Journal of Educational Studies in
Mathematics and Computer Science

Improving mathematical proving skills through interactive theorem proving

Sana Stojanović Đurđević [1, a, *], Andrija Urošević [1, b], Filip Marić [1, c]
[1]University of Belgrade, Faculty of Mathematics, Department of Computer Science and Informatics, Belgrade, Serbia
[a]sana.stojanovic.djurdjevic@matf.bg.ac.rs, [b]andrija.urosevic@matf.bg.ac.rs, [c]filip.maric@matf.bg.ac.rs

Received: 10 October 2024; **Accepted:** 14 December 2024

Abstract

Understanding and constructing proofs of mathematical theorems is a fundamental component of mastering mathematics and developing the logical apparatus. In addition, the theorems in mathematical textbooks are often only understandable through their proofs. However, students sometimes lack the precision needed to write detailed proofs and the understanding of basic proving concepts.

In this paper, we propose the use of interactive theorem provers by math teachers with the goal of improving students' mathematical proving skills and understanding of logical rules. This approach utilizes feedback from interactive theorem provers while gradually progressing through carefully selected areas of mathematics and computer science. We propose the focused use of two areas, firstly the subset of elementary mathematical theorems and secondly an area that can be selected by teachers in relation to their subject area (and can include almost any area of mathematics and computer science).

The proposed approach was implemented in the elective course "Introduction to Interactive Theorem Proving" for fourth year undergraduate students at the Faculty of Mathematics in Belgrade. The course utilizes the interactive theorem prover Isabelle and encompasses constructing formal proofs of elementary mathematical theorems (selected as the first area) and the formal verification of appropriate functional algorithms (selected as the second area). Upon completion of the course, students attain proficiency to formalize tasks and solutions from international mathematical Olympiads and to verify certain algorithms and data structures.

The results of our approach are based on the experience with six generations of students who have carried out a variety of formalization projects from different areas of mathematics and computer science, thus demonstrating the vast applicability of the proposed approach. We show that under intensive teacher guidance, students are quite capable of understanding and constructing formal proofs. This resulted in an extremely positive outcome in the first use of interactive theorem proving at the undergraduate level. The course is designed in such a way that the required prior knowledge is ground-level mathematical knowledge. Thus, it is possible to learn interactive theorem proving in parallel with the mathematical logic courses, with the aim of additionally understanding important logic and proof concepts used in all areas of mathematics from the very beginning.

Keywords: proof understanding, formal proof, interactive theorem proving, Isabelle, teaching mathematics

MSC2020: 97P99, 97C70

[*] Corresponding author
E-mail address: sana.stojanovic.djurdjevic@matf.bg.ac.rs

Original Research Article

1. Introduction

What is a proof? *The proof* of a statement is usually presented as a sequence of logical steps within a particular context and a theory. Starting from the premises of the statement, using previously given axioms and previously established and proven theorems, a chain of steps is derived until the conclusion of the statement is found. *The purpose of the proof* is twofold. First, the proof provides *a justification* for the statement that ensures its truth. Secondly, the proof provides *an explanation* for the statement, i.e. it clarifies and demonstrates *why* the statement is true. The first context is more important for mathematicians, while the second context is crucial for learning and understanding mathematical proofs in the educational process (and is often used in the construction of proofs in exams).

Depending on the logical system, there are different ways in which a proof can be conducted. Some systems use only basic rules throughout the proof, while others use proof steps based on previously proved lemmas, thus shortening the proof. Some systems take a constructive approach, while others allow the law of excluded middle and thus enable proof by contradiction and indirect proofs. Therefore, the difficulty of constructing a proof, as well as the readability and understanding of the finished proof, depends to a large extent on the experience of the mathematician constructing or reading the proof.

In an educational setting, forward chaining pen-and-paper proofs is usually the goal. These proofs, while informal, are accepted when the trained and competent mathematician is confident that a more detailed formal proof of the statement can be constructed in a specific formal system. Typically, proofs in formal systems (such as natural deduction) are extremely tedious and detailed, containing many steps that often seem unnecessary, and they are rarely carried out in an educational context. The type of proofs used also depends on the area of mathematics. Mainly, proofs in geometry are established as visual proofs, and this is exactly the area where the need to demonstrate the correctness of the conclusion was originally recognized due to its practical nature (the measurement of land).

The language of proofs. Looking at the language of mathematical proofs, one can notice that almost every proof consists mainly of mathematical formulas linked by explanations in natural language. Natural language can be imprecise, which can lead to a certain vagueness. A pure format of proofs, without natural language, is possible and reserved for some areas of mathematics.

Are proofs always correct? Before the XXth century, the correctness of proofs was only checked by a proficient mathematician. Errors in the proofs were usually found by the authors, sometimes by the reviewers, but there were cases where it took more than a decade for someone to discover an error in a proof. As the proofs get more and more complicated, finding errors becomes a challenge, and the use of natural language can become ambiguous. For pen-and-paper proofs, sometimes a detailed analysis of the proof follows. This is particularly noticeable in proofs of geometric theorems, which are often followed by a discussion of the nondegeneracy conditions.

