Formal Methods are Fun

Notes by J.H.Davenport

2–3 December 2019

Contents

Chapter 1

2 December

1.1 Mismatch between programmer intuition and program semantics: an education related vulnerability: Magne Haveraaen

Claims this is related to parameterised unit testing. There was an interview with radioogists on Norwegian broadcaster TV2: system slow, sometimes wrong patient, notes wrongly recorded. These can be reproduced as simple programming mistakes, or incorrect use of collection classes. In Java, we can have `equals` and `hashCode` methods out of sync.

In Java, `equals` is meant to be an equivalence relation (except on `null`, since nothing is equal to `null`). In general, the specifications are essential for use with collection classes, but often skipped in programming education. The Java documentation uses textual specifications. The axioms are often clumsily written, and there are many intricacies around the corner cases and exceptions. The axioms could be formulated as Java code, where method parameters declare the free variables. We can have aunit testingframework: equational logic, conditional equation logic and Boolea expression logic.

Java claims that two `equal` ojects should have the same `hashCode`. Java comes with default mplementions via machine addresses, becaus ethese are in practice unusable. Conforming methods can be auto-generated by the IDE, but must be re-generated when the class is upgraded. Probelsm interacting with manual tweaking.

Java's specification of `compareTo` (total order) is clumsy.

1.1.1 Parameterised UnitTesting

Use axioms as test oracles. See Jaxt or Catsfoot `https://bldl.ii.uib.no`. Quickcheck for Haskell etc.

1.1.2 Afterwards

Apparently it was merely a guess that these problems were related to hash codes, but at least the "wrong patient" ones could sensibly be attributed this way.

Apparently C++ is going the way of code generation

1.2 Fun with FM for Better Eucation: Shilov

Speaker from Innopolis University. Wikipedia claims "This is the smallest town in Russia[8] with 96 inhabitants according to the 2016 estimate". They had a similar workshop but only five participants.

Had a buggy program to estimate π by random sampling in $[0,1] \times [0,1]$.

Uses "Dancing Men" story [Sir05] — a straw poll in the audience showed half of the audience had read it.

Spoke about "esoteric languages" — designed to teach concepts of semantics rather than practical use. Had a bizarre example of steganography with programs.

1.3 Adapting to Differnet TYpes of Target Audiences in Teaching Formal Methods: Cerone

Various student statements "make CS non-understandable", "industry don't use them" etc.

Strategy for teaching:

Motivation enabling learners to build themselves their instrinsic and extrinsic motivations to develop FM

Fun to keep the learners contiuously engagedin orde to retain interest

Practice using tools.

Suggests for university students: start with general context; then ether success sories or current trends, and then problem solving: suggests games/puzzles as appropriate. In his view, tools feed into practice, and formal semantics are a result.

1.4 Inquiry- and Reasearch-based Teaching in a Course of Model Checking

New algorithms are often being implemented in well-known model checkers. These are comoex with a large code volume, so a high entry barrier. Shows a set of typical LOs, and asks how a lecture-based course can do it.

Hence "acquire the theoretical foundations by identifying and analysing common software errors. Align these foundations with the body of knowledge. Design and implement a novel model checker as independently as possible." But the cognitive requirements are higher, progression is less linear.

Shows an example: postits for "hazard collection for elevator", then sorted by x=%age software; y=severity. Then needed to solve high/high. Course was speaker+6–7 students, essentially doing a group project. Used Kanban project management. Timeline of repository commits versus "ideas". Ended up writing [POKG19].

Q Feasible with 30 students?

A Probably not. There we have small individual projects.

Q You were mixing "model checking" with "writing a model checker".

A Yes, we

1.5 White paper discussions

Chair Summary of notes.

One We should move away from Set Theory to List Theory as a foundation.

Holge should it be specialist or generalist?

JHD Not either/or. For example, P/NP & Pspace are specialist, but basic $O(n^2)$ v. $O(n \log n)$ is core.

1.6 Prototyping Games using Formal Methods:

Two kinds of examples: theoretical (e.g. the water boiler), or taken from industry (e.g. several thousand lines of B, and half-way through "here is where the train moves"). "Games to the rescue". Usually well-known to the students. Saves time on describing the model. Chess, for example,is quite complicated. We use B method (classical and Event-b), with ProB and ProB 2.0, with Rodin and BMotion Web.

Pac-Man

Chess : Piece-centred or figure-centric model? Gives rise to great debates.

Lightbox 3D, levels of game, reading in configurations

Problems with usability of tools especially Rodin.

Q What parts of B were useful?

A Really tool chain.

1.7 Schlingloff

Games are very old. Improve planning skills etc. This was a graduate course on verification. Target is skills in the use of verification tools (not games themselves, not verification tool construction). `https://osf.io/jxra3`.

