Aiding an Introduction to Formal Reasoning Within a First-Year Logic Course for CS Majors Using a Mobile Self-Study App

David M. Cerna[1], Martina Seidl[2], Wolfgang Schreiner[1], Wolfgang Windsteiger[2], and Armin Biere[1]

[1]Institute of Formal Methods and Verification, Johannes Kepler University
[2]Research Institute for Symbolic Computation, Johannes Kepler University

Abstract

In this paper, we share our experiences concerning the introduction of the Android-based self-study app AXolotl within the first-semester logic course offered at our university. This course is mandatory for students majoring in Computer Science and Artificial Intelligence. AXolotl was used as part of an optional lab assignment bridging clausal reasoning and SAT solving with classical reasoning, proof construction, and first-order logic. The app provides an intuitive interface for proof construction in various logical calculi and aids the students through rule application. The goal of the lab assignment was to help students make a smoother transition from clausal and decompositional reasoning used earlier in the course to inferential and contextual reasoning required for proof construction and first-order logic. We observed that the lab had a positive influence on students' understanding and end the paper with a discussion of these results.

1 Introduction

Logic plays a fundamental role in modern computer science and has even been referred to as the calculus of the subject. While this fact seems to be ubiquitously recognized, programming and other related topics remain at the forefront of the average computer science curriculum. J. A. Makowsky and A. Zamansky [30] reported on this undesirable fact as well as how courses on logic are slowly disappearing from the computer science curriculum at many universities.

Given the abstract nature of the subject, it is often thought of as an advanced/elective course to be taken late in the computer science curriculum. This is most often justified by folklore concerning the struggle students of computer science have with formal reasoning [22, 23] when introduced to the subject early in the curriculum. In its purest form, logic and formal reasoning require mathematical maturity and are better left to advanced courses, however, this train of thought ignores the methodological approaches based on logical reasoning which can have a great impact on how students approach problems in the core computer science curriculum. Important branches

of computer science such as symbolic artificial intelligence [35] are built upon fundamental aspects of logic. Furthermore, fields such as software and hardware verification, which are of industrial importance [28, 14, 16], rely on automated reasoning and the evaluation of logical formulas. To top it off, students' understanding of certain related topics, such as induction [37], seems to predict students' performance in related computer science topics such as recursion. It is highly likely similar correspondences may be found for other topics within software engineering and programming. If anything, formal reasoning is becoming more essential to practical computer science than the current outlook suggests.

At our university , an introductory logic course has been part of the computer science curriculum for some years now and has been added to the recently developed artificial intelligence curriculum. This introductory logic course is offered during the first semester of studies to provide a logical foundation essential to the rest of the curriculum. While the struggles of introducing formal reasoning to computer science students are still faced in every iteration of our course we have followed some of the advice already present in [23], i.e. providing educational tools to aid the learning process. Furthermore, rather than presenting a purely formal take of mathematical logic, we follow the approaches already outlined in works such as "Logic for Computer Science" by Steve Reeves and Michael Clarke [34] and "Mathematical Logic for Computer Science" by Mordechai Ben-Ari [5] which emphasizes topics most relevant to computer science students.

However, as we discuss in detail in Section 3, tools which directly aid the students through formal reasoning within formal calculi are still missing from the syllabus. In particular, the lack of such educational aids has been evident when transitioning from the propositional Module to the first-order Module where there is a significant drop in student performance year after year. We addressed this drop in student performance by introducing an optional lab assignment (Section 6) which required the use of the self-study tool AXolotl [15] (see Section 5) and provided practice with inferential reasoning and proof construction within the natural deduction calculus for propositional logic. Furthermore, the inclusion of the lab assignment within the course provided the opportunity to develop a case study concerning the use of mobile apps as a self-study tool for formal reasoning. In particular, we asked, as our main research question, whether such technology can help students make a smoother transition from clausal and decompositional reasoning used earlier in the course to inferential and contextual reasoning required for proof construction and first-order logic.

To answer our question, we compared the performance of students who participated in the lab assignment with students who did not and noted improvement in topics concerning proof construction. We discuss our methodology and results in Section 6. We conclude with a discussion of this case study, how we plan to further develop the course as well as related courses, and the role of tools like AXolotl in logic education for computer science students (Section 6.2 & 7).

2 Related Work

Investigating the effect computer-based self-study tools have on learning logic is not new, nor is the development of such tools. What we are considering in this work is the use of tutoring and self-study tools to bridge various approaches to formal reasoning we introduce in our course. As illustrated by the characterization introduced in [25], certain approaches may only emphasize certain aspects of the introduced logical oper-

ations and concepts. Before discussing the structure of the course and the application we developed to provide a bridge between two of the course's Modules, we will discuss related software and its uses.