The importance of proof assistants and formal proofs. One of the main reasons for the increased use of computer-assisted theorem proving is errors found in published proofs of mathematical theorems, as well as errors found in some significant computer programs (Marić, 2015; Buzzard, 2022). Errors found in computer programs can have immense consequences, depending on the type of program used. Some errors can simply be a nuisance to the user, while others can harm security or financial status of the user. All this has led to an intensive use of formal proofs that can be verified using a computer. In recent years, they have gained more attention and have been researched more than ever.

Reliability of computer provers. Since errors can also occur in computer programs, one can question the reliability of programs used for theorem proving as well. Programs used for this purpose must have a small core that can be checked by hand. These systems tend to use a self-checking system that must verify the validity of all definitions before they are used in proofs.

When students first encounter interactive theorem proving, it is noticeable that their understanding of the notion of proof seems to be at a relatively low level, as students often make trivial mistakes. This paper proposes the use of interactive theorem provers to lighten the learning curve of basic proving techniques. On the other hand, it provides a basis of formal software verification.

The approach. Interactive theorem provers have so far been implemented and used almost exclusively in the research setting. The approach proposed in this paper advocates the use of interactive theorem provers in the educational setting. The proposed approach was implemented in the course "Introduction to interactive theorem proving" at the Faculty of Mathematics, University of Belgrade, Serbia. The course was organized in two parts. The first part utilizes proving elementary mathematical theorems (about logic, sets, functions, relations, natural and real numbers) using various techniques (natural deduction rules, case analysis, and induction). The second part utilizes the verification of functional programs ensuring their correctness.

The result is a better understanding of mathematical proving concepts and rules, as well as a better understanding of the relationship between the program specification and the mathematical theorems (that describe it) used in the formal proof of the correctness of the algorithm. This methodology focuses on student engagement in formalizing complex mathematical and programming concepts, such as International Mathematical Olympiad problems, geometry textbook, functional algorithms, and abstract algebra.

Organization of the paper. In Section 2 we introduce basics of the interactive prover Isabelle and give an overview of similar ideas and courses on interactive theorem proving. In Section 3 we present the methodology and its implementation in the course "Introduction to interactive theorem proving". In Section 4 we briefly show the results of our approach based on student contributions and in Section 5 we draw final conclusions and present some ideas and recommendations for the following years.

2. Background

Expressing mathematical theories in such a way that computers can understand them and verify their correctness dates back to the 1960s and the Automath language developed by de Bruijn (Dechesne, 2012). It is considered a forerunner for some interactive theorem provers based on type theory. When working on mathematical problems, there are some discrepancies between formal and pen-and-paper proofs that can lead to changes in the proofs and make interactive theorem proving in some cases very time-consuming. These differences are best measured by the de Bruijn factor, i.e. the ratio between the length of the formal proof and the original informal textbook proof. Another challenge with theorem proving using a computer is the steep learning curve, which limits its usability for broader groups of mathematicians.

2.1. Different levels of computer assisted theorem proving

Computer assisted theorem proving has three levels of interaction between a computer and a human. In all cases, the mathematician first must formulate all axioms, additional lemmas, and the theorem to be proved in the language of the corresponding prover.

Automated theorem proving uses computers to check whether a given conjecture is a theorem. In most cases, an automated theorem prover outputs a yes/no answer. Some automated theorem provers can generate a proof, but the proof is not intended for human manipulation and understanding (although an experienced user can follow the proof to a considerable extent). Automated theorem provers are mostly intended for specific and narrow domains (Chou, 2001). For example, automated theorem provers based on algebraic methods achieve the best results in the field of geometry (Chou, 1988). In the context of computer assisted theorem proving, the average student is probably more familiar with automated theorem provers, i.e. dynamic geometry systems, such as GeoGebra (Hohenwarter & GeoGebra Team, 2001). The automated theorem provers used

in such systems are mostly efficient algebraic provers, but the generated proofs are not human-readable and do not look like textbook proofs.

Proof-checking uses both a conjecture and its proof, verifying individual steps in a specific formal system. It is rarely used because it is too tedious for humans, but it can help to find errors in important theorems (Dowek, 2015).

Interactive theorem proving uses special languages for theorem proving, which in most cases are inspired by functional programming languages. They represent an ideal balance between computer and human effort. The mathematician guides the prover and invokes its capabilities (and some level of automated theorem proving in the background) for carefully selected parts of the proof. The mathematician formulates high-level descriptions of the proof, which the computer checks, while the simple proof steps are often left to the prover to automatically verify (and sometimes to automatically fill in). There are already a substantial set of interactive theorem provers (Wiedijk, 2006), and the popularity of interactive theorem proving has increased in recent years with the development of new interactive theorem provers such as Lean (de Moura & Ullrich, 2021).

2.2. Interactive theorem provers and education

In recent years, there has been a growing number of conferences and summer schools that are directed towards the relationship between education and interactive theorem proving. The workshop series *Theorem proving components for educational software* (ThEdu, 2024) brings together experts in automated deduction with experts in education to further clarify the shape of new software production and discuss existing systems.

The summer school *Proof assistants for teaching proof and proving* (PAT, 2023), taking place in France, focuses on joining researchers, teachers, students and stakeholders interested in the use of proof assistants in teaching. Such gatherings offer a unique perspective and provide insight into the influence that theorem provers have on educational practice.

