Morgan, Kareem Omar, Jonathan Su, *Increasing engagement in automata theory with JFLAP*, Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2009, Chattanooga, TN, USA, pages 403-407, March 4-7, 2009.

4.7 Teaching Formal Foundations of Computer Science with Iltis

Marko Schmellenkamp (Ruhr-Universität Bochum, DE)

Joint work of Marko Schmellenkamp, Fabian Vehlken, Thomas Zeume
Main reference Marko Schmellenkamp, Fabian Vehlken, Thomas Zeume: "Teaching formal foundations of computer science with Iltis". Educational Column of the Bulletin of EATCS 14, 2024.
URL http://bulletin.eatcs.org/index.php/beatcs/article/download/797/842

Iltis is a web-based educational support system for the formal foundations of computer science. In the field of logic, Iltis offers exercises for many typical reasoning workflows for propositional, modal, and first-order logic. This includes exercises for modelling a scenario with formulas, transforming these formulas into appropriate normal forms, and testing these formulas for satisfiability using different methods. In the field of formal languages, Iltis includes exercises on regular expressions, finite automata, context-free grammars, and push-down automata. In the field of computational and complexity theory, Iltis supports students with exercises for working with graph problems and designing graph-based reductions. Core objectives in the development of Iltis were to facilitate the straightforward incorporation of new educational tasks, the sequencing of these individual tasks into multi-step exercises, and the cascading of sophisticated feedback mechanisms. Iltis is regularly used in courses with more than 300 students. We welcome all readers to try Iltis at `https://iltis.cs.tu-dortmund.de`.

4.8 Automata Tutor

Maximilian Weininger (IST Austria – Klosterneuburg, AT)

Joint work of Loris D'Antoni, Martin Helfrich, Jan Kretinsky, Emanuel Ramneantu, Maximilian Weininger
Main reference Loris D'Antoni, Martin Helfrich, Jan Kretinsky, Emanuel Ramneantu, and Maximilian Weininger. Automata tutor v3. In CAV (2), volume 12225 of Lecture Notes in Computer Science, pages 3–14. Springer, 2020.
URL 10.1007/978-3-030-53291-8_1

I shortly demonstrate the teaching-support system Automata Tutor [1], showing the ability to generate feedback quickly, in the form of counter-examples. The three core messages, relating to the communities present at the seminar, are:
1. As TCS-teacher (working in foundations), you might want to use Automata Tutor.
2. As an educations researcher, you might be interested in the data Automata Tutor generates.
3. As a tool-developer, you might be interested in exchanging experiences on technical challenges.

References

1 Loris D'Antoni, Martin Helfrich, Jan Kretínský, Emanuel Ramneantu, and Maximilian Weininger. Automata tutor v3. In *CAV (2)*, volume 12225 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2020.

4.9 Karp

Chenhao Zhang (Northwestern University – Evanston, US)

In CS theory courses, NP reductions are a notorious source of pain for students and instructors alike. Invariably, students use pen and paper to write down reductions that work in many but not all cases. When instructors observe that a student's reduction deviates from the expected one, they have to manually compute a counterexample that exposes the mistake. We introduce Karp, a language for programming and testing NP reductions.

5 Working groups

5.1 Formal foundations in schools

Erik Barendsen (Radboud University – Nijmegen, NL & Open University – Heerlen, NL), Rodrigo Duran (Federal Institute of Mato Grosso do Sul, BR), Judith Gal-Ezer (The Open University of Israel – Ra'anana, IL), Sandra Kiefer (University of Oxford, GB), Dennis Komm (ETH Zürich, CH), Tilman Michaeli (TU München, DE), Liat Peterfreund (The Hebrew University of Jerusalem, IL), Ramaswamy Ramanujam (Azim Premji University – Bengaluru, IN), Susan Rodger (Duke University – Durham, US), Florian Schmalstieg (Ruhr-Universität Bochum, DE), R. Benjamin Shapiro (University of Washington – Seattle, US), and John Slaney (Australian National University – Canberra, AU)

The members exchanged the current state of affairs concerning the role of formal methods in K-12 curricula and teacher education in Switzerland, Israel, The Netherlands, Germany, Brazil, the United States, India, and Australia.

Formal methods include algorithmic reasoning, automata, formal languages, and logic. The group discussed possible advantages and hindrances related to teaching aspects of formal methods in K-12. It was useful in distinguishing societal needs, CS as a discipline, curricular content, and pedagogies.

