Lessons of Teaching Formal Methods with Isabelle

Jacobsen, Frederik Krogsdal; Villadsen, Jørgen

Published in:
Proceedings of Isabelle Workshop 2022

Publication date:
2022

Document Version
Peer reviewed version

Lessons of Teaching Formal Methods with Isabelle

Frederik Krogsdal Jacobsen and Jørgen Villadsen

Technical University of Denmark, Kongens Lyngby, Denmark

Abstract

We present our recent experiences teaching two courses on formal methods which use Isabelle/Pure and Isabelle/HOL. We explain our overall approach and our experiences with implementing beginner-friendly tools to help students understand how Isabelle works. We also describe the issues that we often see students struggle with when attempting to learn how to program and prove theorems using Isabelle, and suggest ideas for potential solutions to some of these concerns.

1 Introduction

At the Technical University of Denmark, we currently teach two courses on logic and automated reasoning using Isabelle as our main tool. The first course is a BSc level course on Logical Systems and Logic Programming. This course serves as an introduction to logic as well as logical programming in Prolog, and we use Isabelle to teach the logic part of the course. The second course is a MSc level course on Automated Reasoning. This course serves as an introduction to automatic and interactive theorem proving, and we use Isabelle to formalize and exemplify nearly all concepts taught in the course.

Figure 1 shows the (mandatory) courses leading to our BSc course on Logical Systems and Logic Programming. It also shows our MSc course on Automated Reasoning and other courses with our BSc course among the prerequisites.

We have previously described the curriculum of our courses and how they fit into the overall program in [12]. The present paper describes the experiences since then. 77 students took the BSc course in autumn 2021 and 43 students the MSc course in spring 2022. More information about the courses is available here:

`https://courses.compute.dtu.dk/02156/` `https://courses.compute.dtu.dk/02256/`

The BSc course has run for many years, but has recently become more Isabelle-oriented. The MSc course has run for a few years, and has been designed around Isabelle from the beginning. We have developed a number of tools and associated reading material to support a gentle introduction to logic in Isabelle, and additionally use the "Programming and Proving in Isabelle/HOL" tutorial [7] as a main component of our MSc course.

Our approach is to start by introducing logic in a system which is much simpler, but thus also much less overwhelming, than Isabelle. From this, we progress through systems that gradually begin to look more like the "real" Isabelle experience, until we arrive at Isabelle/HOL. Additionally, we introduce some basic proof methods such as induction and rule inversion. Some of this has to happen in parallel due to time constraints, but the pacing of each component of the courses is such that students can constrain themselves to learning about a few aspects of using Isabelle at a time.

Overall our approach is different than previous uses of the Agda, Coq and Isabelle proof assistants in computer science courses [5,6,8,9]. These courses generally start by introducing the proof assistant, and then build up the complexity through repeat exposure to concepts within the proof assistant itself. It is our experience, however, that this approach is very difficult to