The international summer school *Interactions of Proof Assistants and Mathematics* (ITP, 2023), held at the University of Regensburg, Germany, presented the state of the art in proof assistants from different angles: theoretical foundations, engineering aspects, applications in different areas including mathematics and computer science. The aim of this summer school was to introduce proof assistants to students of mathematics and computer science, while fostering collaboration and exchange between these two communities.

2.3. Interactive theorem prover Isabelle

Isabelle is a generic interactive theorem prover (Wenzel et al., 2008) that can be used to express mathematical formulas in a formal language, which can then be proven by logical calculus. Isabelle/HOL (Nipkow et al., 2002), an instance of Isabelle for Higher Order Logic (HOL), is a logical framework that considers functions as first-class citizens (and allows quantification over them), and supports a wide range of mathematical theories. There is also an online collection of formal Isabelle proofs called *AFP - Archive of Formal Proofs* (Blanchette, 2015), which is developed and maintained by the Isabelle community. The popularity of the prover is reflected in the development of online systems for proving competitions using interactive theorem prover Isabelle (Haslbeck & Wimmer, 2018).

Interactive theorem proving gives the user the opportunity to examine the proof in more detail than with pen and paper. Each step of the proof must be fully justified, for example, each proof by cases must be done thoroughly and for both cases. One can not use phrases such as “the proof is similar to the previous one” or “this part of the proof is trivial”. The smallest details of the proof must be justified and verified by the prover. In the earlier versions of the prover, the proofs were mostly written manually, but in later years a higher degree of automation is available as part of the Isabelle theorem prover platform. Automated theorem provers are not needed or used in the approach described here.

2.4. Similar courses

There are several university courses that deal with the topic of interactive theorem proving.

Isabelle/HOL is a system being developed at the Technical University of Munich and the University of Cambridge. It is not surprising that there are courses on Isabele/HOL at these two universities. For example, the course Formal Proof in Mathematics and Computer Science at TUM (Nipkow et al., 2018) gives a broad perspective on formal proofs and their applications. Tobias Nipkow, one of the main authors of Isabelle/HOL, describes his experience of teaching a course on the semantics of programming languages using proof assistant, and shows that completing such a course helps students to overcome elementary mistakes when writing proofs (Nipkow, 2012). In a paper with the very interesting title "Teaching Semantics with a Proof Assistant: No More LSD Trip Proofs", he describes how the use of proof assistants has a positive impact on students' proof writing skills. It is not uncommon for students' arguments to look more like a LSD trip rather than a coherent logical chain, and this is corrected after working with proof assistants.

The courses at the University of Cambridge (Paulson, 2020) and the course at the University of Freiburg (Biere, 2024) provide an introduction to interactive formal verification. Jacobsen and Villadsen describe a course on automated reasoning using the Isabelle proof assistant at the Technical University of Denmark (Jacobsen & Villadsen, 2023). They focus on the problem of testing proof correctness and advocate that the use of Isabelle enables almost automatic grading of large parts of the exam. The course of automated reasoning at the University of Edinburgh (Fleuriot, 2021) relies heavily on the use of Isabelle/HOL proof assistant. There are several courses where proofs are taught using the proof assistant Coq (Inria, 1984) (e.g. at Chalmers University, Sweden (Juan, 2017) and at the University of Strasbourg, France).

Silvia Ghilezan teaches several courses on interactive theorem provers at the University of Novi Sad, Faculty of Technical Sciences (Ghilezan, n.d.). The courses focus on introducing students to various interactive theorem provers, including Coq, Isabelle and Agda (Norell, 2009).

More detailed list of courses on theorem provers can be found in a recent survey (Minh at al., 2024). They explore usage of various automated theorem provers, interactive theorem provers and specialized interactive theorem provers in educational settings together with challenges in their adoption.

The syllabus of these courses show that many of them are aimed at final-year students and prepare them for advanced topics in computer science (program verification, programming language semantics, etc.). Elementary proof techniques are often assumed and only very briefly demonstrated in the theorem prover. In contrast, our focus is on the acquisition of general techniques and proofs, so we devote much more time to teaching elementary examples. Somewhat similar to our approach are the courses in Strasbourg, France, where the topic of interactive theorem proving is covered in several courses in all years of undergraduate studies (Narboux, 2021). At the beginning, they focus on general proof techniques, using specialized environments (e.g. Edukera), and in the final year they apply interactive theorem proving to applications in computer science and software foundations.

3. Methodology

The main idea of the approach presented in this paper is to utilize the extensive knowledge and doctrine accumulated in research in the area of interactive theorem proving and to give students the opportunity to practice various logical problems in the context of interactive theorem prover setting. The ultimate goal was to sharpen their observation skills and improve their understanding of important mathematical concepts. The goal was to construct mathematical proofs more precisely using the rules of natural deduction, intuitionistic logic, and first-order logic (FOL). We propose a carefully chosen set of elementary and advanced problems, designed to strengthen students' logical thinking skills within a specific subject area.

3.1. Implementation of the proposed approach

The proposed approach was implemented in the course "Introduction to interactive theorem proving" at the Faculty of Mathematics, University of Belgrade, Serbia. The interactive theorem prover used in our implementation of this approach is Isabelle/HOL. Although the authors' previous experience with theorem proving using Isabelle (Marić, 2008) influenced the choice, we must emphasize that there are reasons that justify the use of Isabelle in teaching mathematics and formal verification of algorithms, since substantial mathematical knowledge has already been formalized in Isabelle. Although not required, prior knowledge of functional programming languages has been shown to have a great impact on students' ability to quickly understand and learn the Isar programming language.

