OnlineProver: Experience with a Visualisation Tool for **Teaching Formal Proofs**

Ján Perháč * †

Samuel Novotný

Sergej Chodarev

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Košice, Slovakia

{jan.perhac,samuel.novotny,sergej.chodarev}@tuke.sk

Joachim Tilsted Kristensen

Lars Tveito

Department of Informatics, University of Oslo, Oslo, Norway

{joachkr,larstvei}@ifi.uio.no

Oleks Shturmov

Michael Kirkedal Thomsen

Department of Informatics, University of Oslo, Oslo, Norway Department of Computer Science, University of Copenhagen, Copenhagen, Denmark {michakt,olekss}@ifi.uio.no

OnlineProver is an interactive proof assistant tailored for the educational setting. Its main features include a user-friendly interface for editing and checking proofs. The user interface provides feedback directly within the derivation, based on error messages from a proof-checking web service. A basic philosophy of the tool is that it should aid the student while still ensuring that the students construct the proofs as if they were working on paper.

We gathered feedback on the tool through a questionnaire, and we conducted an intervention to assess its effectiveness for students in a classroom setting, alongside an evaluation of technical aspects. The initial intervention showed that students were satisfied with using OnlineProver as part of their coursework, providing initial confirmation of the learning approach behind it. This gives clear directions for future developments, with the potential to find and evaluate how OnlineProver can improve the teaching of natural deduction.

Introduction 1

When teaching theoretical computer science courses (e.g., introduction to first-order logic), the focus is often not on programming; instead, we aim to teach an understanding of discrete abstractions and their properties through proofs. For this reason, spending several weeks of the course on introducing a theorem prover can be counterproductive, as it diverts the attention of students from the foundational concepts. In our project¹, we propose a tool, OnlineProver², which provides exercises for Gentzen-style derivation proofs over user-definable deduction systems.

Iceland Liechtenstein Norway The authors thanks EEA and Norway Grants for support of this work under initiative no. FBR-

"Working together for a green, competitive and inclusive Europe." **Norway** grants

grants https://www.eeagrants.sk/

[†]The author thanks the "European Research Network on Formal Proofs" under EU COST Action CA20111 for support of this work.

https://onlineprover.github.io/

²http://onlineprover.com/

J. Narboux, W. Neuper and P. Quaresma (Eds.): Theorem Proving Components for Educational Software 2024 (ThEdu'24) EPTCS 419, 2025, pp. 55-74, doi:10.4204/EPTCS.419.4

© J. Perháč et al. This work is licensed under the Creative Commons Attribution License. The main objective is to build an educational tool that include some of the capabilities of computer-aided theorem proving, without requiring students to learn a new syntax or dedicating several lectures to introducing the tools. Our initial setup involves a tool for which the teacher must learn a simple domain-specific language (DSL) for specifying derivation systems and exercises. However, students use the tool solely to solve exercises provided by the teacher through a web interface designed for this purpose. The interface is capable of checking students' derivations and providing feedback directly within the derivation where it is needed.

Computer science (CS) students typically follow several programming courses before delving into theoretical computer science, during which they gain extensive experience through a trial-and-error approach [25]. On the other hand, theoretical computer science courses all require mathematical prerequisites, which CS students frequently struggle with. Initial research [18] and personal experiences [19] suggest that the pedagogical approaches effective for teaching CS students differ from those tailored to mathematics students. Similar findings are observed in the field of physics education [22].

Building upon this foundation, we have developed *onlineprover*, a tool that aids CS students by facilitating the exercise of mathematical rigour while offering a taste of trial-and-error exploration. Students must specify all derivations and type all rule constructs of the proofs; there is no interactive rule selection. We want the tool to closely resemble classroom teaching with pen and paper. The learning objectives that OnlineProver could support in a course include:

• Skills in deciding and proving formal properties of logical formulas (e.g., satisfiability, validity, implication, and equivalence) through natural deduction arguments.

Following the tradition in CS education [6], we therefore aim to foster deep reflection through meticulous tool design, rather than solely promoting problem-solving through trial-and-error. Drawing from experiences in introductory programming, we acknowledge that while students appreciate the simplicity of block-based frameworks (e.g., Scratch), they may find them lacking in authenticity [25]. Research on tools for automated assessment (auto-graders) [1, 5, 12], often used for giving quick and direct feedback on programming tasks, has also shown that this can shift the students' focus away from problem-solving. Instead, students tend to optimize their efforts towards passing specific tests.

In this work, we have introduced the OnlineProver in at the course "Foundations of Computing - Discrete Mathematics" [4] with 165 students, at the IT University of Copenhagen, Denmark. A first and simple intervention were presented in [14] The following is a description of our first experience use of mechanical reasoning systems for teaching. First, we will give the research questions on the work (Section 2) followed by an outline for related work (Section 3). Then, in Section 4 we outline the design of OnlineProver, after which Section 5 will exemplify the experience of the students. In Section 6 we present the intervention and the results. These will be discussed in Section 7 and finally in Section 8 we conclude. The online version of OnlineProver tool along with all exercises that is used in this experiment, can be found at http://onlineprover.com/.

2 Research Questions

To guide our research, we formulated the following research questions focusing on the usage, strategies, and perceived effectiveness of OnlineProver as a teaching and learning tool for formal proofs.

- RQ1: Is our tool used as a proof assistant or a proof checker?
- RQ2: How do students use trial-and-error strategies with OnlineProver to overcome difficulties in solving proofs?