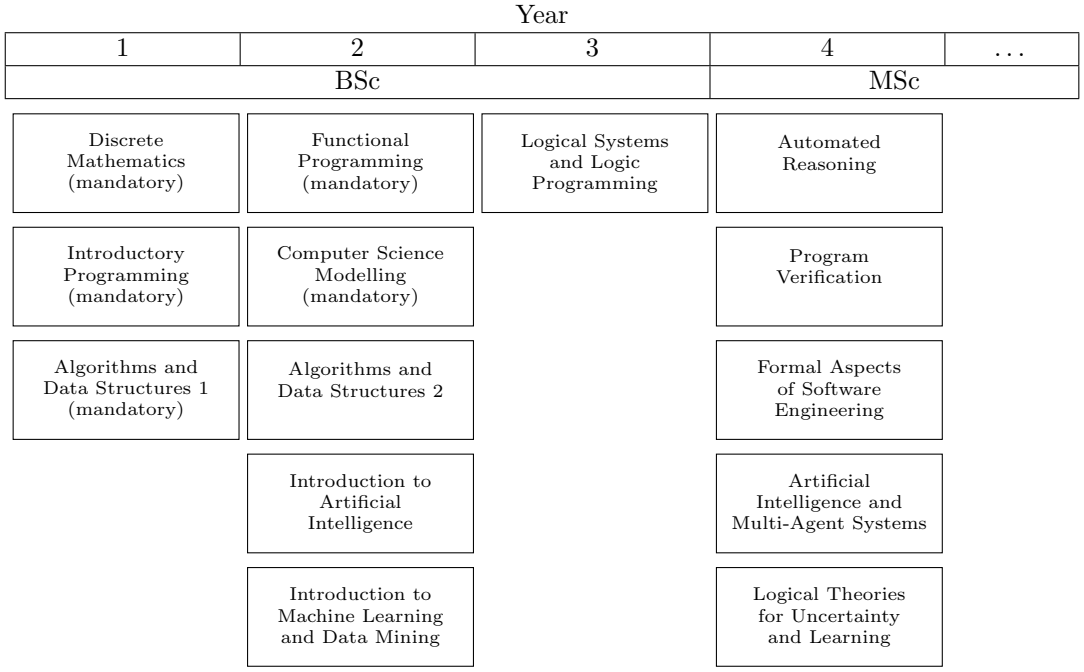

Figure 1: Selected courses surrounding our sequence of logic-oriented courses in the first row.

understand for some students, who may become "stuck" almost immediately. At the scale of our courses we do not have the option of individually guiding students onto the right track for more than a few minutes, and so we are interested in an approach that allows every student to independently move forward with only limited individual guidance.

In the next sections, we describe our experiences using Isabelle in each of the components of our courses.

2 Experiences with tools interfacing with Isabelle

As part of our course material, we have designed several tools that aid the use of formalized logic for beginners [1,2,4]. The first of these, the Natural Deduction Assistant (NaDeA) [10], is a graphical web interface for creating natural deduction proofs. These proofs may be exported to a proof system which is deeply embedded in Isabelle/HOL for verification. Another tool is the Sequent Calculus Verifier (SeCaV) [3], which is a sequent calculus proof system for the same logic, and an "Unshortener" which is a web app that translates a short, readable syntax into proofs in Isabelle while giving instant feedback on any mistakes in the proof. Both the NaDeA interface and the SeCaV Unshortener are external web apps which are not implemented using Isabelle (a few parts are in fact implemented as functions in Isabelle and translated to JavaScript using SMLtoJS). We could imagine an even tighter integration by implementing similar tools directly in Isabelle/jEdit as a sort of "beginner mode", but we have not yet attempted to do so. The tools are available here:

`https://nadea.compute.dtu.dk/` `https://secav.compute.dtu.dk/`

We have developed these tools to make formal proofs easier to get started with, and to provide a gentle introduction to Isabelle. While the Isabelle/jEdit interface is very powerful, it can also be overwhelming for beginners, and so can the logic of Isabelle/HOL, and even Isabelle/Pure. The NaDeA interface is designed such that syntax errors are impossible to input, since the user must always choose from a finite set of options that make sense in the context when inputting formulas or proof rules. The proofs created in this graphical environment can then be exported to Isabelle proofs, which students can study and understand, though they may not be able to write proofs directly in the Isabelle syntax. Our experience is that students initially find this helpful and intuitive, but that they quickly begin to find the constrained interface tedious and slow when they understand the syntax of formulas and the most common proof patterns.

Next, we have designed the SeCaV tool, which allows the students to input formulas and proofs as text. This is much faster than choosing from a set of options for each connective and proof rule, but allows for syntax errors and misapplication of proof rules. Our tool warns students if their proof has syntax or logical errors, but this is still a step down in intuitiveness since these types of errors are not possible at all in the NaDeA tool. On the other hand, the proofs in this tool look much more like the "real" Isabelle proofs that they can be exported to. In fact, the translation consists essentially only of adding some Isabelle syntax such as cartouches and proof methods, which is why we call the web app the "Unshortener". Our experience is that students can, after having used this tool to prove some formulas, begin to write Isabelle proofs directly, though only by following the very strict structure imposed by the tool.

Our goal with these tools is to prepare students to write "real" Isabelle proofs [2]. After having used the tools, they are used to proofs that look essentially like proofs in Isabelle/Pure, and in the next section, we will explain how we introduce this and progress to Isabelle/HOL.