3.2. Course structure

The first half of the course is dedicated to learning the syntax and semantics of the script language and the Isar language, used in Isabelle for writing structured proofs. Various areas of logic and mathematics are used for this purpose. In this part of the course, automation is used only minimally. The second half of the course focuses on the verification of important algorithms and uses more automation.

Formulation of the problem. At the beginning of the course, students are introduced to the interactive theorem prover Isabelle, focusing first on basic Isabelle theories, such as simple FOL formulas that can be proved automatically (without user assistance). For this purpose theories involving syllogisms and logical puzzles are used, such as those presented in *Logical Labyrinths* (Smullyan, 2008), allowing students to practice stating and proving theorems in an interactive theorem proving environment. When working with a formal system (such as natural deduction), students are usually presented with a precise formulation of a theorem to be proved. In these cases, only the student's ability to conduct the proof is tested. Nevertheless, it is important to point out that the (perhaps even more) important step in theorem proving is the formulation of the correct statement of the theorem based on a natural language description of the problem. At the beginning of the course, students are taught how to formulate the statement, and how to use Isabelle's power to verify that the statement they are trying to prove is correct. Even if it seems like a trivial problem, it is important to point out that this step has been found difficult by students because it is easy to formulate a conjecture that can be formally proved but does not represent the original statement. For example, if the assumptions of the theorem are contradictory, any conclusion can be derived.

Mathematical aspects of the course. Throughout the course, students are introduced to natural deduction rules within Isabelle, examining both intuitionistic and classical rules. Most of the FOL formulas used (and proved automatically) at the beginning of the course can be proved using natural deduction rules only, and have been used as an exercise in this part of the course. This process enables students to understand the natural deduction rules beyond the usual level. In the next phase of the course, students learn the syntax of the Isar language, which is used to write structural proofs, and apply these skills to prove basic algebraic theorems about sets and some function properties with limited automation. Students also explore mathematical induction in Isar, proving basic summation formulas, equalities and inequalities. Finally, this part of the course covers the foundations of natural numbers and equips students with the essential tools for advanced formal reasoning.

Computer science aspects of the course. In this part of the course, students are introduced to algebraic data structures, starting with lists and binary trees. They learn how to define, state, and prove properties (operations and functions) for these fundamental data structures in Isabelle. In this way, students gain practical skills in both representing data and defining functions that describe them and can be formally verified. For example, students learn to define functions: the concatenation of lists and the reversal of a list by primitive recursion and then prove properties such as the following: A reversed concatenated list is a concatenation of swapped reversed lists. This

requires concepts such as primitive recursion and structural induction. As students progress, they will explore general recursion and functional induction, applying these concepts to known algorithms. This includes implementing and verifying sorting algorithms such as selection sort, insertion sort, quick sort, and merge sort. They will also learn how to verify algebraic algorithms such as fast exponentiation and Euclid's algorithm for finding the greatest common divisor. In addition, students will work on evaluating prefix expressions simulating stack machines and proving the correctness of such simulations. This comprehensive exposure provides students with a toolkit for formal reasoning in a wide range of computer science problems.

Course organization and materials. The part of the course described above is constructed over twelve weeks with weekly 2-hour lectures and 3-hour practice, supplemented by extensive online material. Over 300 pages of written material are provided, including theoretical content, examples and exercises (Stojanović Đurđević & Marić, 2021). In addition, students at practice receive exercise sheets with partially completed solutions, which are designed by the teachers for interactive engagement during practice sessions. Both the written resources and the practice sheets are fully typed and compiled within the Isabelle framework enabling students to directly modify the given files. The exam consists of three separate components: mathematics and computer science, which each account for 30% of the final grade, and a key project to formalize significant mathematical or programming knowledge, which accounts for 40% of the grade. The project is undertaken following the instructional period of twelve weeks. A public repository has been set up for this purpose (Stojanović Đurđević, 2020). This repository is maintained by the students with the help of the teachers and assistants of the course and contains mechanically checked formulations and solutions of various problems in the interactive theorem prover Isabelle/HOL.

Project organization. After teachers have selected a material for formalization, the primary style and notation of the project must be created to provide structured formalization concepts. To expedite feedback from teachers (or other students), an online chat room has been set up to encourage collaboration between students and teachers. To ensure correctness and prevent contradictions projects had to be monitored and maintained by the teachers. In the event that the material selected for the project contained independent tasks (usually when the formalization tasks were of a larger scope), no strict deadlines were set. However, if the tasks were interdependent, strict deadlines had to be set to avoid bottlenecks.

4. Results of the proposed implementation

An important aspect of this methodology is the organization of group projects for students. During the six years of the implemented course, different projects were presented and students were organized to formalize different areas of mathematics and computer science. Some problems were taken from the set of tasks and solutions from *the International Mathematical Olympiad (IMO)*, several problems from various geometry and algebra books, and some problems from books on functional algorithms and other specific algorithms. The problems that were formalized are listed here in chronological order, which means that the experiences of previous projects were taken into account when the next project was proposed, as can be seen in the text below. The challenges encountered in the previous projects were taken into account in the future, as the following projects had better organization of the students, precise organization methods, and more adequate teacher guidance and preparation.