Potential themes for follow-up activities of the group are:

- Why are formal foundations important in K-12? What should be in a dedicated subject and what is important for "everyone" (or just some group) and why? Which elements are critical for compulsory education? (The situation for CS seems to be different from, eg, physics.) An opinion article (column) could be a result.
- A research study on teachers' perspectives on the "formal foundations": content knowledge, teacher beliefs, PCK, and a cross-cultural inventory of teaching practices w.r.t. formal methods.
- A research study on students' point of view on Computing in society and which elements are based on formal foundations of CS.
- Identifying and utilizing the opportunities w.r.t. 'reasoning' at primary and upper primary levels and increasing the logic content across secondary school curricula.
- Cross-cultural studies on specific aspects of formal methods, taking the respective cultural contexts into account.

5.2 Using LLMs in the process of learning how to perform reductions – An idea

Michael Hielscher (Pädagogische Hochschule Schwyz, CH), Norbert Hundeshagen (Universität Kassel, DE), Johan Jeuring (Utrecht University, NL), Martin Lange (Universität Kassel, DE), and Tilman Michaeli (TU München, DE)

Reductions are a key method in computability and complexity theory to identify unsolvable or intractable problems. Therefore, learning how to reduce one problem onto another is part of every standard curriculum in such courses. We here report on ideas discussed in a working group with the goal of designing an intervention to help students solving exercises on reductions and thus, foster a better understanding of this rather difficult topic. Moreover, our focus is on tool-supported learning by using generative AI. From a learners perspective, a reduction task between two problems A and B essentially can be seen as a programming exercise, where students need to write an algorithm that converts instances of A into instances of B, such that a solution for B can be used to solve A. Our idea of an intervention aims at this programming part of reductions. More specifically, an educational tool should be designed to help students plan the solution to a given reduction task. That is, before implementing a reduction between two given problems, students use our intended tool to describe in natural language how their algorithm should convert problem instances. This description is then used as a prompt for an LLM (e.g. ChatGPT 4.0) with the task of actually producing an implementation, e.g. in Python. The latter allows feedback to be computed in two ways. First, the AI-generated algorithm can be tested for correctness of the solution (and the plan), by running it on positive and negative examples of problem A and providing its answer to the students. In the case of incorrect solution attempts, i.e. positive instances of A are mapped to negative instances of B, or vice versa, further feedback on how to improve the plan could also be provided by the LLM. We suspect that a carefully designed prompt that includes problem-specific information and meta-information on planning can be used for the latter. As the approach described above is intended to be a first idea on the subject, it is clear that a number of issues need to be addressed before the intervention is actually implemented. Among others, it is unclear how planning of reductions in natural language is perceived by students, and how the quality of a plan can be measured. Furthermore, the quality of LLM-output needs to be investigated, especially, if reduction tasks are complex and/or the plans of students are incomplete or ambiguous.

5.3 Brainstorm on Recording Teachers' Observations of Errors for Formal Foundations of CS

Daphne Miedema (University of Amsterdam, NL), Norbert Hundeshagen (Universität Kassel, DE), Martin Lange (Universität Kassel, DE), Alexandra Mendes (University of Porto, PT & INESC TEC – Porto, PT), Sophie Pinchinat (University of Rennes, FR), Anne Remke (Universität Münster, DE), Vaishnavi Sundararajan (Indian Institute of Technology – New Delhi, IN), Maximilian Weininger (IST Austria – Klosterneuburg, AT), and Thomas Zeume (Ruhr-Universität Bochum, DE)

There is not much existing research on misconceptions for FF and TCS. Therefore, we propose to build a base to build such work on. The format will be to gather teachers' observations on students' struggles in courses such as Theory of Computer Science. We do this by creating a form asking for descriptions and example questions that these mistakes occur on. These are collected and organized in a document, which could be written up for a discussion paper. In a later stage, we could follow-up on this research by studying error prevalence and identifying underlying misconceptions.

Additionally, the document could be of use for those building Teaching-Support Systems, as they can gain insight into problems teachers or students typically run into. They could translate this into additional exercises or hints within the TSS.

5.4 Automated Proof Feedback in the Wild

Seth Poulsen (Utah State University, US), Erik Barendsen (Radboud University – Nijmegen, NL & Open University – Heerlen, NL), Felix Freiberger (Universität des Saarlandes – Saarbrücken, DE), Dennis Komm (ETH Zürich, CH), and Thomas Zeume (Ruhr-Universität Bochum, DE)

We discussed the feasibility of doing a large scale study of using teaching support systems to help students learn how to write reduction proofs as part of a computing theory course. Such a study would involve helping 2nd year computer science students learn reduction proofs by using Proof Blocks, Faded Proof Blocks, and feedback from large language models fine-tuned on mathematical proof data. Student proofs would be analyzed through the lens of the following rubric: (1) Overall proof structure is there, (2) Reduction Function is clearly defined, (3) Proving the computability/complexity, (4) Proving the Reduction Property (both directions).

