Experiences and New Alternatives for Teaching Formal Verification of Java Programs*

Ana Romero
University of La Rioja
Spain
ana.romero@unirioja.es

Jose Divasón
University of La Rioja
Spain
jose.divason@unirioja.es

ABSTRACT

Formal verification of algorithms is traditionally taught in Computer Science studies in a *theoretical* way by means of the Hoare logic axioms and doing (by hand) exercises of verification of small programs. This work shows our experience with Krakatoa, an automatic theorem prover which allows students to interactively visualize the steps required to prove the correctness of a program.

CCS CONCEPTS

• **Theory of computation → Automated reasoning**; *Proof theory*; • **Software and its engineering → Formal software verification**;

KEYWORDS

Formal verification of algorithms, Hoare logic, Java, Krakatoa.

ACM Reference Format:
Ana Romero and Jose Divasón. 2018. Experiences and New Alternatives for Teaching Formal Verification of Java Programs. In *Proceedings of 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE'18).* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3197091.3205811

1 INTRODUCTION

In the University of La Rioja, formal verification is part of a mandatory course and was traditionally taught only in a *theoretical* way, explaining the Hoare logic axioms [1] and presenting the inference rules that make it possible to prove that a program satisfies a specification (given by means of a precondition Q and a postcondition R). The proofs of correctness considered in such a course are *restricted* to programs corresponding to the following sketch:

```
<init> while B do {<body>} <end> return <var>
```

where the blocks <init>, <body> and <end> consist of a sequence of elementary instructions, usually assignments and conditional structures. This is not a restriction: if there are several *sibling* loops it can be thought that all but the last one are inside <init>, and if there are nested loops one can think that the internal loops are inside <body>. Following Hoare axioms, five steps are necessary to verify the correctness of a program with the previous sketch: 1) find and invariant P for the loop, 2) verify the specification $\{Q\}$<init>$\{P\}$, 3) verify that P is an invariant, i.e., the specification $\{P$ and $B\}$ <body> $\{P\}$ is satisfied, 4) verify the specification $\{P$ and not$(B)\}$<end>$\{R\}$ and 5) ensure that the loop always stops. The difficult part of the exercises is the determination of the invariant P. It is very frequent that students propose invariants that are not strong enough and they usually carry out different attempts (repeating steps 2-5 for all of them) to find the correct one. This *traditional* way of teaching formal verification was the chosen one at our University until 2013. At that moment, we decided to complement this teaching by means of some support tool to formally verify Java programs in a semi-automatic way. We did a study of the different approaches (interactive theorem provers, model checking, automatic tools based on Hoare logic) and the available software for this task (Isabelle, Pathfinder, KeY, OpenJML, etc). The chosen support tool was Krakatoa, since it is easy to use for beginners, powerful enough for our examples, provides understandable feedback for students and closely follows Hoare logic, i.e., the steps presented in the theoretical lessons.

In 2013 we only used Krakatoa as support tool during theoretical lessons, showing to students some basic examples. Since 2014 we decided to include in the course some practical lectures in a computer classroom where students could use the tool themselves to prove the correctness of some Java programs, such as checking if an array is sorted, exponentiation, linear search in an array and computation of square roots. These years, we have observed that after using Krakatoa the students understand the different steps of the (theoretical) formal proofs in a better way; more concretely, when Krakatoa was not used as a support tool many of the students *memorized* the exercises of formal verification (and very frequently they did not really understand them). This better understanding of students has been shown in the marks of the formal verification exercises in the final exam that have increased significantly (see Table 1, exercises are marked with a number between 0 and 10, the higher the better). The first year of use of Krakatoa just as a support tool (2013), the average of the marks in the final exam of the formal verification part was higher than previous years. The difficulty was very similar. During the following courses the marks remained higher than in 2012, although deeper contents and higher difficulty of the exercises were demanded in the exams.

Table 1: Results

Year	Marks
2012	6.14
2013	7.50
2014	7.06
2015	7.17
2016	6.81
2017	7.23

REFERENCES

[1] Charles Antony Richard Hoare. 1969. An axiomatic basis for computer programming. *Commun. ACM* (1969), 576–580.

*Work partially funded by the Vicerrectorado de Profesorado of the University of La Rioja, Spain, by means of an educational innovation project.

ITiCSE'18, July 2–4, 2018, Larnaca, Cyprus

ACM ISBN 978-1-4503-5707-4/18/07…$15.00
https://doi.org/10.1145/3197091.3205811