12-1-2002

Providing Assistance for Proofs in the Teaching of Theory of Computation

Padmanabhan Krishnan
Bond University, Padmanabhan_Krishnan@bond.edu.au

Providing Assistance for Proofs in the Teaching of Theory of Computation

Padmanabhan Krishnan
School of Information Technology
Bond University
Gold Coast, Queensland 4229, Australia
E-mail: pkrishna@bond.edu.au

Abstract

In this article we present a technique which helps students in understanding proofs in the context of automata theory. The main conclusion is that student understanding can be improved by using a collection of lemmas and trying to automate the proof in a mechanical theorem prover.

1. Introduction

It is fair to say that a typical undergraduate student in Computer Science finds the classical course on automata theory very hard. This has the effect that more practical courses that depend on automata theory are avoided. For instance, students who have found the course on automata theory hard do not study applications of model checking as it depends on automata theory. However, the automata theory used in such a course on model checking is not very deep. Furthermore, in the advanced course the students are not expected to construct proofs by hand. Rather they have to use a model checker as a tool to prove the required properties.

One of the key reasons for this is the difficulty in understanding and writing proofs. Permitting students to collaborate does not necessarily overcome this problem. Common mistakes made by a majority students are only amplified. Providing a fixed number of hints is also not always appropriate. The students have to be encouraged to construct their own proof (and not be constrained to any fixed proof).

Hence a rigorous tool which can find a variety of errors is required. The tool must be also able to accept a variety of arguments as input. Existing tools such as those described in [3, 6] focus more on “mechanical concepts” such as determinisation and minimisation of finite automata, derivation of strings using context free grammars etc. However, they do not assist in the more “theoretical concepts” such as pumping lemma for regular languages or proving elementary properties about languages [4, 2].

In this paper we argue for the benefits of using a mechanical theorem prover in the teaching of automata theory. However, as theorem provers are not easy to use, we do not expect students to use them. In fact, students are not aware about the existence of the tool. The role of the instructor is to act as the interpreter between the student and the tool. We suggest that our approach helps the student to understand proofs with the assistance of a tool and an instructor.

Owing to the generality of theorem provers, they can be used in a variety of ways. Apart from encoding the various student arguments, the instructor can also provide some of the lemmas needed to automate the proof. The student can then choose from these lemmas to attempt the proof. If a wrong lemma is chosen, the prover is unlikely to make progress. One can then undo the various commands, and proceed by choosing another lemma. The details necessary to complete a proof provide the student with the necessary practice. Once the proof process is complete, the axioms used in the process can be examined. This can also be used by the students to learn about the proof process and how to structure proofs. This is obtained by the lecturer examining the proof trees (similar to those shown in Figure 1). For instance, the sub-tree shown in the figure indicates selection of the counter-example agenAnBn(n!1) and then using two lemmas, viz, agenEQ followed by EQABAX. The other steps are just theorem prover commands used to complete the proof.

There has been some research on better ways to structure proofs [5] and ways to generate student interest in proofs [1]. However, the first report is more concerned about structuring large proofs while the second report is on a better way to introduce logic and other aspects to an uninterested student. Here the aim is to be assist the student in getting a better understanding of the proof process.

The author has been involved in this course for the past nine years. Initially, collaboration on assignments was permitted. However this did not translate to better understanding. But after the process was adopted, there has been a noticeable improvement in the understanding of the topics

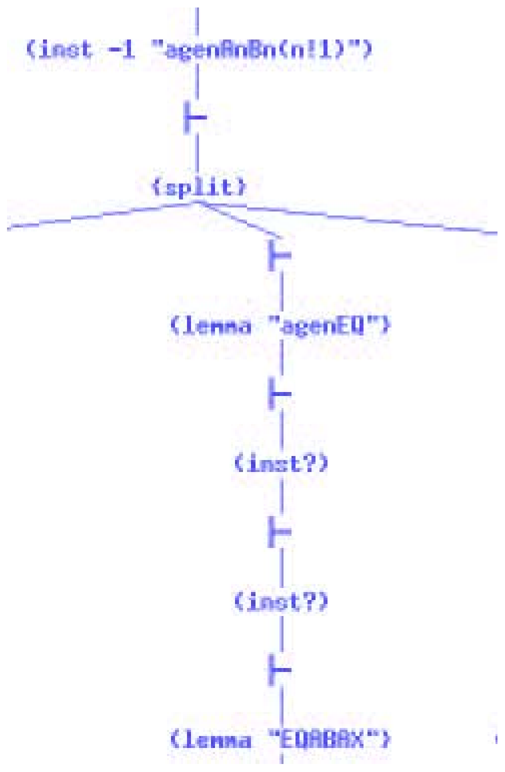

Figure 1. Proof Tree

(in terms of the questions asked in class, attempts at solving homework problems etc.) The author's teaching evaluation also increased.

2. Costs

In this article we have presented the details of a technique to aid students to overcome bottlenecks in the process of constructing proofs. We now comment on the cost to the teacher. The first cost of adopting this approach is the need to code up all the simple definitions in the theorem prover. We use the theorem prover PVS [7] to code the relevant models and prove the required theorems. Completing the basic definitions for the various examples used in a typical course took the author about two days. But the author was already familiar with PVS and its various features. One of the key issues will be to identify a set of definitions that can be used for a variety of issues thereby reducing the coding effort. Another significant cost is the need to carry out the proofs and encode the lemmas that will aid learning. These lemmas are used to provide hints when a student is stuck. Here it is harder to estimate the time spent as the lemmas used were identified over 2-3 years. These were based on the questions asked by the students and the mistakes made in their home-works. However, we estimate that it is not longer than 4 weeks of work.

A more time consuming aspect is the translation of the student's reasoning into PVS input. Trying to do this online is not satisfactory. The main problem is the time required to incorporate the current state of the proof tree into the reasoning presented by the student. Most of the time was actually spent in examining the proof trees and generating the feedback which characterises potential problems. In some cases, the only valid answer appears to be 'you are way off target, try something else'.

3. Conclusion and Future Work

In principle, a theorem prover can be used to provide assistance for a student to understand the proof process. Given the current state of the art in theorem provers, this cannot be done without instructor intervention. However, by not restricting the tool to provide fixed hints, the learning can be tailored to particular students. The generality of theorem provers can be used to cater to variety of students; but this generality is also a limitation. The next phase of the research is to build a tool that translates between the formal language of PVS and the informal but precise language that students should use.

References

[1] D. Gries and F. B. Schneider. A New Discrete-Math Course. Technical report, Cornell University, April 1994.

[2] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, 2001.

[3] T. Hung and S. H. Rodger. Increasing visualization and interaction in an automata theory course. In *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education*, pages 6–10, 2000.

[4] D. Kozen. *Automata and Computability*. Springer, 1997.

[5] L. Lamport. How to Write a Proof. Technical report, DEC SRC Report, 1993.

[6] M. B. Robinson, J. A. Hamshar, J. E. Novillo, and A. T. Duchowski. A java based tool for reasoning about models of computation through simulating finite automata and turing machines. In *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education*, pages 105–109, 1999.

[7] J. Rushby. Specification, proof checking, and model checking for protocols and distributed systems with PVS. In *Tutorial presented at FORTE X/PSTV XVII '97: Formal Description Techniques and Protocol Specification, Testing and Verification*, Nov. 1997.