5.5 What makes translating formal languages in set notation to context free grammar difficult?

Florian Schmalstieg (Ruhr-Universität Bochum, DE), Rodrigo Duran (Federal Institute of Mato Grosso do Sul, BR), Liat Peterfreund (The Hebrew University of Jerusalem, IL), Jakob Schwerter (TU Dortmund, DE), John Slaney (Australian National University – Canberra, AU), and Maximilian Weininger (IST Austria – Klosterneuburg, AT)

The conversion of formal languages (e.g. set notation to context free grammar or PDA) is a difficult problem for students to tackle. But some tasks are more difficult than others. This leads to the question: What properties make a conversion task difficult? We call such properties *difficulty-generating factors*. They can help to understand why a specific task might be difficult and to adapt tasks for different needs.

But in which ways can such factors be found systematically? And how can hypothesized factors be verified?

To this end we propose a mixture of qualitative and quantitative research designs. First we want to assess possible difficulty-generating factors. For this we will be doing expert and novice interviews to find out, which aspects of such a task they look at. The tasks for these interviews are informed by existing student performance data.

We will then try to verify the found factors in a randomized control trial study by systematically manipulating the aspects and using Rasch-analysis to compare the result with the expected ranking.

5.6 Can teaching support systems affect students self-regulated learning behavior?

Jakob Schwerter (TU Dortmund, DE), Michael Hielscher (Pädagogische Hochschule Schwyz, CH), Daphne Miedema (University of Amsterdam, NL), Marko Schmellenkamp (Ruhr-Universität Bochum, DE), Jan Vahrenhold (Universität Münster, DE), and Thomas Zeume (Ruhr-Universität Bochum, DE)

Self-regulated learning (SLR) has been shown to be positively correlated to academic succes (Zimmerman et al. 1992, Greene et al. 2021). In this breakout group, we considered the question of whether – and if so how – data from digital learning environments (DLEs) can be used to gain insights into the state of a self-regulated learning process the learner is currently in. For the sake of conciseness, we decided to focus on the ILTIS DLE while reminding ourselves of the fact that there is a broad spectrum of such systems. Building on Zimmerman's conceptualization (2008), we used the SRL cycle consisting of "Forethought Phase" – "Performance Phase" – "Self-Reflection Phase". Going through each of the phases of this cycle, we started with the question which SRL-related variables can be generated or derived by the system and what gaps currently exist. We hypothesize that using these variables, we can create an individual learner profile which then in turn can ultimately be used to suggest tailored interventions to help learners improve their study and learning behavior.

References

1 Greene, J. A., Plumley, R. D., Urban, C. J., Bernacki, M. L., Gates, K. M., Hogan, K. A., Demetriou, C., & Panter, A. T. (2021). Modeling temporal self-regulatory processing in a higher education biology course. Learning and Instruction, 72, 101201. https://doi.org/10.1016/j.learninstruc.2019.04.002

2 Zimmerman, B. J. (2008). Investigating Self-Regulation and Motivation: Historical Background, Methodological Developments, and Future Prospects. American Educational Research Journal, 45(1), 166-183. doi.org/10.3102/0002831207312909

3 Zimmerman, B. J., Bandura, A., & Martinez-Pons, M. (1992). Self-Motivation for Academic Attainment: The Role of Self-Efficacy Beliefs and Personal Goal Setting. American Educational Research Journal, 29(3), 663-676. doi.org/10.3102/00028312029003663

5.7 Tool building: Experience exchange

Maximilian Weininger (IST Austria – Klosterneuburg, AT), Tiffany Barnes (North Carolina State University – Raleigh, US), Felix Freiberger (Universität des Saarlandes – Saarbrücken, DE), Michael Goedicke (Universität Duisburg – Essen, DE), Michael Hielscher (Pädagogische Hochschule Schwyz, CH), Dennis Komm (ETH Zürich, CH), Daphne Miedema (University of Amsterdam, NL), Seth Poulsen (Utah State University, US), Susan Rodger (Duke University – Durham, US), Florian Schmalstieg (Ruhr-Universität Bochum, DE), Marko Schmellenkamp (Ruhr-Universität Bochum, DE), Vaishnavi Sundararajan (Indian Institute of Technology – New Delhi, IN), and Thomas Zeume (Ruhr-Universität Bochum, DE)