3 Building up to the full Isabelle experience

In contrast to the tools mentioned above, where we explored the pedagogical benefits of using a deep embedding approach to logical modelling, here we employ a logical framework modelling. We have developed a series of Isabelle/Pure theories with a natural deduction formalization of intuitionistic and classical logic [11]:

- Pure_I: Intuitionistic propositional logic
- Pure_I_Quantifiers: Intuitionistic higher-order logic
- Pure_True: Classical higher-order logic

These developments are based on the Pure/Examples by Makarius in the Isabelle sources. The final theory includes a concise proof of Cantor's theorem. The formalizations give rise to simple and natural teaching examples. We have a large set of exercises and in general the solutions work in Isabelle/HOL too — a feature that is appreciated by the students and aids their understanding of structured proofs in Isabelle/Isar [13]. By moving through these increasingly complicated systems, students are able to train proof techniques while having to learn only a few concepts in each step.

After progressing through this series of systems, students have learned all of the essential proof rule of Isabelle/HOL, and can thus write and prove purely logical formulas. They have not, however, seen functions, inductive definitions, proof automation, or any of the more advanced features of Isabelle/HOL. The final component of our MSc course introduces these concepts.

4 Learning to program and prove in Isabelle/HOL

One of the learning goals of our MSc course is that students should be able to actually use an interactive theorem prover to prove theorems. We thus also need our students to get acquainted with "the real" Isabelle/HOL, including concepts such as functions and inductive definitions, and not just the logic. For this purpose, we primarily use the tutorial "Programming and Proving in Isabelle/HOL" [7]. During the course, students are asked to read the tutorial and solve every exercise, either as part of weekly exercise sessions, or as part of assignments. The students progress through the tutorial throughout the semester such that Isabelle concepts and keywords are introduced as needed for the rest of the course.

Our experience is that students find it quite difficult to solve the exercises, even when getting help and hints from teaching assistants. Many of our students do not have functional programming experience (though this is an explicit prerequisite for our course, our institution does not have strict prerequisites), which could explain some of the difficulties. We have considered whether a brief introduction to functional programming using Isabelle could help bring all students up to speed, but have not yet attempted to implement this in our course. Even then, it is our experience that even the more capable students sometimes have a difficult time understanding and remembering the commands and proof methods introduced in the tutorial. Students will thus often either search for examples online or attempt to look up commands and proof methods in the Isar reference manual [13], which often leaves them confused due to the higher technical level of the explanations. One possible solution could be to create a "beginner" version of the reference manual which contains only concepts introduced in the tutorial.

It is also our experience that the exercises in the tutorial are of widely varying difficulty. Some exercises, such as the final exercise, are so difficult that only a handful of students out of a large cohort make significant progress at all, and only one or two can complete it, even when given hints by teaching assistants. One possible solution could be to add more "easy" exercises, but it is difficult to come up with exercises that are relatively easy without being trivial. Another possibility is to break up the more difficult exercises into a series of exercises that guide the reader onto the right path, e.g. by suggesting intermediate lemmas. We believe that this is a worthwhile endeavour, especially for those exercises which we use as parts of assignments, but note that creating a robust guided series of exercises is difficult, since bad choices made by a student in one of the early exercises can often make the later exercises much more difficult.

While students thus find following the tutorial difficult, our experience is that most students do eventually solve many of the exercises. Additionally, most students are able to use Isabelle at least somewhat proficiently at the end of our course. While this is not enough for all students to go out into the world and begin solving real problems using Isabelle, it is enough for interested students to pursue e.g. thesis projects about formalizations in Isabelle.

5 Conclusions and future work

We have explained our approach for teaching students formalized logic and how to use Isabelle. We have detailed some of the issues we have experienced that students have while attempting to learn how to use Isabelle, and how our tools were designed to alleviate these. We have also explained some of the issues we encountered when implementing our tools, and some possible future directions in which we think they could be improved.

We are currently grading the students based on a two hour exam and six assignments handed in during the course.