Already in Fung *et al.* [23] computer-based educational tools were seen as a way to help students cope with learning formal reasoning. One of the earliest systems for teaching formal reasoning, Jape [9], provided students with an environment they can use to construct formal proofs. The developers discussed their experience with the software in the book "Using computers to learn logic: undergraduates' experiences" [1].

Since these early investigations, there have been many approaches to the teaching of formal reasoning as well as many case studies testing these approaches. For example, using interactive theorem provers as tutoring aides [31, 7], adding hint generation to a proof construction tools [4], as well as considering different visualizations/proof representation methods [11, 20, 6, 3, 2]. Much of this earlier work was summarized in a survey by Antonia Huertas [26] which covered tools for teaching logic developed before 2010.

Though this survey was comprehensive at the time, there has been much recent development in the area, for example *Carnap.io* [29] is a web-based interactive textbook developed for an introductory philosophical logic course. Similarly, Terrance Tao developed an interactive textbook [38] for understanding the logic behind mathematical theorems. The *Sequent Calculus Trainer* [21] is a derivation construction tool similar to earlier tools. It has a user-friendly interface as well as a hint-engine powered by the Z3 SMT solver [19]. Similarly, for the natural deduction calculus, there is the web-based application *NaDeA: Natural Deduction assistant* [40].

Many of the above-mentioned tools use a standard proof representation for proof construction. However, *The Incredible Proof Machine* [10] instead provides a more circuit-like proof construction and provides a more game-like interface. While many of the earlier tools for logic tutoring where either web-based or computer-based, the rise in mobile technology provides a new approach to logic tutoring software. Two existing apps for logic tutoring are Peanoware [39] which provides a simple interface for visualizing derivations in natural deduction and *Natural Deduction* [24] for proving statements. The app *Logic++* [41] may also be used to prove statements and transform formal statements into various normal forms, it can be seen as a kind of logical calculator.

While there exists a plethora of tutoring and self-study software to choose from, for the most part, the software is web-based or computer-based. For our lab assignment, we wanted to provide students with a system that is more flexible, i.e. always in their pocket, to practice with. Our future goal with this project is a game-like logic tutor that can be played anywhere. Furthermore, while interactive proof systems like Coq [8] provide a flexible formalism and have been used as tutoring aids, they are also quite complex. our goal is to develop a simple logic tutor which allows one to restrict the formalism used by the students and also allows the instructor to define more specialized rules. This is usually not possible with tutors like the *Sequent Calculus Trainer* [21].

3 Course Structure

Our introductory logic course is scheduled as a mandatory first-semester course of both the Computer Science and Artificial Intelligence curriculum of our university. As a consequence, all incoming students majoring in these subjects, roughly 400 students, must take the course. Furthermore, prior knowledge within the group concerning logic

varies greatly. Thus, to accommodate such a large heterogeneous audience we developed a course structure that provides various avenues for attaining a satisfactory completion of the course as well as timely feedback. Students can and do take advantage of the flexible course structure.

The course takes place over 12 weeks with one lecture block per week. The lecture block consists of a lecture introducing a new topic (roughly $\sim$ 90 minutes in duration), followed by a mini-test (roughly $\sim$ 15 minutes in duration) testing student's understanding of the previous week's material, followed by an exercise session (roughly $\sim$ 45 minutes in duration) providing exercises and solutions based on the topics introduced earlier during the lecture. The exercises provided during the exercise session are practice questions for next week's mini-test. Exercises are only partially solved during the exercise session, thus leaving students with a few exercises for self-study. These remaining exercises are not graded.

Before going into detail concerning lab assignments and weekly challenges, let us first consider the organization of the course in terms of topics covered. The course is split into three distinct parts, we refer to these parts as *Modules*. Module 1 covers propositional logic from the perspective of clausal reasoning and SAT solving. Module 2 covers inferential reasoning, proof construction, and first-order logic: syntax and semantics. Module 3, the shortest of the Modules, covers satisfiability modulo theories (SMT) and its applications. Module 1 constitutes the first 4 weeks of the course, Module 2 the next 6 weeks, and Module 3 the last two weeks.

In addition to mini-tests, which are mandatory and occur once a week, each Module has several associated lab assignments dependent on the length, i.e. Module 1 & 3 have one lab, while Module 2 has two labs. Lab assignments are optional and can be used to replace the grade a student received on a mini-test within the same Module as the lab assignment. For example, lab assignment 1 can only be used to replace the grade received on one of the first four mini-test (the mini-test of Module 1). Lab assignments typically go beyond the material introduced during the lecture and use software illustrating some of the more practical aspects of the topics covered. For students who find the pace of the lecture tiresome, lab assignments provide the opportunity to see aspects of the subject they most likely haven't encountered. Furthermore, for students who are having quite some difficulty with the course, lab assignments also tend to provide a completely different perspective than the lectures, thus they can aid a student's understanding of the subject. Similarly, weekly challenges, which can be thought of as mini-labs, are offered weekly and can be used to improve a student's grade on the mini-test from the same week as the weekly challenge by up to 20%. Typically, the use of software tools is required.