- RQ3: Do students perceive the tool's feedback useful?
- RQ4: Do students consider learning proofs using OnlineProver more effective than learning proofs by pen and paper?

We will answer these questions in Section 7.

3 Related Work

Although traditional proof assistants such as Coq, Agda, and Isabelle use a programming style, we do not find them well-suited for introductory courses. It takes a significant amount of time for students to learn the underlying system's languages to be able to start working on formal proofs. Several other tools have been developed with a focus on a domain-specific framework and implement a trial-and-error approach, giving students less room for reflection. Such tools are dedicated to various frameworks, such as proof trees for Hoare logic [13], sequent calculus [7], Gentzen-style proofs [9], Tableau [23, 24], and λ -calculus [8]. While these tools often include automation and rule limitations, they are generally designed to encourage trial-and-error, which can diminish the opportunity for deep reflection.

For example, the work by [13] presents an online tool for Hoare logic that includes LaTeX export, as well as automation and rule limitations, generally promoting a trial-and-error approach. Similarly, [2, 17, 21] discuss tools relevant to natural deduction, though some lack comprehensive references. In the context of sequent calculus, [7] provides a well-written paper with a focus on first-order logic. The Gentzen-style approach is discussed in [9], while Tableau methods are explored by [24] and [23]. Lastly, the λ -calculus framework is presented in [8]. However, these tools are often designed to be limiting for students, exhibiting similar issues to block-based programming frameworks. They might simplify initial learning but at the cost of limiting deeper understanding and flexibility. Closer to our approach is the work by [15], which in a more general and flexible framework implements support for Fitch-style proofs. Also relevant is [26], which, though it focuses on elementary symbolic logic, takes a teaching approach aligned with our educational goals.

4 Tool Design

OnlineProver consists of three modular components: a proof engine, a web server, and a web-based editor for Gentzen-style derivation trees. The proof engine checks derivations, provides feedback, and supports alternative calculi implementations. In this paper, the proof engine used is based on natural deduction from *The Logic Manual* by Volker Halbach [10]. The web server acts as an intermediary between the user-facing editor and the proof engine, providing endpoints for serving a front page, a collection of exercises, and requesting feedback on exercises. The editor allows users to select exercises, where each exercise contains a description of the task and along with one or more editable proof trees that comprise their answer. Derivation trees are edited by adding or deleting nodes, where each node is a plain text field. Exercises can be checked, which submits the derivations of the exercise to be annotated with feedback provided by the proof engine.

Figure 1 illustrates how the components of OnlineProver communicate. A user accesses the editor by visiting a URL, which results in an HTTP GET request that responds with the exercise, along with the sourcecode for the editor itself. The user can then begin solving the exercise by editing the derivation tree. A user may at any time *check* their solution, which is materialized as a HTTP POST request, containing a JSON encoding of the derivation tree. The server consults the proof engine, which returns

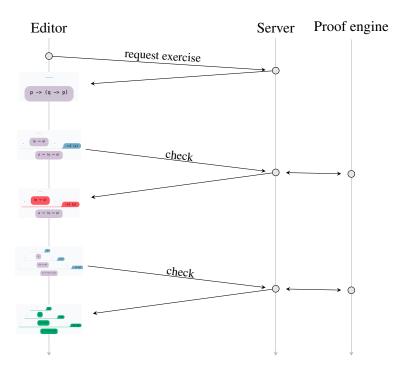


Figure 1: An example of interacting with OnlineProver.

a derivation tree annotated with feedback. The user may continue to edit and check their proof in a feedback-loop, similar to how a programmer interacts with a compiler, until the proof is finally verified by the proof engine.

The server and proof engine are written in Haskell, while the editor is implemented in ClojureScript. Derivation tree states are stored in LocalStorage, ensuring that the server and proof engine remain completely stateless. All user interaction events, along with timestamps, are recorded, making it possible to replay and reconstruct every state of the user's derivation. We also support both data export and import: exports can be used for analysis (see Section 6.4), while imports enable users to continue their work on a different machine.

5 Teaching with OnlineProver: Motivating example

In this section, we present a simple example of the use of OnlineProver from a student's perspective. For this purpose, we have deployed a version of the OnlineProver system with a set of exercises for Gentzen-style proofs using natural deduction in propositional logic and first-order logic. The exercises consist of lists of incomplete derivations, and a student must complete them in an empty context.

We have implemented exercises on natural deduction based on the notation used in *The Logic Manual* [10]. This style uses an implicit context for assumptions. Therefore, when applying rules that introduce assumptions, one must name each assumption with a tag (a *lowercase Latin letter*). In the graphical user interface, an exercise in this logic appears as a formula, above which the corresponding derivation must be constructed. Table 1 summarizes the notation used by OnlineProver compared to the standard symbols of logic.

Logic Symbol	OnlineProver Notation
\forall	forall
3	exists
	!
^	/\
V	\/
\rightarrow	->
	False
T	True

Table 1: Mapping of Logic Symbols to OnlineProver Notation.

Exercise 6 - d

Prove the validity of the following formula in predicate logic



Figure 2: Exercise 6-d from onlineprover.com.