4.1. Formalization of the International Mathematical Olympiad

The International Mathematical Olympiad is one of the most grandiose mathematical competitions in the world. In recent years, its challenging aspects have been recognized as so impressive that they have inspired the *IMO Grand Challenge* for Artificial Intelligence (AI), i.e. the development of an AI that would be capable of winning a gold medal at the Olympiad (IMO Grand Challenge, n.d.). The more detailed analysis of the level of difficulty in various areas (Marić,

2020) states that this task might be too overwhelming even for a computer. The main reason for this is the challenging and potentially difficult aspect of formulating a formal statement of the given problem (as we also noted at the beginning of the implemented course presented here, even for simpler problems). The use of different problem representations can significantly affect the difficulty of formal proving of the original problem (Wiedijk, 2006). There are specialized provers that use machine learning algorithms, developed solely for this purpose alone, that can, if given a manually translated formulation of the problem (in the formal language of the proposed system), compete with the average score of a human gold medalist in the area of geometry (Trinh et al., 2024) and silver medalist in other areas (AlphaProof and AlphaGeometry Teams, 2024).

IMO is organized annually for high school students in over 100 countries. Each year, the *Shortlist* is formed - a series of 30 challenging problems formulated by mathematicians from around the world. The competition itself consists of six problems from this list. After the competition, the shortlist is available together with detailed solutions in *The IMO Compendium* (Djukić et al., 2011). This made it possible for us to use these lists as material for the formalization. The problems come exclusively from secondary school mathematics, which makes both the problems and their solutions easy to understand for the students. It is therefore not surprising that the majority of students choose these problems as a first step towards formalizing significant mathematical knowledge. However, we must emphasize that these problems and especially their solutions are anything but trivial.

The IMO problems are cataloged in the following four areas: Geometry, Number Theory, Algebra, and Combinatorics. All areas except geometry have been formalized for the following reasons. Problems describing high school geometry require basic geometric knowledge that is easily comprehensible for humans and are therefore not explicitly stated in the presented solutions. Isabelle/HOL does not currently have enough knowledge to be able to express these solutions without first stating all the necessary axioms used in the specific problem and a solution. Since IMO problems were selected in the initial implementation of the proposed methodology, students were allowed to use as much as possible of the original shortlist solutions (and end up formulating proofs that largely resemble the original textbook-like proofs). The challenge for the students is similar to the challenges at the beginning of any formalization - identifying the correct formulation of the problem that can be described in a precise and comprehensible manner suitable for the computer theorem prover. Afterwards, the difficulty of the formalization varied from trivial to more complicated tasks that required the active participation of the professor's.

4.2. Formalization of the geometry theorems

The value of geometry is undisputed for understanding and learning the notion of proof. Since Euclid's Elements, detailed geometric proofs have been used for teaching young mathematicians (Heath, 1956). Subsequently, various axiomatic systems have been used, of which Hilbert's axiomatic system is the most popular (Hilbert, 1922). However, with the development of computer programs aimed at theorem proving, geometry turned out to be a very challenging task underlining the importance and difficulty behind understanding geometry proofs. While there is significant geometric knowledge that has been formally verified (Pham et al., 2011; Narboux 2006), formalizing geometry is still an excellent exercise for learning interactive theorem proving (and also for proof comprehension). Proofs from the existing formalizations are not trivially reused, so in most cases a new formalization has to start from scratch by defining basic axioms and lemmas used in the specific mathematical textbook that is being formalized. In some cases, even small changes to the theory used amounts to the same level of work that was previously done. This is the main reason that motivated development of some geometry specific automated theorem proving systems that can output fully readable and formally verifiable proofs (Stojanović-Đurđević, 2019; Gonzales 2024).

The project of formalization of the book "*Euklidska i hiperbolička geometrija - Euclidean and hyperbolic geometry*" (Lučić, 1994) was carried out over a period of 3 months. The challenge was

to organize a group of up to 50 students. The project was organized within only 3 shared files corresponding to the three groups of axioms (axioms of order, incidence axioms, and congruence axioms), which posed numerous challenges, both technical and organizational. To ensure a minimum number of conflicts, careful preparation and regulation was required by the teaching staff, who had to formally state axioms in the initial version of the files. Also, mindful enumeration of the definitions and theorems from the book, and grouping of these definitions and theorems into meaningful disjoint groups, was necessary. Definitions and theorems were formulated in the order in which they were stated in the book. The students got to formalize the first available group of definitions and theorems (which had not been previously formulated). This part of the project had a short deadline of just a few days. The limited number of days was needed in order to prevent a bottleneck (definitions and theorems usually required definitions from previous groups). In the second part of the project, the students got to prove a few theorems (not necessarily the ones they had formulated themselves). The project involved constant monitoring and providing guidance/feedback for each student. Also, consistency across formalization, style, and notation had to be regulated.