In this paper we will only concern ourselves with the software used during the first lab assignment, for the sake of completeness, we discuss the other software used during the course. Given that the main focus of Module 1 is propositional logic, in particular, satisfiability (SAT) solving, the weekly challenges mainly consist of encoding problems within the language of propositional logic to have a SAT solver provide a solution, or using a SAT solver to test whether a problem is satisfiable or unsatisfiable. In particular, one of the weekly challenges concerns the encoding and solving of an instance of Sudoku. Similarly, when introduction of SMT [17], students are given problems like Sudoku but are guided to construct more elaborate encodings with the help of the expressive language a typical SMT solver provides. For example, student may be asked to construct an SMT instance for the factoring large numbers.

During Module 2, rather than using existing software such as SAT and SMT solvers, more educationally-oriented software was developed and used to provide illustrating

examples to the students. For example, RISCAL [36] is a language and associated software system for the formal modeling of mathematical theories and algorithms in first-order logic. This system was used to aid students when judging the correctness of a given formalization as well as when considering if a given formalization completely characterizes a given informal statement. In addition to RISCAL, The interface extension Theorema [12] of Mathematica [27] provided students with an environment to manipulate, construct, and analyze mathematical proofs. For the most part, the system was used for relatively simple proofs and proofs by induction. Each tool was used for half of the weekly challenges offered in Module 2 and one of the lab assignments.

Given that AXolotl [15] has particular importance to the case study we discuss in Section 6, we will discuss it in detail in Section 5.

4 Motivating Educational Scenarios

Before discussing AXolotl, we provide an extended description of the educational scenario we are presented with at the transition between the Module 1 and Module 2. We highlight particular points that motivated the development of AXolotl and the design of our lab assignment.

As we mentioned earlier, we are investigating what impact the introduction of our logic self-study application has on student performance. In particular, how well students transition from the clausal and decompositional reasoning of Module 1 to the inferential reasoning introduced in Module 2.

Even before the introduction of quantification, students have trouble interpreting formal statements in a meaningful way and struggle to develop a proof strategy based on the identified interpretation of the said statement. Too often students rush to apply exhaustive decomposition as was done using the tableaux-like [18] sequent calculus [13] provide in Module 1. Contrary to this approach, many of the exercises provided to students (in the later Modules) can be solved simply by application of the "right" inference rule. Why that inference rule is the right one is usually clear from the formal statement itself and the provided context. One of the goals of AXolotl is to help students identify the "right" inference rules to apply and aid them through the application of the inference rules, i.e. help students avoid inconsistent or incorrect application of rules.

While the sequent calculus introduced in Module 2 shares some of the decompositional rules introduced in Module 1, clausal forms of formal statements are no longer exclusively used and additionally, non-decompositional rules are added. For example, *Modus Ponens* as well as *Modus Tollens* are added. For those who are knowledgeable about sequent calculi and related systems, the introduced sequent calculus allows at most one formula on the right side of the "turnstile" $\vdash$, but with the addition of the above-mentioned rules and a few others, the calculus still captures classical logic. One can consider the introduced calculus as a hybrid between a natural deduction calculus [32] and a sequent calculus.

Approaching formal proof construction using a variety of calculi and inference rules illustrates that there is an underlying meaning of the statements and operators students are introduced to which is only partially captured by operational use of a particular calculus/set of inference rules. During Module 1, students tend to see a derivation in a particular calculus as analogous to the statement itself. This belief is particularly evident when students are asked to use the more complex calculus introduced in Module 2 on statements which are not in clausal normal form. For example, it

comes as a surprise to the students that implication can be treated inferentially without switching to its disjunctive form.

AXolotl provides a problem input language that allows one to adjust the set of inference rules which may be used to solve a problem. This allows an instructor to guide students towards alternative derivations of familiar formal statements using unfamiliar inference rule sets and thus provide additional insight into the meaning of the statement and the standard logical operators.

5 AXolotl: Logic Self-Study App

In this section, we first outline the type of reasoning students are introduced to when using AXolotl and then discuss the system design and usage.

5.1 Rule Based reasoning in AXolotl

The scope of AXolotl lies between propositional and first-order logic, essentially the type of problems which may be formulated and solved are of the quantifier-free fragment of first-order logic. Rather than being a derivation construction system for this formal language, AXolotl mainly focuses on introducing students to abstract reasoning over the language of propositional logic. In some sense, rather than asking students to prove something within a given calculus, we are asking them to describe how a given statement follows from a provided set of rules. This is something beyond a typical introduction to propositional logic in that it provides preparation for reasoning in first-order where this type of rule-based reasoning is essential for dealing with theories of educational importance.