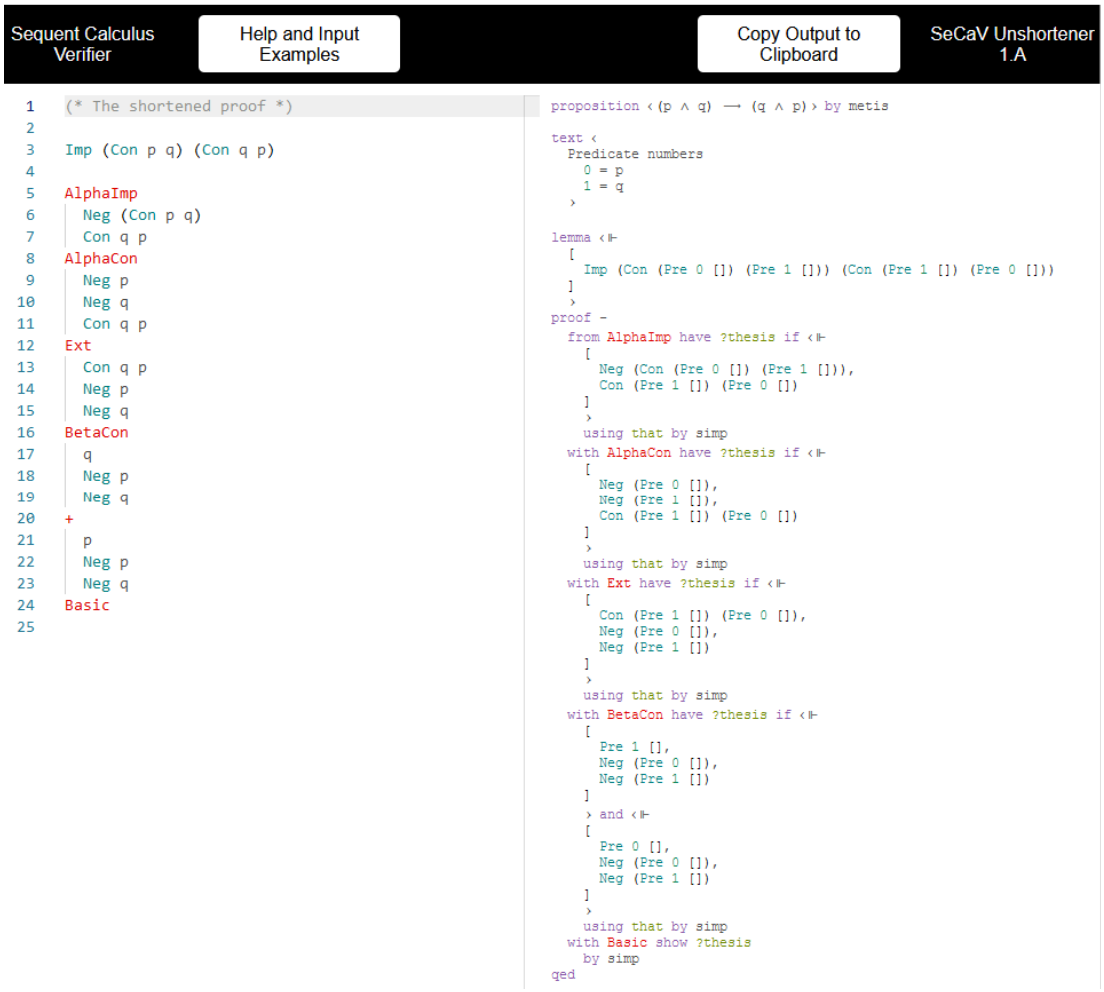

Figure 2: Test version of SeCaV with the unshortened proof to the left and the Isabelle/HOL output to the right.

Our tools are still under development, and we continuously evaluate whether they should be changed or supplemented with new tools. As an example, fig. 2 shows a test version of the SeCaV Unshortener with line numbers and syntax highlighting. Numbers are used for predicates and functions but we also have a version which uses strings [4].

We believe there are still many possibilities for improving the user experience of Isabelle for beginners, and we hope that others will also share their concerns, approaches and tools to stimulate fruitful discussion and ventures in this direction. We believe there are valuable discussions to be had not only about the design of concrete systems, but also about the possibilities for interplay and continuity between different systems and Isabelle itself, including coherence between the interfaces of tools and pen-and-paper proofs.