4.3. Formalization of abstract algebra

Proofs in abstract algebra, similar to geometry, are built up step-by-step from basic assumptions, providing an excellent foundation for developing a deep understanding of deductive reasoning. The advantages of abstract algebra in teaching the understanding of proofs is its ability to generalize specific mathematical concepts across various structures. For example, the group structure provides a framework for proving theorems about symmetries, permutations, and modular arithmetic. Formalizing abstract algebra poses significant challenges in formulating and proving different theorems, as many of them require different algebraic structures. For example, there is a group structure, but there are also cyclic and abelian groups which stresses the need for defining precise hierarchy of algebraic structures. The process of formalizing algebraic concepts forces students to engage with the fundamentals of the subject in a way that goes beyond traditional pen-and-paper proofs. In recent years, formalizing abstract algebra has become one of the important domains in interactive theorem proving.

The formalization of the abstract algebra book (Judson, 2020), like the formalization of geometry, was done with a group of students, using a shared file, over a period of 3 months. This time the group of students was smaller, which made it easier to monitor and regulate. The preparation was much more detailed, which led to a smoother execution.

4.4. Formalization of functional programming algorithms

Functional programming is a field of computer science that focuses on function composition and immutability as the main tool for developing algorithms. Due to its strong mathematical foundation, functional programming provides a natural link between algorithms and formal methods, making it an ideal candidate for the formalization of algorithms within interactive theorem provers. Formalizing functional algorithms provides reliability and ensures the correctness of software. On the other hand, it can deepen the understanding of algorithmic reasoning and mathematical proofs. Such formalized algorithms can then be exported to the source code of functional programming languages such as Haskell, OCaml, and Scala and used as a safety-critical module of a broader project.

Each student had the task of working on one functional pearl from “Pearls of Functional Algorithm Design” (Bird, 2010). The task consisted of: defining relevant functions for the algorithm, managing terminations of recursive functions, proving properties about functions, dealing with abstractions, proving correctness of the algorithm and optionally formalizing algorithm efficiency. Overall, students were successful in the following tasks: defining functions, proving termination, establishing properties about functions, and proving some of these properties, but sometimes they needed a lot of help to prove overall correctness of the algorithm. The reason

for this is partially that students lacked skills, experience and time, but most importantly, many materials providing functional algorithms lacked sufficient proofs of their correctness. In the future, it is worth considering grouping students in small groups of 2 or 3 students, such that the chance of success is maximized.

4.5. Formalization of algorithms and data structures

Algorithms and data structures are fundamental areas of computer science, and their formalization is essential for software reliability. Since interactive theorem provers provide a functional programming interface, formalizing imperative algorithms and data structures first requires formulating programs in a functional style, which can be a challenge in itself (Research Papers/Functional Pearls, 2024; Okasaki, 1998). Many imperative algorithms, when transformed into functional programs, worsen time complexity due to the immutability. Therefore, monadic frameworks (which provide a "do" syntax, which resembles imperative programming), on top of the interactive theorem provers, had to be developed (Lammich, 2019; Bulwahn et al., 2008).

The students had the task of formalizing a specific algorithm or data structure with certain properties. To do this, they first had to formulate an algorithm or data structure in the functional programming style. Afterwards, they had to formulate the properties, termination, efficiency, and data structure invariant. The results were very similar to those of the formalization of functional algorithms.

4.6. Student feedback after the course

In order to evaluate different aspects of the course, the survey was created and presented to students after the course was completed. The goal was to assess the level of understanding of different types of mathematical proofs and formal software verification. Over 20 students responded and all agreed on the successful organization of the course and praised the active approach of the professor and teaching assistants. For example, a quote from one student: "Lectures and exercises were coordinated and the pace of work was excellent. I especially liked that the exercises insisted on independent work, which, in my opinion, is a very good use of time."

Overall understanding of key concepts improved, such as understanding of the notion of a proof (here the median score increased from 7 to 9 on a scale of 1-10), natural deduction rules and applications, proof by cases, and proof by induction (here all students agreed that they had reached a high level of comprehension of these concepts after the course). When asked whether this course could be introduced in earlier years of undergraduate studies, and whether it could lead to increasing the level of understanding the mathematical subjects, over 50% of students agreed and 30% of students had a neutral stance. One of the students even suggested: "To connect the course *Discrete mathematics* with the basis of interactive theorem proving since mathematical induction is very relevant and natural to demonstrate in computer science. Proofs using the counter position are often used in courses covering *Analysis*, but they are taken for granted. It only came naturally to me after this course, and it could have been much earlier if some version of this course existed beforehand."

5. Conclusions and recommendations

In the general framework, we propose the idea of a structured use of interactive theorem provers to improve students' proof skills. In the mathematical framework, we show that a careful selection of a good set of basic mathematical theorems and a good set of computer algorithms can improve students' reasoning skills in specific domains.

We have proposed a thoughtful selection of a set of basic mathematical problems and a set of advanced specific domain problems that can improve students' reasoning skills in a particular area. In the first stage, we used a range of most common techniques used in elementary mathematics. In the second stage, we focused on correctness of algorithms. Even in this part of the course, the emphasis is on the logical conclusions and observation of the properties of the specific algorithm.

The proofs are built up step by step as the students discover the necessary properties in a kind of backward reasoning manner starting from the final goal.

We have introduced the reader to the implementation of the proposed approach through the course "Introduction to interactive theorem proving", offering detailed insight into the methodology of the course and the organizational aspect of the course. Course covers different areas of mathematics and computer science, giving students two perspectives of theorem proving that are different at the first glance, but fundamentally come from the same background. Proving the correctness of an algorithm is essentially the same as the proof of a corresponding theorem.