Some of the less intuitive rules of natural deduction and Hilbert systems are presented to the students as a type of variable instantiation where the context is made explicit. This context is essentially the current proof situation. Later in this section, we give an example within AXolotl of this type of rule application. Educationally, what we are trying to introduce to students is the mechanism behind the calculi they have seen so far in the course.

The logical core may be thought of as follows: we have a proof "situation" consisting of a set of "expressions" $E_1, \cdots, E_n$ where each expression is an arbitrarily nested application of "function" symbols to "constants". In the initial proof situation, there is typically only one such expression. We have inference rules of form $\Delta, \cdots \Rightarrow \Delta, \cdots$, where Δ denotes an arbitrary set of expressions and $\cdots$ denotes an expression "pattern" that may contain "variables". Note that the variables on the left and right side of $\Rightarrow$ need not match, i.e. new variables may be introduced. On the left-side only a single pattern may occur, on the right side, there may be any number of patterns. A rule with zero patterns on the right side is an "axiom".

For example, the axiom inference rule of the sequent calculus and implication elimination of natural deduction would be encoded as follows in AXolotl:

$$\Delta, (x, y \vdash x, z) \Rightarrow \Delta \qquad \Delta, (z \vdash x) \Rightarrow \Delta, (z \vdash y), (z \vdash x \to y)$$

An inference rule can be "applied" to a proof situation if the "$\cdots$" on the left side of the rule matches (by substituting the variables by the sub-expressions) one of the expressions in the situation; the application then replaces the expression by the "$\cdots$" on the right side of the rule. Here in the "$\cdots$" pattern, all variables which also occur on the left side of the rule are replaced by the expressions determined by the matching of the

left side; for all other variables arbitrary new expressions (e.g., chosen by the human) may be substituted. Thus, by application of an axiom, expressions are removed. The proof is complete when the situation is "empty", in other words, no expressions are left.

As mentioned earlier in this section, the didactic goal of AXolotl is to introduce students to abstract concepts such as "rules", "patterns", and "matching" which they used unknowingly during the Module 1 of the course. Furthermore, in preparation for the rest of the course, we want to teach students how to select the appropriate rule to apply to the given situation as well as how to determine if an inference matches the given situation. In the following subsection, we demonstrate, by example, how this reasoning process is implemented in AXolotl.

5.2 AXolotl by Example

The current version of AXolotl is implemented for Andriod 9.0 (Pie) with the minimum system requirements being Android 6.0 (Marshmallow), i.e. the minimum acceptable Android API is 23 and the app is compatible with all APIs up to 29. While this accommodates the majority of mobile device users, we are aware that other systems, such as IOS, are in use. In particular, when designing the lab assignment we presented students with alternative solutions such as Android emulators. These alternatives provided an equally adequate environment for using the app, though we nonetheless plan to release a version of AXolotl for other major mobile operating systems.

The current release contains a library of around 70 pre-installed problems which can be accessed through the main menu (Figure 1). Additionally, from the main menu, one can change the font size, turn "observation mode" on or off, view the current proof situation, or view the current problem (if there is one). Observation mode effects inference rule application, essentially every time an inference rule is applied to the current proof situation AXoolotl shows, in detail, how the proof situation changes. If students are to be presented with problems that are not in the pre-installed library, a load problem option can be found in the Options menu (Figure 2). The Options menu also contains a tutorial.

Let us consider the statement of classical propositional logic $p \to \neg\neg p$, i.e. the double negation of a given formula is equivalent to the formula. Proving this statement with the standard inference rules of natural deduction was one of the exercises included in the lab assignment. The rules and the proof are displayed in Figure 3 as they would be displayed in AXolotl's proof viewer.

When the file containing the problem statement is opened in AXolotl the student is confronted by the situation illustrated in Figure 4. The allowed inference rules are listed below the mascot (this list can be scrolled through) and the proof situation is located to the right of the mascot (can contain more than one goal), i.e. the current list of goals (or open branches in the proof tree). If one long clicks on an inference rule a new display is opened pretty-printing the rule similarly to the rules displayed in Figure 3.

Once an inference rule and a goal from the proof situation are selected swiping right applies the chosen rule to the goal if it is applicable. If it is not the mascot warns that the chosen inference cannot be applied. If the rule can be applied and observation mode is activated, the matching between the rule and the goal, computed by AXolotl, is displayed to the student, otherwise, the proof situation is updated and no additional information is displayed.