As a motivating example, we prove the following formula in OnlineProver:

$$(\forall x.(A(x) \land B(x))) \to \exists x.B(x) \tag{1}$$

which can be translated into the OnlineProver notation as:

(forall x .
$$(A(x) / B(x))$$
) -> exists x . $B(x)$

Exercise 6-d in OnlineProver contains the formula (1), as depicted in Figure 2. The button with a line symbol above the formula adds a new line on top of it together with the input box for the rule name. The next step is to input the rule name (with a tag for the assumption name if needed), and, if necessary, add more input boxes for premises. OnlineProver is developed as a proof assistant, so it can check if the (partial) derivation is correct. If not, it will provide feedback via the Check button. For example, Figure 3 shows a situation where an incorrect rule was applied.

Input boxes turn red, and an error message is displayed. Once a correct (partial) derivation is provided, the correctly filled input boxes turn green. Figure 4 depicts a situation where two steps of the derivation are correct, but the proof is still incomplete.

Finally, the proof is complete when all branches of the proof tree are closed by an assumption. To apply an assumption, the user must input its tag into the rule name box. The proof is considered complete when all input fields turn green. Figure 5 shows the entire exercise page with the closed proof.

The top panel contains the following functionalities:

• Check button: Provides feedback on the correctness of the derivation.

Exercise 6 - d

Prove the validity of the following formula in predicate logic

```
SemanticError: \bigwedge I does not apply to formulas of the form (forall x. A(x) \bigwedge B(x)) -> (exists x. B(x))

(forall x . (A(x) / \setminus B(x))) -> exists x . B(x)
```

Figure 3: An example of an incorrectly applied rule.

Exercise 6 - d

Prove the validity of the following formula in predicate logic

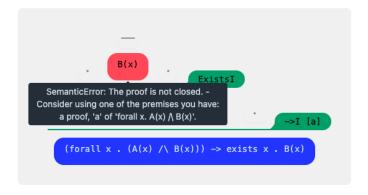


Figure 4: An example of the error with a partially written derivation.

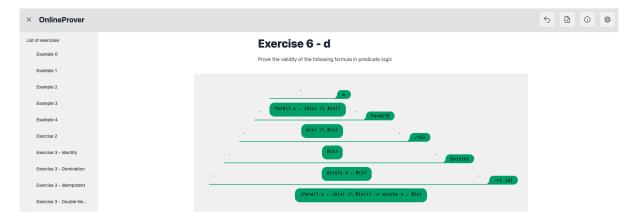


Figure 5: An examples of a closed proof.

- Undo button: Reverts one step back in the derivation.
- Info button: Displays the derivation rules on the right side of the screen.

• **Settings button**: Allows the user to delete all derivations, change the theme, and more.

6 Evaluation of Teaching Intervention with OnlineProver

The formal evaluation of OnlineProver is based on one semester - Autumn 2024, in the bachelor level first year course "Foundations of Computing - Discrete Mathematics" [4] with 165 students, at the IT University of Copenhagen.

We have decided to evaluate the deployment of OnlineProver using three methods:

- The system usability scale (SUS) [3], which is a standardized method for evaluation of software usability [16].
- We asked open questions to collect feedback on functionality, technical details, teaching, and cognitive presence.
- Data analysis of the EDN files provided by students.

In the rest of the section, we present the results of the SUS, questionnaire, and EDN file data analyses. The interpretation of these findings, including how they address the research questions and suggestions for improvement, will be thoroughly discussed in the next section, titled Discussion.

6.1 Intervention Description

As mentioned earlier, we deployed OnlineProver in the course "Foundations of Computing - Discrete Mathematics." Before OnlineProver, students used an online teaching tool called ProofWeb [11], along with the traditional pen-and-paper approach for practicing proofs and logical reasoning. This is a first-year bachelor-level course; therefore, we can assume that students have little to no prior experience with formal proofs.

Half of the course is dedicated to propositional logic, first-order logic, and formal proofs. The first three lectures focused on logic and natural deduction. OnlineProver was introduced to students during the first lecture, where the lecturer solved example exercises from slides (referred to as "Example" in OnlineProver). During the first lab session, students were tasked with solving problems up to and including Exercise 3-i in the current deployment. These exercises are direct transcripts of the ones at the end of each chapter in the corresponding reading material for each lecture. In the second lab session, students were asked to complete the remaining exercises up to Exercise 8-d. Students could receive help from the lecturer and teaching assistants during the lab sessions if they requested it.

During the third lecture, students were given an assignment to assess their understanding of the topic, requiring them to complete proofs in a pen-and-paper format. Students were asked to fill out the questionnaire during the same lecture, after reviewing the assignment text for the first assignment, but before submitting it for grading. After that, students were asked to export their data from OnlineProver in the EDN format and send it to us. We use the data for data analysis.

The questionnaire consisted of the SUS questions to calculate a standardized score and open questions for quantitative and qualitative evaluation.

6.2 System Usability Scale Score

The first part of the questionnaire consisted of the System Usability Scale (SUS) form: a standardized set of 10 questions in the Likert scale style [3], which is a widely used method for assessing system usability. A total of 55 students filled out the form during the third lecture of the course. Therefore, we can assume that students had sufficient exposure to the tool, allowing them to provide a meaningful assessment.

In this method, system usability is measured using a SUS score [20], which represents a grade on a 0–100 scale, with an average score of 68.

Based on the collected data, we calculated the **SUS score** to be **67.27**. After applying a 10% trimmed mean, the adjusted **score** is 67.45. This score corresponds to a grade of C.

We would like to note that the final SUS score could potentially be higher, as one of the questions in the SUS form might have been misinterpreted: "I needed to learn a lot of things before I could get going with this system." In this context, users might have interpreted the question as referring to their effort in learning logic and natural deduction proofs, rather than the usability of the system.