A more complex implementation of the approach was initially carried out as part of the doctoral course, but the number of interested students was small. It is noteworthy that when the undergraduate course presented in this paper was offered, students' interest was so great that there are more than 50 students each year, which encouraged the professors to formulate various project topics relevant to different areas of mathematics and computer science.

We noticed that mathematical subjects have certain advantages over the verification of algorithms, due to the fact that most mathematical resources are already provided with detailed proofs, and algorithms are rarely followed by a detailed analysis of their correctness. Large mathematical projects have been challenging to manage and in subsequent years we plan to divide students into smaller groups with more specific goals and directed guidance that will lead to more focused contributions to general formalized knowledge.

Furthermore, we have already shown the applicability of this approach and that the subject areas can be tuned in by teachers through the various formalization projects that have already been carried out by students and teachers.

Future research will focus on developing surveys to assess students' knowledge both before and after they complete a course, with particular emphasis on understanding of proofs and their ability to construct it. The prior and post course assessments could test both understanding of the notion of proof and basic proof techniques, and would involve careful preparation of a set of theorems to assess students' ability to perform clear and justified proofs. It is important to note that these tests should be conducted in a familiar environment, such as pen-and-paper proofs, as students have no prior knowledge of interactive theorem proving. In this way, it would be possible to analyze how students' initial understanding of proofs correlates with their improvement over the course. These efforts would contribute to a deeper understanding of teaching and learning proofs via interactive theorem proving and could lead to improved courses in the future.

At the Faculty of Mathematics in Belgrade, students have mainly courses on different areas of mathematics in the first two years and practical or industry-related courses (depending on the module) in the last two years of study. Our recommendation, based on our experience and students' input, would be to introduce a course of this type possibly in earlier years of undergraduate study as part of the basic mathematical logic courses for institutions already familiar with interactive theorem proving. For institutions that do not use interactive theorem provers, we hope that this paper can motivate mathematicians to use interactive theorem provers. The community using interactive theorem provers in Serbia is very small. Expanding this community would certainly contribute to the development of critical thinking, logic, and long-term knowledge acquisition, which would benefit the entire mathematical community.

Acknowledgment

The authors are grateful to all students who took part in the course, because without them this journey would not have been so eventful and full of learning opportunities for both sides.

References

ALPHAPROOF AND ALPHAGEOMETRY TEAMS. (2024, July 25). *AI achieves silver-medal standard solving International Mathematical Olympiad problems*. Google DeepMind. Retrieved September 9, 2024, from https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

BIRD, R. (2010). *Pearls of Functional Algorithm Design*. Cambridge University Press.

BLANCHETTE, J. C., HASLBECK, M., MATICHUK, D., & NIPKOW, T. (2015). Mining the Archive of Formal Proofs. *Intelligent Computer Mathematics*, 3-17.

BIERE, A. (2024). *Lecture on Isabelle and Program Verification*. Retrieved September 15, 2024, from https://cca.informatik.uni-freiburg.de/isabelle/ss24/

BULWAHN, L., KRAUSS, A., HAFTMANN, F., ERKÖK, L., & MATTHEWS, J. (2008). Imperative functional programming with Isabelle/HOL. *Theorem Proving in Higher Order Logics: 21st International Conference*, *21*, 134-149.

BUZZARD, K. (2022). What is the point of computers? A question for pure mathematicians. *International Congress of Mathematicians*, *2*(2022), 578-688.

CHOU, S.-C. (1988). *Mechanical geometry theorem proving*. Springer Netherlands.

CHOU, S. C., & GAO, X. S. (2001). Automated reasoning in geometry. *Handbook of automated reasoning*.

DECHESNE, F., & NEDERPELT, R. (2012). N.G. de Bruijn (1918-2012) and his road to Automath, the earliest proof checker. *The Mathematical Intelligencer*, *34*(4), 4-11.

DE MOURA, L., & ULLRICH, S. (2021). *The Lean 4 Theorem Prover and Programming Language*. Retrieved September 15, 2024, from https://lean-lang.org

DJUKIĆ, D., JANKOVIĆ, V., MATIĆ, I., & PETROVIĆ, N. (2011). *The IMO Compendium: A Collection of Problems Suggested for The International Mathematical Olympiads* (2nd ed.). Springer.

DOWEK, G. (2015). *Computation, Proof, Machine: Mathematics Enters a New Age* (P. Guillot & M. Roman, Trans.). Cambridge University Press.

FLEURIOT, J. (2021). *Automated Reasoning course*. Retrieved November 8, 2024, from https://www.inf.ed.ac.uk/teaching/courses/ar/

GHILEZAN, S. (n.d.). *Teaching courses*. Retrieved November 8, 2024, from https://imft.ftn.uns.ac.rs/silvia/Teaching

GONZALEZ, S. T., JANIČIĆ, P., & NARBOUX, J. (2024). Automated Completion of Statements and Proofs in Synthetic Geometry: an Approach based on Constraint Solving. *arXiv preprint arXiv:2401.11898*.

HASLBECK, M. P. L., & WIMMER S. (2018). *Proving for Fun*. Retrieved September 15, 2024, from https://do.proof.in.tum.de/

HEATH, T. L. (1956). *The Thirteen Books of Euclid's Elements*. Dover Publications.

HILBERT, D. (1922). *Grundlagen der Geometrie*. Springer.