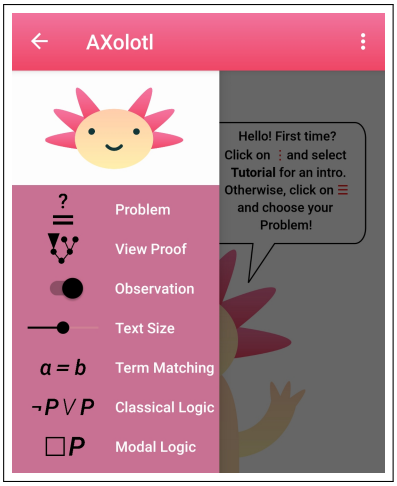

Figure 1: Main menu of AXolotl.

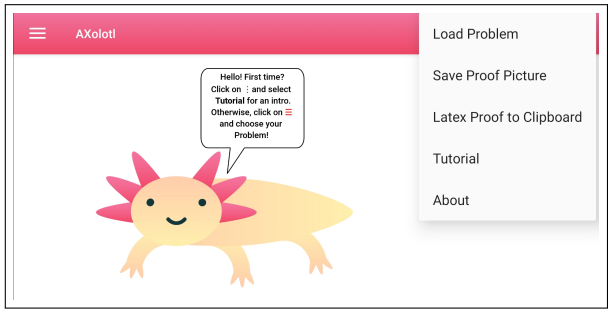

Figure 2: Options menu of AXolotl.

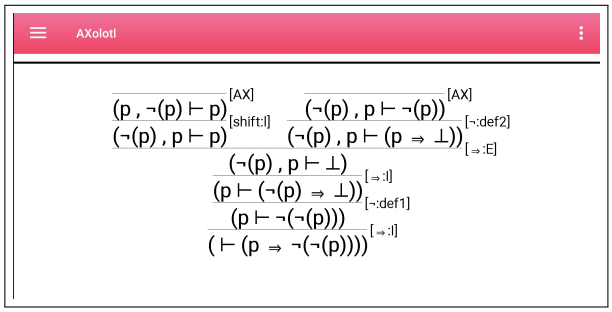

Figure 3: Proof of $p \leftrightarrow \neg\neg p$ displayed in AXolotl.

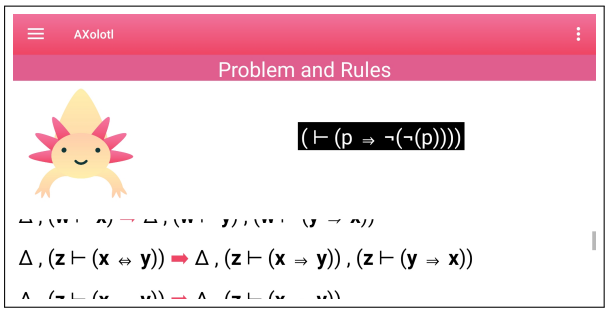

Figure 4: Proof situation upon loading problem from file.

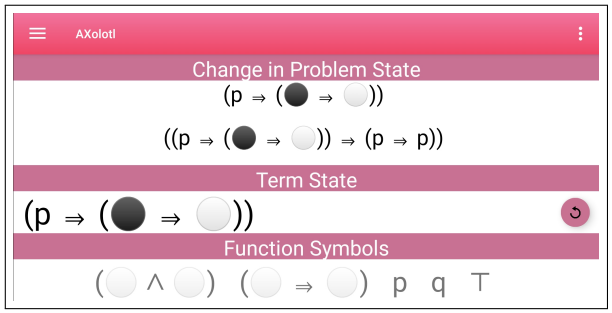

Figure 5: When a rule requires the student to provide a term to update the proof situation this window is displayed.

In some cases, such as the implication elimination inference rule, the student needs to provide a term. In such cases, a new window is opened displaying what changes will occur to the proof situation if the student chooses a particular term (See Figure 5). Terms are constructed using the calculator like interface found in the bottom section of the screen displayed in Figure 5. This is done to avoid typographical errors.

Note that the proof display (See Figure 3) can be opened at any point, even when the proof is partially constructed. In such cases, the partial proof is displayed with "?" indicating open branches. Another point of educational importance we would like to highlight is that rule application is handled by the app rather than by the student. This is done to avoid erroneous rule application. Observation mode is present to aid students through rule application even though it is done automatically by the app.

6 Lab assignment

The lab assignment was provided to students immediately after mini-test 4, the last mini-test of the Module 1. Two weeks were provided during which students could read the provided material on the natural deduction calculus and use AXolotl. If a student felt confident enough, they could complete the lab during this time.

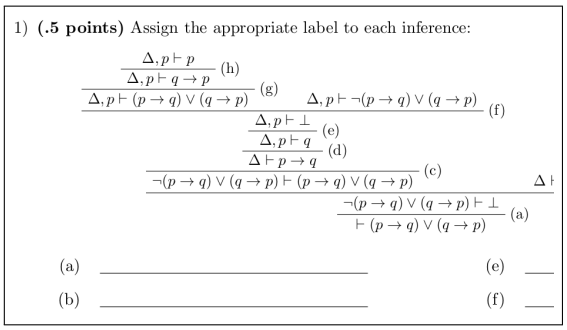

Figure 6: Lab Question: Fill in the rule.