6.3 Questionnaire

As mentioned earlier, 55 students filled out the questionnaire in total. Since the questionnaire included both closed and open questions (and answering was optional), we divided the questions into the following categories:

Quantitative questions:

- How many exercises did you solve on onlineprover.com?
- How much time would you estimate that you spent on onlineprover.com?

Qualitative questions:

- Mention something that you struggled with.
- How would you typically resolve this particular struggle?

Comparison (doing proofs by hand vs OnlineProver):

- What did you find positive about doing exercises by hand compared to using OnlineProver?
- What did you find positive about doing exercises using OnlineProver compared to doing them by hand?

The last three questions are reserved for the discussion section:

- Do you have any suggestions that would improve the quality of the tool for you?
- How would this improvement help you use the tool?
- Do you have any other suggestions?

6.3.1 Quantitative Questions

We asked two control questions. The first was a closed question to estimate the approximate number of exercises solved by students. We can later validate the results through data analysis of the EDN files. All 55 students answered this question.

The chart in Figure 6 shows that 40% of students claimed to have solved about half of the exercises, while approximately 30% reported solving either less or more than half of the exercises. This distribution matches our expectation that the results would approximate a Gaussian distribution.

How many exercises did you solve at onlineprover.com? 55 odpovedí

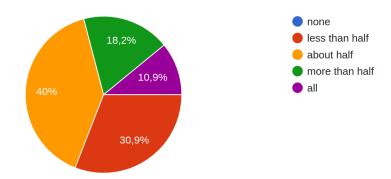


Figure 6: Question 1: Number of exercises solved on OnlineProver.

The second control question aimed to determine if students spent an appropriate amount of time using OnlineProver. This could indicate whether the tool is easy to use, well-designed, and serves its purpose effectively. We will further validate this in the discussion section based on the EDN files data analysis.

We asked the following open-ended question:

How much time would you estimate that you spent on onlineprover.com?

A total of 50 students answered this question. We categorized the responses and summarized them in Figure 7.

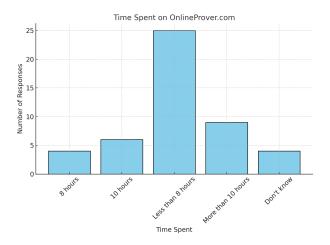


Figure 7: Question 2: Time spent on OnlineProver.

"The most common responses were 10 hours and 8 hours. The largest group of students (25) reported spending less than 8 hours on OnlineProver, while 9 students reported spending more than 10 hours, with some outliers spending 20–30 hours. Additionally, one student mentioned spending more time using OnlineProver than doing pen-and-paper proofs, while another felt they should have dedicated more time to using OnlineProver."

6.3.2 Qualitative questions

We aimed to investigate students' behavior, particularly the types of problems they encountered and how they addressed them. To achieve this, we posed two *general* open-ended questions:

- 1. Mention something that you struggled with.
- 2. How would you typically resolve this particular struggle?

A total of 48 students responded to the first question, while 45 answered the second.

We analyzed the responses and categorized them based on the frequency of specific topics or issues mentioned. The challenges identified were not solely related to OnlineProver but also extended to broader topics, such as logic and natural deduction. Interestingly, the most common problems students faced were not with the tool itself but with understanding the theoretical concepts that the tool is designed to illustrate.

As shown in Table 2, the most frequently reported challenge (20 responses) was understanding formal methods, including general comprehension of proofs and the application of deduction rules. As noted earlier, these are first-year bachelor-level students, many of whom likely have little to no prior experience with formal proofs. Other difficulties unrelated to the OnlineProver system included the complexity of exercises (6 responses) and the lack of a visible list of assumptions, which was mentioned once.

Among issues related to OnlineProver, the most common challenges involved the input system (5 responses), such as difficulties entering logical symbols in ASCII code and the inability to delete internal nodes or move entire branches in a proof tree. Additionally, 4 students reported difficulties interpreting error messages in the feedback system. Bugs were also noted by two respondents; however, these have already been corrected.

Problem Category	Number of Responses	
Not Related to OnlineProver		
Understanding deduction rules	20	
Exercises are hard to solve/understand	6	
Missing list of assumptions (implicit context)	1	
Related to OnlineProver		
Input system (difficulty entering special symbols)	5	
Feedback system (hard to understand feedback)	4	
Bugs (already corrected)	2	

Table 2: Student Problems.

As shown in Table 3, the most common strategy students used to address challenges was, as expected, asking for help from teaching assistants or classmates, with 19 responses. Trial-and-error methods were also widely used, as noted by 9 respondents, while a smaller group relied on studying materials or external sources (3 responses). Interestingly, only one respondent reported using large language models like ChatGPT as part of their problem-solving process.

6.3.3 OnlineProver vs Pen and Paper Proofs

To compare the traditional approach of teaching formal proofs using pen and paper with OnlineProver, we asked the following questions:

Solution Strategy	Number of Responses
Trial and error / brute force	9
Asking a teaching assistant or a classmate for help	19
Asking large language models	1
Reading studying materials / external sources	3

Table 3: Students' Solutions.

- 1. What did you find positive about doing exercises by hand compared to using Online-Prover?
- 2. What did you find positive about doing exercises using OnlineProver compared to doing them by hand?

These questions aim to identify student challenges and preferences, as well as compare learning efficiency and effectiveness between the two approaches.

