Teaching Logic for Computer Science Students: Proof Assistants and Related Tools

Jacobsen, Frederik Krogsdal; Villadsen, Jørgen

Published in:
Proceedings of LogTeach-22

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Jacobsen, F. K., & Villadsen, J. (2022). Teaching Logic for Computer Science Students: Proof Assistants and Related Tools. In *Proceedings of LogTeach-22*

Teaching Logic for Computer Science Students: Proof Assistants and Related Tools

Frederik Krogsdal Jacobsen and Jørgen Villadsen

Technical University of Denmark, Kongens Lyngby, Denmark

Abstract. In the last decade we have focused our main logic courses on proof assistants and related tools. We find that the modern computer science curriculum requires a focus on applications instead of just pen-and-paper proofs. Notably, we teach the metatheory of logic using tools with formalizations in proof assistants like Isabelle such that we have both implementations and theorems about them.

Keywords: Logic · Automated Reasoning · Isabelle Proof Assistant

1 Introduction

While the "traditional" logic course might no longer have a place in an undergraduate computer science program, we believe that logic is still very useful as a tool in many computer science applications. In fact, we believe that logic has never been more relevant in the current computer science landscape, where tools such as type systems, formal verification tools, and proof assistants are becoming more and more popular. Consequently, we no longer focus on pen-and-paper proofs, but on implementations and applications of logical systems, while still retaining foundational material on metatheory.

In the last decade, we have thus focused the main logic courses at the Technical University of Denmark on proof assistants and related tools. This focus was initially also part of a larger strategic project within our department [2]. In particular, we use the Isabelle proof assistant [4] to formalize a number of logical systems and their metatheory [3, 5]. This allows us to focus on applications while still ensuring that students are exposed to concepts such as soundness and completeness, and the formalizations additionally serve as examples of how to prove these properties for concrete systems. In our undergraduate course, we also teach logic programming in Prolog, which has recently enjoyed a resurgence of interest in both academia and industry, especially with regards to the Datalog subset (see e.g. the Datalog 2.0 workshop series [1]).

The very basics of logic is taught in a discrete mathematics course at the beginning of our undergraduate program, but the main logic course comes only towards the end of the program. This means that students already have a well-developed mathematical maturity when starting the course, and that we can thus quickly go into relatively advanced topics. It also means that students have already seen many of the potential applications of logic in their other courses.

This allows us to draw from a wide range of example applications. Finally, placing the logic course towards the end of the undergraduate program means that logic is fresh in students' memory when they encounter more advanced applications in their graduate studies (which, following Danish custom, a large majority of our students pursue immediately after their undergraduate studies).

2 Our undergraduate course on logic

Our main reason for teaching logic is to give students the tools to become software *engineers* instead of "mere" programmers. As mentioned above, our main undergraduate course on logic is split in two conceptual parts: one about logical systems and proofs, and one about logic programming in Prolog. Both of these parts are meant to give students the knowledge to understand the logical underpinnings of a number of topics in computer science. A working knowledge of logical systems is, in our opinion, essential for understanding type systems, model checking, traditional artificial intelligence, and tools such as proof assistants. Logic programming can give a logical understanding of concepts in databases, program analysis, and traditional artificial intelligence.

The two parts are of course closely related, and we teach them in parallel and show several implementations of logical systems in Prolog during the course. Even in the part about logical systems and proofs, we use a very programming-oriented approach, implementing most of the concepts we teach in the Isabelle proof assistant instead of using just pen-and-paper proofs. This includes simple implementations of sequent calculus, natural deduction, axiomatic systems and resolution, and various proofs of the correctness of the systems. This allows students to get a feel for how logical tools are actually implemented, while also exposing them to a bit of formal proof in a proof assistant. Proof assistants are becoming more and more popular, and so we want students to at least get a glimpse of what they can do.

Our undergraduate course thus prepares students to understand the logical background necessary in advanced computer science courses, while also giving them concrete tools and implementations that can be used to implement and prove theorems about these topics.

3 Topics for which logic is useful

As mentioned above, there are many topics in computer science which have logic as a prerequisite, either directly or indirectly. Applications within these topics typically use logic as a tool, so students need to be familiar with how to implement logical systems in practice.

At our university, a number of graduate courses assume some familiarity with logical systems and formal proofs. The most obvious is our graduate course on automated reasoning, which is intended as a direct continuation of our undergraduate course, going even deeper into the mechanics of formal proofs. Most courses relating to formal methods rely on logic for specification and logical systems for verification, making logic a prerequisite for topics such as model checking, security protocols, program verification, and real-time systems. Many courses on artificial intelligence also rely on logic, including such topics as multi-agent systems and logical theories for uncertainty and learning. Courses on software engineering often use logical systems to specify requirements and the interaction and integration of systems. Courses on programming languages use logical systems in the form of type systems, which are indispensable for courses on compilers and advanced functional programming. A working familiarity with formal proofs is also very useful for algorithms courses, where complicated proofs by induction are often needed and can be easier to keep track of if formalized. Finally, many students are interested in undertaking thesis work on various topics which could benefit from logic, and it is thus useful for them to already have some knowledge of the possible applications.

4 Conclusion

We believe that logic should still be part of the undergraduate computer science program, but that the focus should be on logic as a tool. We note that the development in industry seems to trend towards more widespread use of formal methods and features inspired by logic, with e.g. model checking becoming more widely used for distributed systems, Datalog having a resurgence in popularity in program analysis and databases, and the type system of the Rust programming language being inspired by substructural logic. We have thus focused on implementations, applications, and machine-checked proofs, since we believe that this is the best way to prepare students to understand and apply both the tools of today and those of the future.

References

1. Alviano, M., Pieris, A.: Datalog 2.0 (2022), `https://sites.google.com/unical.it/datalog20-2022`, workshop website
2. DTU Compute: Proof Assistants and Related Tools - The PART Projects (2015–2017), `https://part.compute.dtu.dk/`, project website.
3. From, A.H., Villadsen, J., Blackburn, P.: Isabelle/HOL as a meta-language for teaching logic. In: Quaresma, P., Neuper, W., Marcos, J. (eds.) Proceedings 9th International Workshop on Theorem Proving Components for Educational Software, ThEdu@IJCAR 2020, Paris, France, 29th June 2020. Electronic Proceedings in Theoretical Computer Science, vol. 328, pp. 18–34. Open Publishing Association (2020). https://doi.org/10.4204/EPTCS.328.2
4. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic, Lecture Notes in Computer Science, vol. 2283. Springer (2002)
5. Villadsen, J., Jacobsen, F.K.: Using Isabelle in two courses on logic and automated reasoning. In: Ferreira, J.F., Mendes, A., Menghi, C. (eds.) Formal Methods Teaching. Lecture Notes in Computer Science, vol. 13122, pp. 117–132. Springer (2021)