HOHENWARTER, M., & GEOGEBRA TEAM (2001). GeoGebra®, Retrieved September 15, 2024, from https://www.geogebra.org/

INRIA. (1984). The Coq Proof Assistant. Retrieved September 15, 2024, from https://coq.inria.fr

ITP (2023). Interactions of Proof Assistants and Mathematics [Summer school]. Retrieved September 15, 2024, from https://itp-school-2023.github.io/

IMO GRAND CHALLENGE (n.d.). Retrieved May 20, 2021, from https://imo-grand-challenge.github.io/

JACOBSEN, F. K., & VILLADSEN, J. (2023). On exams with the Isabelle proof assistant. *ThEdu@FLoC*. arXiv preprint arXiv:2303.05866.

LAMMICH, P. (2019). Refinement to imperative HOL. *Journal of Automated Reasoning*, *62*(4), 481-503.

JUDSON, T. W. (2020). *Abstract algebra: theory and applications*.

JUAN, V. L. AT. AL. (2017). *Coq @ Chalmers*. Retrieved November 8, 2024, from https://github.com/vlopezj/coq-course

LUČIĆ, Z. (1994). *Euklidska i hiperbolička geometrija*. Matematički fakultet i Grafiti.

MARIĆ, F. (2008). Formal verification of modern SAT solvers. *Archive of formal proofs*.

MARIĆ, F. (2015). A survey of interactive theorem proving. *Zbornik radova*, *18*(26), 173-223.

MARIĆ, F., & STOJANOVIĆ ĐURĐEVIĆ, S. (2020). Formalizing IMO problems and solutions in Isabelle/HOL. *ThEdu@IJCAR*. arXiv preprint arXiv:2010.16015.

MINH F. T., GONNORD L., & NARBOUX J. (2024). Research report - Proof assistants for teaching: a survey. *LCIS, Grenoble-INP*.

NARBOUX, J. (2006). Mechanical theorem proving in Tarski's geometry. *International Workshop on Automated Deduction in Geometry*, 139-156.

NARBOUX, J. (2021). *My experience using ITP for teaching maths*. Retrieved November 8, 2024, from https://leanprover-community.github.io/lt2021/slides/julien-lean_workshop_ITP_teaching.pdf

NIPKOW, T. (2012). Teaching Semantics with a Proof Assistant: No More LSD Trip Proofs. *Verification, Model Checking, and Abstract Interpretation: 13th International Conference*, 24-38.

NIPKOW, T., PAULSON, L. C., & WENZEL, M. (2002). *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer Berlin Heidelberg.

NIPKOW, T., PRÄHOFER, M., & HASLBECK, M. P. L. (2018). *Seminar - Formal Proof in Mathematics and Computer Science*. Retrieved September 15, 2024, from https://www21.in.tum.de/teaching/proof21/SS18/

NORELL, U. (2009). Dependently typed programming in Agda. *Proceedings of the 4th International Workshop on Types in Language Design and Implementation*, 230-266.

OKASAKI, C. (1998). *Purely functional data structures*. Cambridge University Press.

PAT (2023). *Proof assistants for teaching proof and proving* [Summer school]. Retrieved September 9, 2024, from https://pat2023.icube.unistra.fr/

PAULSON, L. C. (2020). *Interactive Formal Verification*. Retrieved September 15, 2024, from https://www.cl.cam.ac.uk/teaching/2021/L21/

PHAM, T., BERTOT, Y., & NARBOUX, J. (2011). A Coq-based library for interactive and automated theorem proving in plane geometry. *Computational Science and Its Applications-ICCSA 2011*, 368-383.

RESEARCH PAPERS/FUNCTIONAL PEARLS. (2024). *HaskellWiki*. Retrieved May 20, 2024, from https://wiki.haskell.org/Research_papers/Functional_pearls

SMULLYAN, R. M. (2008). *Logical Labyrinths* (1st ed.). CRC Press LLC.

STOJANOVIĆ ĐURĐEVIĆ, S. (2019). From informal to formal proofs in Euclidean geometry. *Annals of Mathematics and Artificial Intelligence*, *85*(2), 89-117.

STOJANOVIĆ ĐURĐEVIĆ, S. (2020). *UIDT [github repository]*. Retrieved August 1, 2024, from https://github.com/sanastojanovic/UIDT

STOJANOVIĆ ĐURĐEVIĆ, S., & MARIĆ, F. (2021). *Uvod u interaktivno dokazivanje teorema* [Lecture notes]. Retrieved August 1, 2024, from https://poincare.matf.bg.ac.rs/~sana/uidt/uidt.pdf

THEDU (2024). *Workshop on Theorem proving components for Educational software*. Retrieved September 9, 2024, from https://www.uc.pt/en/congressos/thedu/ThEdu24/

TRINH, T. H., WU, Y., LE, Q. V., HE, H., & LUONG, T. (2024). Solving olympiad geometry without human demonstrations. *Nature*, *625*(7995), 476-482.

WENZEL, M., PAULSON, L. C., & NIPKOW, T. (2008). The Isabelle framework. *Theorem Proving in Higher Order Logics: 21st International Conference*, 33-38.

WIEDIJK, F. (ED.). (2006). *The Seventeen Provers of the World: Foreword by Dana S. Scott*. Springer.