Explain state transition systems.

Sudoku SAT solving via $P(I, j, k)$. 729 propositions, 9000 literals. Then wolf-goat-cabbage. Actually need the path, so ask for the counter-example to the negation. Sliding block puzzles are best suited to state space explosion. Rubik etc. are open extensions.

Board games Does Tic-tac-toe with three players: Board, Naught and Cross. Written in ISPL - about 40 lines. Need to be careful otherwise you prove Naught wins $\wedge$ Cross wins.

SMT solving can use alphametic problems.

It is important to introduce the problem, then pick the tool. Evaluation = I tried it and the students like it. Claims that scientific evaluation of such courses is infeasible: look at cost of PISA. The goals are long term.

1.8 JHD

Chapter 2

Living Lab

2 and 3 both come from a forthcoming back.

2.1 Holge Schlingloff

Student exercises: SAT, nuSMV `http://nusmv.fkb.eu`; MCMAS `http://vas.docic.ac.uk`.

2.2 Algebraic Specification with CASL

2.3 Formal Verification of Security Protocols

Good building tools forprotocol visualisation.

2.4 PacMan and Chess Model

2.5 FM for Programming Contests

I (Innopolis) would like to introduce FM in ICPC.

Chapter 3

3 December

3.1 They Didn't Know They Were Doing Mathematics: Olveczky

What is a good FM course? I use rewriting logic/Maude. [AmazonCACM2015] "FM have a reputation for takig a great deal of effort to verify the trivial". Perception that only used for safety-critical. Speaker cites 737MAX[1]. But of course we are more dependent onthese: sef-driving cars, airplanes (737-AX), power distribution, ... Cloud computing is very much "winner takes all". Notes that Amazon, Facebook now using FM (basically citing JHD's talk).

But people have worse mathematical backgrounds, and are sceptical to mathematics. Also not a core part of the curriculum, and tends to be stand-alone.

More motivating examples:two-phase commit i the contexts of say, plane tickets.

Example in Maude of Needham–Schroeder and intruders. Does this for second-year students (25 students). But full Mause (for OO odels) has deficienices. Explicit state analysis takes time, and there are challenges for scalability.

But this doesn't actually do software verification, SMT is missing, for example. Student feedback is positive (helps that they good grades!).

Author's book: Designing Reliable Distriuted Systems (Springer).

Q Isn't Needham–Schroeder old, and small?

A Took a long time to find the bug(s), and it's small enough.

Q Maybe too small?

A Possibly, but it deeds on the length of the course and the assessment mix. This works for us.

[1] JHD dubious

3.2 Teaching them early: Casey Denner

Perforamnce on frist-year FM has been historically poor. First year students enjoy the gratification of rpogramming Claim is "if we instil within pupils in school the discipline of computational thinking, ...". Technocamps is a pan-Wales schools outreach programme founded in 2003. Less than 40% of computing teachers have an training in ICT, never mind computer science. Most of these teachers are history/PE/English teachers.

One of the workshops is "computational thinking". Two paradigms at primary schools. Show Man-Fox-Chicken-Corn (etter graphics than Wolf-Goat-Cabbage). Use Scratch for a modelling tool. Teaching abstraction. Water jugs is a good one, because it leads them to a transition system.

On average delvered 98 hours/secondary school in South Wales acroos 2014–16. Since 2011, over 7% o fthe Sesh -224 have participate din a technocamps, 43% girls.

How does this feed into "Modelling Computer Systems" (a.k.a. "Discrete Maths") at first-year Swansea. Since Technocamps started, the %age of failshas gone from 10% to very small. %age firsts generlaly rose, with a dip coinciding with a major growth in student numbers. Students can struggle with explaining "why" two diagrams differ. Example of token games as bisimulations. Illustrates copycat strategies. Examples of group working with diagrams of state transition systems.

Q Do you use the words "bisimulation" and "coinduction"?

A No!

Q Material available?

A Technocamps website.

Q You mentioned "small classes".

A We started with 100, but now have one tutor to 40–60.

3.3 From Stories to Concurrency: How Children Can Play with Formal Methods: Cerone

Note that in many schools Computer Science is taught as a new stand-alone subject, independent of others, or as a servcie subject. Taught as intrinsically connected with computers. But at school subjects should e connected, hence F as a bridge between Maths and CS.

Comments that choice books and parallel stories are one way of introducing parallelism. When exploring parallel stories with my children, I found that children are interested in cmplex stories with many characters. Example of a complex paragraph from a story, and various questions based on it. A different

paragraph tells your friend different things. Should you and your friend collaborate? Then we have questions of synchronisation. His 9-year and 13-year old had (different) proofs of the solution.