Table 4 shows that the largest group of students (13 responses) considered pen and paper proofs to be more beneficial for learning and understanding. Students also appreciated the flexibility of doing proofs on paper (12 responses). Some students (3 responses) noted that pen and paper proofs allowed them to use the same notation as in the textbook, which can be particularly helpful for beginners. Additionally, 4 students mentioned that they had not completed any proofs by hand, suggesting that OnlineProver is the dominant approach in their learning process.

Pen and Paper Proofs	Number of Responses
Better for Learning and Understanding	13
Ease of Use and Flexibility	12
Notation like in the book	3
Have not done by hand	4

Table 4: Pen and Paper Proofs Positives.

Table 5 shows that OnlineProver was positively received for its feedback (error messages) with 9 responses, and the possibility to check proof correctness (9 responses). The tool's user interface was also appreciated, with 7 responses mentioning that it made solving proofs faster, while 5 students appreciated its overall ease of use and flexibility. Additionally, 6 responses highlighted how OnlineProver supported their learning process.

OnlineProver	Number of Responses
Feedback system	9
Proof checker	9
Speed and Efficiency	7
Support for Learning	6
Ease of Use and Flexibility	5

Table 5: OnlineProver Positives.

Overall, two responses stand out. One student, who is retaking the course, compared the tool ProofWeb, used in previous years, with OnlineProver and found the latter to be significantly better.

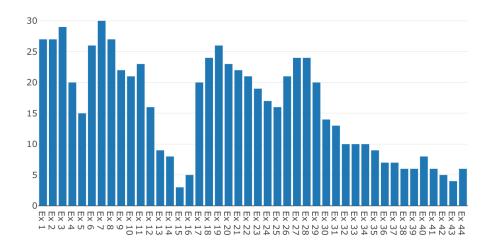


Figure 8: Number of attempts per exercise.

This comparison highlights the improvement and effectiveness of OnlineProver in enhancing the learning experience. On the other hand, another student pointed out a potential drawback of relying solely on digital tools like OnlineProver. They noted that when switching back to pen and paper proofs, they felt insecure about their proofs, pointing out the importance of balancing the use of digital tools with traditional methods to reinforce confidence and understanding.

6.4 Data analysis

We have collected the EDN files that contain history of all actions (explained in Section 4). Overall 36 students voluntarily submitted their data for evaluation. For a demonstration purposes and to validate some of our questions in the questionnaire, we have decided to analyse the student interaction with exercises, time spent, checks performed, the ratio of deletions to additions, edit operations per exercise, and the relationship between edits and checks.

6.4.1 Exercise Attempts Analytics

The current OnlineProver deployment consisted of 44 exercises. The following bar chart (Figure 8) shows the number of students who attempted each exercise.

From this bar chart, we can identify which exercises were less popular. As expected, the later exercises were attempted the least, but it is interesting to note that some exercises in the first half of the set (e.g., 13–17) were attempted by fewer than 10 students.

When this data is converted into a pie chart (Figure 9) and compared with Chart 6, similar trends emerge.

The data broadly correspond to the responses in the questionnaire, even though a different subset of students might have completed the questionnaire and submitted the EDN files. The high number of

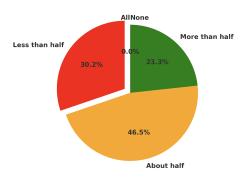


Figure 9: Exercise attempts.

attempts in earlier exercises aligns with the larger proportion of students indicating they solved about half or fewer exercises. Only a small percentage of students solved all the exercises, which is consistent with the lower participation observed in later exercises.

6.4.2 Time Spent Analysis

The following chart (Figure 10) shows the average time spent per exercise in minutes.

The chart illustrates the distribution of time spent on various exercises, labeled from "EX 1" to "EX 44." It highlights significant variations in time spent across exercises, with some exercises showing outliers where participants spent considerably more time. This analysis provides insights into the relative difficulty of the exercises.

6.4.3 Behavioural Analytics

The following charts provide a detailed analysis of user interactions with OnlineProver, focusing on various behavioural metrics. These include the number of checks performed per exercise, the ratio of deletions to additions, the frequency of edit operations across exercises, and the relationship between edit operations and checks.

By analysing these metrics, we aim to investigate patterns in how students interact with the tool, their problem-solving approaches, the effectiveness of their trial-and-error strategies, and the usability of the system. These insights can help us understand how students solve their proofs and refine the learning process.

Figure 11 shows the number of times the check button was used per exercise. This feature allows students to confirm if their (partial) solution is correct or to receive feedback in case of incorrect derivations. As observed, the number of checks aligns with the time spent chart (Figure 10).

If we examine the ratio between edit actions and usage of the check feature (Figure 12), it becomes evident that the feedback system was utilized frequently, as students, on average, used the check button at least once in every ten edit operations.

7 Discussion

In this section, we begin by discussing the suggestions for improvement and their impact collected in the questionnaire. Following that, we will address the research questions formulated in Section 2, based on the evaluation of our intervention.

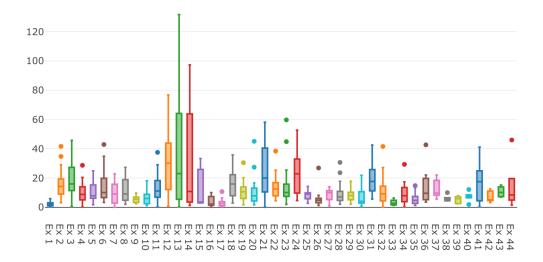


Figure 10: Time spent per exercise in minutes.