In this working group, we discussed best practices for building and maintaining teaching support systems. Below, we summarize the key takeaways:

- Maintainability: This is a crucial problem for teaching-support tools, and often a cause for them to not stay available after some years. A key problem is that they are usually developed by undergraduate or PhD-students and thus not maintained after some time. We identified several ways to mitigate this: i) avoid using the "fanciest" new technology, ii) limit the inclusion of other libraries and dependencies, iii) host a web app and do not require local installation for users, iv) modularize solutions, v) have a plan for how the tool will be maintained (which requires permanent staff to be involved).
  We also discussed the idea of building on top of existing systems (like ILTIS or moodle) which can take care of processes like user management, course management and task management. On the one hand, this simplifies the code of the actual teaching-support system; on the other hand, it introduces a dependency to the other system and thus a potential maintainability problem in the future (keeping versions up to date, what if the existing system is discountinued).
- Scalability: Several tools run on a cluster of virtual machines, which has only limited scalability. Alternative solutions include using systems like Apache Kafka, distributed caching or client-side processing. The latter effectively eliminates scalability problems, however at a cost: Firstly, collecting performance data requires is less immediate, but this can be fixed as has been done for PseuCo. Secondly, client-side grading introduces a vulnerability, as students can reverse-engineer the grader and obtain the sample solution.

- Funding: Acquiring the funding to host a teaching-support system is a key problem, in particular as maintenance and scaling are expensive, but often only development of new features is funded. The following possible solutions were suggested: using teaching improvement funds or asking universities using the tool to pay for a student assistant (since they anyway pay students to grade assignments, they are saving money this way).
- Software-development best practices: Proper practices like writing requirements, performing code-reviews and unit tests should be applied. However, this requires knowledge of software engineering as well as the funding and time to adhere to these practices.
- Data-collection: For education purposes, it can be very useful to not only have the final result of a task, but also a trace of all the actions the student performed when solving the task. Moreover, for analysis, it is very useful if the tool offers a “replay” feature. This allows to “execute” the trace and for every action of the student see the state of the system like the student did.

Participants

- Efthimia Aivaloglou
TU Delft, NL
- Erik Barendsen
Radboud University – Nijmegen, NL & Open University – Heerlen, NL
- Tiffany Barnes
North Carolina State University – Raleigh, US
- Rodrigo Duran
Federal Institute of Mato Grosso do Sul, BR
- Felix Freiberger
Universität des Saarlandes – Saarbrücken, DE
- Judith Gal-Ezer
The Open University of Israel – Ra'anana, IL
- Michael Goedicke
Universität Duisburg – Essen, DE
- Michael Hielscher
Pädagogische Hochschule Schwyz, CH
- Norbert Hundeshagen
Universität Kassel, DE
- Johan Jeuring
Utrecht University, NL
- Sandra Kiefer
University of Oxford, GB
- Dennis Komm
ETH Zürich, CH
- Shriram Krishnamurthi
Brown University – Providence, US
- Martin Lange
Universität Kassel, DE
- Alexandra Mendes
University of Porto, PT & INESC TEC – Porto, PT
- Tilman Michaeli
TU München, DE
- Daphne Miedema
University of Amsterdam, NL
- Liat Peterfreund
The Hebrew University of Jerusalem, IL
- Sophie Pinchinat
University of Rennes, FR
- Seth Poulsen
Utah State University, US
- Ramaswamy Ramanujam
Azim Premji University – Bengaluru, IN
- Anne Remke
Universität Münster, DE
- Susan Rodger
Duke University – Durham, US
- Florian Schmalstieg
Ruhr-Universität Bochum, DE
- Marko Schmellenkamp
Ruhr-Universität Bochum, DE
- Francois Schwarzentruber
IRISA – ENS Rennes, FR
- Thomas Schwentick
TU Dortmund, DE
- Jakob Schwerter
TU Dortmund, DE
- R. Benjamin Shapiro
University of Washington – Seattle, US
- John Slaney
Australian National University – Canberra, AU
- Vaishnavi Sundararajan
Indian Institute of Technology – New Delhi, IN
- Jan Vahrenhold
Universität Münster, DE
- Maximilian Weininger
IST Austria – Klosterneuburg, AT
- Thomas Zeume
Ruhr-Universität Bochum, DE
- Chenhao Zhang
Northwestern University – Evanston, US