6.1 Methodology

During these two weeks, the students were introduced to the syntax and semantics of first-order logic as part of Module 2 of the lecture. Discussion of proof construction and extensions of the calculus introduced during Module 1 was left till after the two weeks allotted to lab assignment. The first mini-test concerning proofs and derivations took place a week after the lab period ended.

On the last day of this period we held a joint session (∼2.5 hours) where, after giving a short introduction to the natural deduction calculus and AXolotl, students could work on the lab assignment, either alone or in a group. We, the instructors, were present for the entire joint session to answer questions and to deal with problems that may arise with the use of the app.

As mentioned earlier when discussing the course, the lab assignments are optional. This implies that students who feel they are doing well enough in the Module 1 may not feel obliged to participate. Given that many students find the Module 1 to be the least troubling part of the course few tend to participate in the lab assignment. To further exacerbate things, the first lab assignment can only replace a mini-test from Module 1. Out of the remaining 270 students (students who did not drop out by this point) participating in the course only 23 students participated in the first lab assignment. Only one student turned in the lab early.

The main intention of the lab assignment was to introduce students to a more abstract way of thinking about inference rules and to improve their ability to construct proofs. Being that the lab introduced students to a calculus which is, in all likelihood, unfamiliar to them, we provided them with example proofs to facilitate their understanding. For example, the question in Figure 6 was designed to help students familiarize themselves with the new rules.

We also provided students with proofs of statements using one logical operator and asked them to prove a statement with the same meaning with a different logical operator. For example, we gave them a proof of $A \to (B \to C)$ and ask them to prove $(A \wedge B) \to C$, the latter being a little more difficult when using natural deduction.

The most insightful series of questions included in the lab assignment concerns the various ways of introducing *excluded middle* into a natural deduction calculus (See Figure 7). We did not introduce students to the concept of *intuitionistic logic* and the intermediate logics (Figure 6 is a proof of the linearity axiom of Gödel logic [33]), but rather ask them to prove equivalent formulations of *excluded middle* using a variety of inference rules and assumptions.

While most students who participated in the lab assignment struggled to complete every question, the majority completed all the questions discussed above. The remaining questions concerned proof construction and provided little insight beyond practice.

6.2 Results and Threats to Validity

While we mentioned our hypothesis abstractly at earlier points in the paper, we were not able to concretely formulate it until we introduced the lab assignment and how it fits into the course.

Essentially, we hypothesized that students who participated in the lab would perform better on mini-test 7, i.e. the mini-test which took place right after the joint session, then they performed on mini-test 4 (concerning proof construction using clausal reasoning). In particular, those who participated in the lab assignment would perform better on mini-test 7 than students who performed similarly on mini-test 4 and did not participate in the lab.

To test this hypothesis we considered only students who took both mini-test 4 and 7. In total 236 students participated in both mini-test and all 23 students who participated in the lab took both tests. Interestingly, the difference between the averages on the mini-test for those who did not participate in the lab was +7 points out of 100, while for those who did participate it was +13 out of 100.

However, these differences are do not show reasonable statistical significance (only 60% confidence that the averages differ according to t-test analysis). However, if (we instead of considering all students who took both tests) we consider those who did poorly (between 40% and 60%) on mini-test 4 we observe that a large number of students in this group jumped to perfect scores on mini-test 7. Eleven of students in this grade range participated in the lab and 62 did not. Of the 11 students 5 attained a perfect score while of the 62, only 17 attained a perfect score. While this is intriguing, the results are only of borderline statistical significance.

The low significance is most likely a side-effect of the low participation in the lab when compared to the number of students who participated in the lecture. Furthermore, what can also threaten the validity of our results is the vast amount of opportunities provided to students to improve their understanding. During the period of the lab assignment, students were assigned bonus exercises as well as a light introduction to the abstraction power of first-order logic. Many students not participating in the lab may have taken advantage of these opportunities and thus end up obfuscating the effect of the lab. Though, the fact that students seemed to benefit from the lab, as well as the use of AXolotl points to a nonetheless positive educational impact. We expect that our future investigations and experiments will help elucidate the beneficial aspects of our educational software and approach.

7 Future Work

We plan to further integrate the concepts and tools used in the Lab assignment into the lecture and further study their impact on student understanding. Also, we plan to perform similar case studies concerning the use of bonus exercises in the course and other aspects to attain a better understanding of the true impact of the lab assignment and our educational tool on student understanding of formal reasoning and derivation construction.

b) The statement $(\neg q \to \neg p) \to (p \to q)$ it is referred to as the *contraposivity axiom* and can be proven equivalent to the *principle of excluded middle* $\neg p \vee p$. Proving $(\neg q \to \neg p) \to (p \to q) \vdash (\neg q \vee q)$ using the sequent calculus is quite simiple, is it provable using the rule set of *ContrapositiveProblem2.txt* or is some assumption missing? If an assumption is missing what is it? (see the file *ContrapositiveProblem3.txt*)

c) What about adding

```
Rule: 1 ⊢(cons(¬(x),z),⊥) ⊢(z,x) [⊥]
```

Figure 7: Example of questions concerning *excluded middle*.