Concludes that a completely unplugged approach can expose students to concurrency concept. No jargon is necessary!

3.4 When the Student becomes the Teacher: Farrell & Wu

This is based on a course (Software Verification) we both took as students. Compulsory third-year[2] CS course. Optional for general science students (but not many). Prereq: Java programming and Discrete Structures.

1 week Constract design

3 weeks Natural deduction and Coq

2 weeks Hoare logic

. . .

30% Continuous assessment: two hour lab each week, graded by demonstrators.

70% Best 3 out of 4 exam questions. The worst question was the SMT one.

Issues Identifying loop invariants, presenting Hoare Logic, understanding low-level SAT/SMT. The online versions of Z3 etc. where not very reliable (?). Natural Deduction in Coq was confusing. Hard to see practical examples — many local software jobs, none of which state they use FM.

Plans Live coding SMT-solver to solve Sudoku. Also try with MSc students.

Q What was the "fun"

A "I enjoyed teachers it" — but that's not the answer, I admit.

3.5 White Paper

- Role (and visibility) of Maths keeps coming up.
- Repository of real-world examples.
- Many people spoke about encouraging potential PhD students etc.
- General question about usability of tools.

JHD Do we need to engage in the same sort of thinking that led to WATFOR etc.?

- Should we look at proving APIs correct, e.g. the hashCode/equals (§1.1).

[2] At Maynooth: third year of four.

3.6 Learning experience of two students with different backgrounds: Omirgaliyev

Course: Modelling in Maude, Theorem-proving in Lean. I had Advanced Data Structures and Algorthms, and Modelling & Simulation; the other students had had FOL and programming paradigms course. The course had a very short description i register. Other student chose this by eleimination, and didn't really understand description. I found debugging in Maude difficult. He found Lean easy given his background, but I found it more difficult. He wondered why more mathematicans don't use Lean. But theorem proving is hard: $a+(b+c) = (a+b)+c$

Verification takes time and money. How much do we need?

3.7 Teaching FM in Academia

Claims FM are under-represented in academia, and there isn't a systematc review. Looked at confereneces 2001-2019 in English.

Challenges Student scepticism. It's not used elsewhere, even where it might be useful, such as Networks/OS. Tools (e.g. Maude) hard to install, not in standard repository, poor feedback (prints entire program and says $\exists$ error). No comprehensive textbook. Inadequate background in mathematics.

Strategies Lots of examples. Real-life examples and proofs. Also Gamification, Electronic Voting system, use of Formal Methods in Programming Contests: see 2.5. Also tool simplification, e.g. use a language such as Python which has built-in expressivity.

Limitations Most of these papers are reporting intuition based on informal feedback, or student enrolment growth.

Self My background was similar to the other student from the previous talk. I still have worries about formal methods, but maybe they could be used in compilers. Maude is sold as functional, but in fact uses Prolog-style matching — realising this helped. I only started to appreciate the book after the 11th chapter.

3.8 White Paper continued

Discussants who actively participate will be co-authors with Antonio+me as editors.

Shared (?GoogleDoc) then discussion period.

JHD:not disagreeing that uptake of FM is slow, but also that it's invisible. Most people don't see in an aircraft "a distributed computing system with unusual peripherals".

1. Should FM be visible everywhere in the curriculum?

2. Should there be a compulsory FM course?

3. How large a class? Resources?

4. Should there be a central web resource of examples, tools etc.?

5. Can we agree objective evaluation criteria for teaching methods?

6. Should we have more presentations from students?

7. Would a recommended curriculum help?

JHD: FM are often seen as binary, but we see work on FM/Agile [Cha16], and MZ's new motto. In theory one could separate most of a website from the part that processes credit cards.

Bibliography

[Cha16] R. Chapman. Industrial experience with Agile in high-integrity software development. In M. Parsons and T. Anderson, editors, *Developing Safe Systems: Proceedings of the Twenty-fourth Safety-critical Systems Symposium*, pages 143–154. Safety-Critical Systems Club, 2016.

[POKG19] Jessica Petrasch, Jan-Hendrik Oepen, Sebastian Krings, and Moritz Gericke. Writing a Model Checker in 80 Days: Reusable Libraries and Custom Implementation. *Electronic Communications of the EASST*, 76, 2019. URL: `https://ubsrvweb09.ub.tu-berlin.de/eceasst/article/download/1074/1041`.

[Sir05] Sir Arthur Conan Doyle. The Adventure of the Dancing Men (Chapter III of The Return of Sherlock Holmes). `https://en.wikisource.org/wiki/The_Return_of_Sherlock_Holmes,_1905_edition/Chapter_3`, 1905.