Proposal for a tutorial on

# "Teaching Formal Methods to Novices"

Satellite event to FM'08

**Topic** Teaching formal methods to students in high school and introductory university courses

**Organizer** Ralph-Johan Back, Abo Akademi University, Turku, Finland

**Objectives** Formal methods are a standard ingredient in CS and SE university curriculum. However, the emphasis on using mathematical methods in building software systems is not always received very well by students, as the methods taught are often seen as overly complex and difficult to apply in practice. We have been working on this problem in our research lab, Learning and Reasoning (Abo Akademi and University of Turku) for a number of years now, and have developed new methods for teaching formal methods to novices that seem to work very well in practice. Novices are here students in high school and first year university students who encounter formal methods for the first time.

The course will focus on two main topics:

- how to teach the systematic construction of mathematical proofs and derivations in high school and introductory university courses, and
- how to teach the construction of correct programs for first year CS and SE students in university.

On the first topic, we describe how to use *structured derivations* [4, 5] for presenting mathematical proofs and derivations. Structured derivations are a further development of Dijkstra's *calculational proof style*. The main extension is the use of nested derivations, which makes it possible to express arbitrary logical derivations in a rather natural fashion. We have adapted the structured derivation proof style to the mathematics curriculum in Finnish high schools [6], and carried out extensive case studies on the use of structured derivations in mathematics teaching. The experiences have been very good, showing that the use of structured derivations improves the performance of students in mathematics quite markedly. Structured derivations also provide a good formalism for computer supported and web based mathematics teaching, because the syntax of mathematical proofs is more explicit and therefore easier to construct, read and analyze.

The second topic is concerned with how to build programs that are correct by construction. The traditional approach is to either use a posteriori verification, where we first construct the program, identify the necessary pre- and postconditions and loop invariants, and then verify that the program is correct. Alternatively, we can use Dijkstra's constructive approach, and build the program code and the proof hand in hand. In *invariant based programming* [2, 1], we take Dijkstra's approach to the extreme, by constructing pre- and postcondition as well as possible loop invariants before any code is written at all. The program is then verified in the usual way by proving the verification conditions (using structured derivations). Invariant based programming is supported by a new visual formalism, *nested invariant diagrams*. This approach, when supported by an appropriate work flow, seems to work quite well in practice, and allows the programmer to quickly identify errors in his program while he is constructing it, and in the end deliver a program that has been proved mathematically correct. We have taught invariant based programming to first year CS students with very good results [3]. The students have learned how to construct correct programs in practice, and have expressed quite a strong appreciation of the method. We have also taught the approach to experienced programmers (some with prior knowledge of formal methods, some without), with similar good results.

**Content and duration** The course will be given in 6 hours. Both structured derivations and invariant based programming will be treated in the same way: we start by first introducing the general general approach, then we describe tool support in connection with a few case studies, and finally we discuss the experiences that we have had in using these approaches in high school and in university. The approximate allocation of time is as follows:

- An overview of structured derivations (1 h)
- Tool support for structured derivations, illustrated with a few case studies (1 h)
- Experiences from using structured derivations in high school and university, discussion (1 h)
- An overview of invariant based programming (1 h)
- Tool support for invariant based programming, illustrated with a case study (1 h)
- Experiences from using invariant based programming in first year university courses, discussion (1 h)

**Prerequisites** There are no real prerequisite for this course, apart from a general understanding of the basic principles of logic and program verification, which we expect everybody attending FM'08 to have.

**Past experience of instructors** Ralph-Johan Back is a well known researcher in formal methods, and has extensive experience in Ph.D. education and in teaching graduate courses. He has been invited speaker to many conferences, workshops and summer schools. He has also organized himself a number of such events. He is the inventor of the refinement calculus, and co-inventor (together with Reino Kurki-Suonio) of the action systems formalism, and has published a large number of scientific articles on these topics, as well as on other topics (parallel computing, software engineering, programming methods, etc.). He is professor of Computer Science at Abo Akademi University.

# References


[1] Ralph Back, Johannes Eriksson, and Magnus Myreen. Verifying invariant based programs in the socos environment. In *Teaching Formal Methods: Practice and Experience, BCS eWiC Proceedings 2006.*, London, December 2006.

[2] Ralph-Johan Back. Invariant based programming: Basic approach and teaching experience. *Formal Aspects of Computing*, To appear.

[3] Ralph-Johan Back, Johannes Eriksson, and Linda Mannila. Teaching the construction of correct programs using invariant based programming. In *SEEFM07: South-East European Worshop on Formal Methods, Thessaloniki*, 2007.

[4] Ralph-Johan Back, Jim Grundy, and Joakim von Wright. Structured calculation proof. *Formal Aspects of Computing*, 9:469–483, 1997.

[5] Ralph-Johan Back and Joakim von Wright. *Refinement Calculus: A Systematic Introduction.* Springer-Verlag, 1998. Graduate Texts in Computer Science.

[6] Ralph-Johan Back and Joakim von Wright. *Mathematics with a Little Bit of Logic: Structured Derivations in High-School Mathematics.* Manuscript, 2006.