Acknowledgements

Supported by the LIT LOGTECHEDU project and the LIT AI Lab both funded by the state of upper Austria.

References

[1] James Aczel, P. Fung, Richard Bornat, Martin Oliver, Tim O'Shea, and Bernard Sufrin. *Using computers to learn logic: undergraduates' experiences*, pages 875–882. 10 1999.

[2] Sandra Alves, Maribel Fernández, and Ian Mackie. A new graphical calculus of proofs. In Rachid Echahed, editor, Proceedings 6th International Workshop on *Computing with Terms and Graphs,* Saarbrücken, Germany, 2nd April 2011, volume 48 of *Electronic Proceedings in Theoretical Computer Science*, pages 69–84. Open Publishing Association, 2011.

[3] Serge Autexier, Christoph Benzmüller, Dominik Dietrich, Andreas Meier, and Claus-Peter Wirth. A generic modular data structure for proof attempts alternating on ideas and granularity. In Michael Kohlhase, editor, *Mathematical Knowledge Management*, pages 126–142, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[4] Tiffany Barnes and John Stamper. Toward automatic hint generation for logic proof tutoring using historical student data. In Beverley P. Woolf, Esma Aïmeur, Roger Nkambou, and Susanne Lajoie, editors, *Intelligent Tutoring Systems*, pages 373–382, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[5] Mordechai Ben-Ari. *Mathematical Logic for Computer Science*. Springer, London, 3rd edition, 2012.

[6] Christoph Benzmüller, Dominik Dietrich, Marvin Schiller, and Serge Autexier. Deep inference for automated proof tutoring? In Joachim Hertzberg, Michael Beetz, and Roman Englert, editors, *KI 2007: Advances in Artificial Intelligence*, pages 435–439, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[7] William Billingsley and Peter Robinson. Student proof exercises using mathstiles and isabelle/hol in an intelligent book. *Journal of Automated Reasoning*, 39(2):181–218, Aug 2007.

[8] Sebastian Böhne and Christoph Kreitz. Learning how to prove: From the coq proof assistant to textbook style. In *Proceedings of the 6th Int. Workshop on Theorem Proving components for Educational software, ThEdu@CADE'17*, pages 1–18, 2017.

[9] R. Bornat and B. Sufrin. Animating formal proof at the surface: The jape proof calculator. *The Computer Journal*, 42(3):177–192, Jan 1999.

[10] Joachim Breitner. Visual theorem proving with the incredible proof machine. In Jasmin Christian Blanchette and Stephan Merz, editors, *Interactive Theorem Proving*, pages 123–139, Cham, 2016. Springer International Publishing.

[11] Krysia Broda, Jiefei Ma, Gabrielle Sinnadurai, and Alexander Summers. Pandora: A Reasoning Toolbox using Natural Deduction Style. *Logic Journal of the IGPL*, 15(4):293–304, 08 2007.

[12] Bruno Buchberger, Tudor Jebelean, Temur Kutsia, Alexander Maletzky, and Wolfgang Windsteiger. Theorema 2.0: Computer-Assisted Natural-Style Mathematics. *JFR*, 9(1):149–185, 2016.

[13] S.R. Buss, editor. *Handbook of Proof Theory*, volume 137. Elsevier, 1 edition, 1999.

[14] Cristiano Calcagno, Dino Distefano, Jérémy Dubreil, Dominik Gabi, Pieter Hooimeijer, Martino Luca, Peter O'Hearn, Irene Papakonstantinou, Jim Purbrick, and Dulma Rodriguez. Moving fast with software verification. In *Proceedings of the NASA Formal Methods Symposium, NFM'15*, volume 9058 of *LNCS*, pages 3–11. Springer, 2015.

[15] David M. Cerna, Rafael P.D. Kiesel, and Alexandra Dzhiganskaya. Axolol. Google Play Store, 2019. https://play.google.com/store/apps/details?id=org.axolotlLogicSoftware.axolotl.

[16] Byron Cook. Formal reasoning about the security of amazon web services. In *Proceedings of the 30th Int. Conference on Computer Aided Verification, CAV'18*, volume 10981 of *LNCS*, pages 38–47, Cham, Switzerland, 2018. Springer.