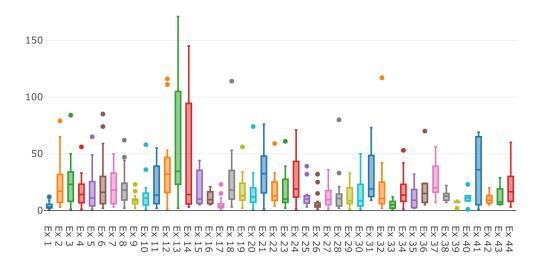


Figure 11: Number of checks per exercise.

Figure 12: Ratio between edit actions and checks.

7.1 Suggestions

The last three questions of the questionnaire aimed to collect overall feedback on the tool, specifically how it could be improved to better meet users' needs. We were interested in understanding how students evaluate the tool, what its strengths and weaknesses are, and whether they have any additional comments or suggestions. To gather this information, we asked the following questions:

- 1. Do you have any suggestions that would improve the quality of the tool for you?
- 2. How would this improvement help you use the tool?
- 3. Do you have any other suggestions?

7.1.1 Suggestions for Improvement

In the first question, we asked students for improvement suggestions. We have categorized them and ordered according the number of responses mentioning the category in Table 6.

The feedback collected on the tool suggests possible improvement in several areas. Formatting and shortcuts being the most requested (11 responses), followed by UI suggestions (8), and the ability to create own proof trees (7).

Students mostly suggested to add a easier way for entering special symbols or keyboard shortcuts for this purpose. For that, we do plan to implement a better way of entering special symbols. UI improvements mentioned draggable elements and customizable layouts for better user experience. The ability to create own formulae was often mentioned, including features like a sandbox mode and generating LaTeX or image outputs. Learning support received six responses, where students suggested to add detailed tutorials, rule explanations, and hints to aid understanding. Feedback system enhancements suggested that

Category	Number of Responses
Formatting and Shortcuts	11
UI Suggestions	8
Ability to Create Own Proof Trees	7
Learning Support and Tutorials	6
Improve Feedback System	2
List of Hypothesis	2

Table 6: Suggestions for Improvement.

Category	Number of Responses
Better Learning and Understanding	9
Time-Saving and Efficiency	8
Customization and Personalization	3
Improved Usability and Accessibility	5
Interface and Navigation	5

Table 7: Expected Impact of Improvements.

there is a room for improvement of error messages. At last 2 students mentioned that it would be nice to be able to display a list with the current hypotheses in a proof tree.

7.1.2 Expected Impact of Improvements

In the second question, we asked students how do they think their suggestion would improve our tool. We have categorized them and ordered according the number of responses mentioning the category in Table 7.

The expected impact of the suggested improvements mentions several areas. Time-saving and efficiency were pointed out as major benefits, with students noting that the changes like ability to input own formulae would eliminate the need for paper, reduce redundant actions, and would make their work faster. Specific mentions included making it easier to fit large proofs on screen, requiring fewer clicks, and make the process of writing and understanding proofs easier.

Better learning and understanding was pointed out as the most important expected impact of the suggested improvement. Students mentioned the importance of using the same notation as in the textbook, smarter error messages, which would create better navigation and readability.

To address these points, we plan to implement a more intuitive system for entering special symbols, such as using LATEX syntax, which will automatically convert the input into the corresponding symbol. This feature will also improve readability, as special symbols will be displayed exactly as they appear in the textbook. Draggable elements and customizable layouts are under consideration to enhance usability. Additionally, we aim to introduce a sandbox mode and export options for exercises, as well as more detailed tutorials to support learning. The feedback system in OnlineProver is particularly important. We plan to expand its capabilities beyond simple error messages to a data-driven, intelligent system that provides tailored feedback based on the user's behavior.

Category	Number of Responses
Customization and Personalization	3
Improved Usability and Accessibility	6
Interface and Navigation	4
Own proof tree	4

Table 8: Other Comments.

7.1.3 Additional Comments

In the last question, we asked students if they want to add any other comment. We have categorized answers in Table 8.

Other suggestions focused on customization, usability, and interface improvements to enhance the flexibility and user experience of OnlineProver. Students requested customizable themes and options to reduce reliance on shift-key inputs for typing. Other suggestions included the ability to save progress and make the layout more compact to handle wide proofs effectively. For interface and navigation, recommendations included clearer distinctions between predicate and propositional logic and smoother transitions between exercises. Additionally, students requested the ability to enter their own formulae.

To address these suggestions, we plan to implement the following. As mentioned earlier, we will introduce a new input system. The ability to change themes has already been implemented, and we will incorporate this feature into the plan tutorials. Saving progress and importing it back is another feature we plan to implement in future versions. Regarding navigation, we will improve the distinction between topics and make switching between them easier.

7.2 Addressing the Research Questions

In this section, we address the research questions formulated in the Section 2.

7.2.1 RQ1

Our first research question is as follows:

Is our tool used as a proof assistant or a proof checker?

As discussed in the Evaluation and Discussion sections, the feedback system is frequently mentioned by students, even though we did not ask about it directly. From Table 5, it is evident that students positively highlighted the feedback system and the check feature, each mentioned exactly nine times. Based on this, we can conclude that OnlineProver is used both as a proof assistant and as a proof checker.

7.2.2 RQ2

The feedback system was designed to simulate the type of guidance a teacher would provide—offering hints rather than solving problems outright—so that students could independently work through their proofs. While many online tools for reasoning provide solutions directly (similar to a calculator), our goal was to develop a system that helps students solve proofs on their own. Since the tool targets computer science students, we anticipated they would adopt a trial-and-error strategy similar to programming. This led us to formulate the following question:

How do students use trial-and-error strategies with OnlineProver to overcome difficulties in solving proofs?