References

[1] Asta Halkjær From, Alexander Birch Jensen, Anders Schlichtkrull, and Jørgen Villadsen. Teaching a formalized logical calculus. In Pedro Quaresma, Walther Neuper, and João Marcos, editors, *Proceedings 8th International Workshop on Theorem Proving Components for Educational Software, ThEdu@CADE 2019, Natal, Brazil, 25th August 2019*, volume 313 of *EPTCS*, pages 73–92, 2019.

[2] Asta Halkjær From, Jørgen Villadsen, and Patrick Blackburn. Isabelle/HOL as a meta-language for teaching logic. In Pedro Quaresma, Walther Neuper, and João Marcos, editors, *Proceedings 9th International Workshop on Theorem Proving Components for Educational Software, ThEdu@IJCAR 2020, Paris, France, 29th June 2020*, volume 328 of *Electronic Proceedings in Theoretical Computer Science*, pages 18–34. Open Publishing Association, 2020.

[3] Asta Halkjær From, Frederik Krogsdal Jacobsen, and Jørgen Villadsen. SeCaV: A sequent calculus verifier in Isabelle/HOL. In *16th International Workshop on Logical and Semantic Frameworks with Applications (LSFA 2021)*, volume 357 of *Electronic Proceedings in Theoretical Computer Science*, pages 38–55. Open Publishing Association, 2021.

[4] Asta Halkjær From and Jørgen Villadsen. A concise sequent calculus for teaching first-order logic. Isabelle Workshop 2020 (informal, no proceedings).

[5] Asta Halkjær From and Jørgen Villadsen. Teaching automated reasoning and formally verified functional programming in Agda and Isabelle/HOL. In *10th International Workshop on Trends in Functional Programming in Education (TFPIE 2021) — Presentation Only / Online Papers*, pages 1–20, 2021.

[6] Tobias Nipkow. Teaching semantics with a proof assistant: No more LSD trip proofs. In Viktor Kuncak and Andrey Rybalchenko, editors, *Verification, Model Checking, and Abstract Interpretation - 13th International Conference, VMCAI 2012, Philadelphia, PA, USA, January 22-24, 2012. Proceedings*, volume 7148 of *Lecture Notes in Computer Science*, pages 24–38. Springer, 2012.

[7] Tobias Nipkow. *Programming and Proving in Isabelle/HOL (Tutorial)*, 2021. https://isabelle.in.tum.de/doc/prog-prove.pdf.

[8] Tobias Nipkow. Teaching algorithms and data structures with a proof assistant (invited talk). In Catalin Hritcu and Andrei Popescu, editors, *CPP '21: 10th ACM SIGPLAN International Conference on Certified Programs and Proofs, Virtual Event, Denmark, January 17-19, 2021*, pages 1–3. ACM, 2021.

[9] Benjamin C. Pierce. Lambda, the ultimate TA: using a proof assistant to teach programming language foundations. In *ICFP*, pages 121–122. ACM, 2009.

[10] 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, THedu@FLoC 2018, Oxford, United Kingdom, 18 July 2018*, volume 290 of *Electronic Proceedings in Theoretical Computer Science*, pages 14–29. Open Publishing Association, 2018.

[11] Jørgen Villadsen, Asta Halkjær From, and Patrick Blackburn. Teaching intuitionistic and classical propositional logic using Isabelle. In João Marcos, Walther Neuper, and Pedro Quaresma, editors, Proceedings 10th International Workshop on *Theorem Proving Components for Educational Software,* (Remote) Carnegie Mellon University, Pittsburgh, PA, United States, 11 July 2021, volume 354 of *Electronic Proceedings in Theoretical Computer Science*, pages 71–85. Open Publishing Association, 2022.

[12] Jørgen Villadsen and Frederik Krogsdal Jacobsen. Using Isabelle in Two Courses on Logic and Automated Reasoning. In João F. Ferreira, Alexandra Mendes, and Claudio Menghi, editors, *Formal Methods Teaching*, pages 117–132, Cham, 2021. Springer International Publishing.

[13] Makarius Wenzel. *The Isabelle/Isar Reference Manual*, 2021. https://isabelle.in.tum.de/doc/isar-ref.pdf.