[17] Denis Cousineau, Damien Doligez, Leslie Lamport, Stephan Merz, Daniel Ricketts, and Hernán Vanzetto. TLA+ proofs. In *Proceedings of the Int. Conference on Formal Methods, FM'12, Paris*, volume 7436 of *LNCS*, pages 147–154, Berlin, Germany, 2012. Springer.

[18] Marcello D'Agostino, Dov M. Gabbay, Reiner Hhnle, and Joachim Posegga, editors. *Handbook of Tableau Methods*. Springer Netherlands, 1999.

[19] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[20] Ewen Denney, John Power, and Konstantinos Tourlas. Hiproofs: A hierarchical notion of proof tree. *Electronic Notes in Theoretical Computer Science*, 155:341 – 359, 2006. Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI).

[21] Arno Ehle, Norbert Hundeshagen, and Martin Lange. The sequent calculus trainer with automated reasoning - helping students to find proofs. In *Proceedings 6th International Workshop on Theorem proving components for Educational software, ThEdu@CADE 2017, Gothenburg, Sweden, 6 Aug 2017.*, pages 19–37, 2017.

[22] P. Fung, T. O'Shea, D. Goldson, S. Reeves, and R. Bornat. Computer science students perceptions of learning formal reasoning methods. *International Journal of Mathematical Education in Science and Technology*, 24(5):749–759, 1993.

[23] P. Fung, T. O'Shea, D. Goldson, S. Reeves, and R. Bornat. Why computer science students find formal reasoning frightening. *Journal of Computer Assisted Learning*, 10:240–250, 1994.

[24] Jukka Häkkinen. Natural deduction. proof generator. proof checker., 2018.

[25] Casey Hawthorne and Chris Rasmussen. A framework for characterizing students' thinking about logical statements and truth tables. *International Journal of Mathematical Education in Science and Technology*, 46, 04 2015.

[26] Antonia Huertas. Ten years of computer-based tutors for teaching logic 2000-2010: Lessons learned. In *Proceedings of the 3rd Int. Congress Conference on Tools for Teaching Logic, TICTTL'11*, pages 131–140, Berlin, Heidelberg, 2011. Springer.

[27] Wolfram Research, Inc. Mathematica, Version 11. Champaign, IL, 2018.

[28] Roope Kaivola, Rajnish Ghughal, Naren Narasimhan, Amber Telfer, Jesse Whittemore, Sudhindra Pandav, Anna Slobodová, Christopher Taylor, Vladimir A. Frolov, Erik Reeber, and Armaghan Naik. Replacing testing with formal verification in intel coretm i7 processor execution engine validation. In *Proceedings of the 21st Int. Conference on Computer Aided Verification*, volume 5643 of *LNCS*, pages 414–429. Springer, 2009.

[29] Graham Leach-Krouse. Carnap: An open framework for formal reasoning in the browser. In *Proceedings 6th International Workshop on Theorem proving components for Educational software, ThEdu@CADE 2017, Gothenburg, Sweden, 6 Aug 2017.*, pages 70–88, 2017.

[30] J. A. Makowsky and A. Zamansky. Keeping logic in the trivium of computer science: A teaching perspective. *Formal Methods in Systems Design*, 51(2):419–430, November 2017.

[31] Hendriks Maxim, Cezary Kaliszyk, Femke Raamsdonk, and Freek Wiedijk. Teaching logic using a state-of-art proof assistant. *Acta Didactica Napocensia*, 3, 06 2010.

[32] Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Dover Publications, 1965.

[33] Norbert Preining. Gödel logics – a survey. In Christian G. Fermüller and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 30–51, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[34] Steve Reeves and Michael Clarke. *Logic for Computer Science*. Addison-Wesley, Boston, MA, USA, 1990.

[35] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach, Third Int. Edition*. Pearson Education, Upper Saddle River, NJ, USA, 2010.

[36] Wolfgang Schreiner. Theorem and Algorithm Checking for Courses on Logic and Formal Methods. In Pedro Quaresma and Walther Neuper, editors, *Post-Proceedings ThEdu'18*, volume 290 of *EPTCS*, pages 56–75, 2019.

[37] Thérèse Smith and Robert McCartney. Computer science students' concepts of proof by induction. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling 14, page 5160, New York, NY, USA, 2014. Association for Computing Machinery.

[38] Terry Tao. Qed, 2018.

[39] Laurent Théry. Peanoware-natural deduction. Google Play.

[40] Jørgen Villadsen, Andreas Halkjær From, and Anders Schlichtkrull. Natural deduction assistant (nadea). In Pedro Quaresma and Walther Neuper, editors, Proceedings 7th International Workshop on *Theorem proving components for Educational software,* Oxford, United Kingdom, 18 july 2018, volume 290 of *Electronic Proceedings in Theoretical Computer Science*, pages 14–29. Open Publishing Association, 2019.

[41] Yale Weiss. Logic++, 2016.