As shown in Table 3, the trial-and-error approach is the second most common strategy used for problem-solving. This hypothesis is further supported by the analysis of EDN files, as shown in Figure 12, where students, on average, used the check button after almost every derivation step.

7.2.3 RQ3

We spent a lot of time designing and developing a useful feedback system. As mentioned earlier, we aimed to strike a balance between providing enough information to give a hint on how to continue and explaining where the derivation is incorrect, without turning our tool into a "calculator." This led us to ask the following question:

Do students perceive the tool's feedback as useful?

The data analysis and students' responses in the questionnaire confirm our hypothesis. Although four students reported that the error messages could be improved, overall, we can conclude that the tool's feedback system was positively evaluated.

7.2.4 RQ4

The last question aimed to investigate whether students consider our tool more useful for learning formal proofs compared to the traditional pen-and-paper approach. We consider our tool to be a teaching aid; it should not replace pen-and-paper proofs but rather supplement them. Therefore, we formulated the following question:

Do students consider learning proofs using OnlineProver more effective than learning proofs by pen and paper?

When analyzing the responses in the comparison section (Section 6.3.3), we found that students consider pen-and-paper proofs to be more effective for learning than using the tool. However, they also valued OnlineProver positively as a teaching aid, particularly the feedback system and check feature, which enhanced the learning process. Thus, our initial hypothesis can be considered confirmed.

8 Conclusion and Future Work

OnlineProver has demonstrated significant potential as a teaching tool for enhancing the understanding of formal proofs in computer science education. Its user-friendly design, together with a realtime feedback system, helps students address challenges in learning abstract logical concepts. The System Usability Scale (SUS) score and student feedback indicate that while the tool is well-received, there is room for improvement, particularly in usability features and feedback clarity.

The trial-and-error strategy supported by OnlineProver encourages active learning and aligns well with the pedagogical needs of computer science students. The integration of structured error messages enhances reasoning with proofs, offering a effective learning experience. Notably, while students viewed the tool positively, they also emphasized the importance of traditional methods, such as the use of penand-paper proofs. OnlineProver is an innovative tool offering valuable features for both teachers and students. For teachers, it enables analysis of student behavior, helping identify "problematic" exercises

and pinpointing areas where students struggle. From a student's perspective, OnlineProver serves as a practical learning aid without requiring the mastery of a new language, as would be necessary with full-scale proof assistants. Students benefit from real-time feedback and the ability to check correctness of their (partial) solutions.

Future work will focus on improving the tool's interface, enhancing the feedback system, and adding more interactive tutorials. The current deployment relies on a hard-coded proof engine. A significant future challenge is the planned development of two domain-specific languages: one for defining formal methods and their deduction rules, and another for creating exercises. These will compile into a proof engine, expanding the tool's flexibility and scope.

References

- [1] Elisa Baniassad, Lucas Zamprogno, Braxton Hall & Reid Holmes (2021): STOP THE (AUTOGRADER) INSANITY: Regression Penalties to Deter Autograder Overreliance. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, Association for Computing Machinery, pp. 1062–1068, doi:10.1145/3408877.3432430.
- [2] Krysia Broda, Jiefei Ma, Gabrielle Sinnadurai & Alexander Summers (2007): *Pandora: A Reasoning Toolbox using Natural Deduction Style*. *Logic Journal of the IGPL* 15(4), pp. 293–304, doi:10.1093/jigpal/jzm020.
- [3] J Brooke (1996): SUS: A quick and dirty usability scale. Usability Evaluation in Industry.
- [4] IT University of Copenhagen (2024): Foundations of Computing Discrete Mathematics. Available at https://learnit.itu.dk/local/coursebase/view.php?s=ft&view=public&ciid=1474. Accessed: 25.11.2024.
- [5] Fatima Abu Deeb & Timothy Hickey (2023): *Impact of reflection in auto-graders: an empirical study of novice coders.* Computer Science Education 0(0), pp. 1–22, doi:10.1080/08993408.2023.2262877.
- [6] Stephen H. Edwards (2004): Using software testing to move students from trial-and-error to reflection-in-action. In: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '04, Association for Computing Machinery, p. 26–30, doi:10.1145/971300.971312.
- [7] Arno Ehle, Norbert Hundeshagen & Martin Lange (2018): *The Sequent Calculus Trainer with Automated Reasoning Helping Students to Find Proofs.* In Pedro Quaresma & Walther Neuper, editors: Proceedings 6th International Workshop on *Theorem proving components for Educational software*, Gothenburg, Sweden, 6 Aug 2017, *Electronic Proceedings in Theoretical Computer Science* 267, Open Publishing Association, pp. 19–37, doi:10.4204/EPTCS.267.2.
- [8] Mario Frank & Christoph Kreitz (2018): A Theorem Prover for Scientific and Educational Purposes. In Pedro Quaresma & Walther Neuper, editors: Proceedings 6th International Workshop on Theorem proving components for Educational software, Gothenburg, Sweden, 6 Aug 2017, Electronic Proceedings in Theoretical Computer Science 267, Open Publishing Association, pp. 59–69, doi:10.4204/EPTCS.267.4.
- [9] Olivier Gasquet, François Schwarzentruber & Martin Strecker (2011): Panda: A Proof Assistant in Natural Deduction for All. A Gentzen Style Proof Assistant for Undergraduate Students. In Patrick Blackburn, Hans van Ditmarsch, María Manzano & Fernando Soler-Toscano, editors: Tools for Teaching Logic, Springer Berlin Heidelberg, pp. 85–92, doi:10.1007/978-3-642-21350-2_11.
- [10] Volker Halbach (2010): The Logic Manual. Oxford University Press.
- [11] CS Kaliszyk, F van Raamsdonk, Freek Wiedijk, Hanno Wupper, Maxim Hendriks & Roel de Vrijer (2008): Deduction using the ProofWeb system.
- [12] Ville Karavirta, Ari Korhonen & Lauri Malmi (2006): On the use of resubmissions in automatic assessment systems. Computer Science Education 16(3), pp. 229–240, doi:10.1080/08993400600912426.
- [13] Joomy Korkut (2023): A Proof Tree Builder for Sequent Calculus and Hoare Logic. In Pedro Quaresma, João Marcos & Walther Neuper, editors: Proceedings 11th International Workshop on Theorem Proving

- Components for Educational Software, Haifa, Israel, 11 August 2022, Electronic Proceedings in Theoretical Computer Science 375, Open Publishing Association, pp. 54–62, doi:10.4204/EPTCS.375.5.
- [14] Joachim Tilsted Kristensen, Ján Perháč, Lars Tveito, Lars-Bo Husted Vadgaard, Michael Kirkedal Thomsen, Oleks Shturmov, Samuel Novotný & Sergej Chodarev (2024): *OnlineProver: First Experience with Teaching Formal Proofs*. In Hans Georg Schaathun, Per Lauvås & Torstein Strømme, editors: *UDIT (The Norwegian Conference on Didactics in IT education)*, doi:10.5324/nikt.6205.
- [15] Graham Leach-Krouse (2018): Carnap: An Open Framework for Formal Reasoning in the Browser. In Pedro Quaresma & Walther Neuper, editors: Proceedings 6th International Workshop on Theorem proving components for Educational software, Gothenburg, Sweden, 6 Aug 2017, Electronic Proceedings in Theoretical Computer Science 267, Open Publishing Association, pp. 70–88, doi:10.4204/EPTCS.267.5.
- [16] James R. Lewis (2018): *The System Usability Scale: Past, Present, and Future.* International Journal of Human–Computer Interaction 34(7), pp. 577–590, doi:10.1080/10447318.2018.1455307.
- [17] Benjamín Machín & Luis Sierra (2011): *Yoda: a simple tool for natural deduction*. In: Proceedings of the Third International Congress on Tools for Teaching Logic (TICTTL'11), 6680.
- [18] Carroll Morgan (2016): (In-)Formal Methods: The Lost Art. In Zhiming Liu & Zili Zhang, editors: Engineering Trustworthy Software Systems, Springer International Publishing, Cham, pp. 1–79, doi:10.1007/978-3-319-29628-9
- [19] John Newsome Crossley (2023): What is mathematical logic? An Australian odyssey. Logic Journal of the IGPL 31(6), pp. 1010–1022, doi:10.1093/jigpal/jzac057.
- [20] Jeff Sauro (2011): *Measuring Usability with the System Usability Scale (SUS)*. Available at https://measuringu.com/sus/. Accessed: 2024-11-26.
- [21] Jeremy Seligman & Declan Thompson (2015): *Teaching natural deduction in the right order with Natural Deduction Planner*. arXiv:1507.03681.
- [22] Bruce L. Sherin (2001): A Comparison of Programming Languages and Algebraic Notation as Expressive Languages for Physics. International Journal of Computers for Mathematical Learning 6(1), pp. 1–61, doi:10.1023/A:1011434026437.
- [23] Rafael del Vado Vírseda, Eva Pilar Orna, Eduardo Berbis & Saúl de León Guerrero (2011): *A Logic Teaching Tool Based on Tableaux for Verification and Debugging of Algorithms*. In Patrick Blackburn, Hans van Ditmarsch, María Manzano & Fernando Soler-Toscano, editors: *Tools for Teaching Logic*, Springer Berlin Heidelberg, pp. 239–248, doi:10.1007/978-3-642-21350-2_29.
- [24] Davi Romero Vasconcelos (2023): ANITA: Analytic Tableau Proof Assistant. In Pedro Quaresma, João Marcos & Walther Neuper, editors: Proceedings 11th International Workshop on Theorem Proving Components for Educational Software, Haifa, Israel, 11 August 2022, Electronic Proceedings in Theoretical Computer Science 375, Open Publishing Association, pp. 38–53, doi:10.4204/EPTCS.375.4.
- [25] David Weintrop & Uri Wilensky (2015): To block or not to block, that is the question: students' perceptions of blocks-based programming. In: Proceedings of the 14th International Conference on Interaction Design and Children, IDC '15, Association for Computing Machinery, New York, NY, USA, p. 199–208, doi:10.1145/2771839.2771860.
- [26] Frank Zenker, Christian Gottschall, Albert Newen, Raphael van Riel & Gottfried Vosgerau (2011): Designing an Introductory Course to Elementary Symbolic Logic within the Blackboard E-learning Environment. In Patrick Blackburn, Hans van Ditmarsch, María Manzano & Fernando Soler-Toscano, editors: Tools for Teaching Logic, Springer Berlin Heidelberg, pp. 249–255, doi:10.1007/978-3-642-21350-2_30.