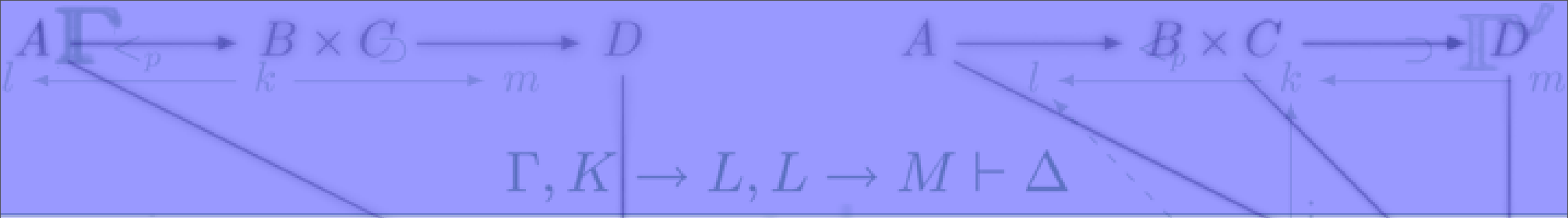

Program Verification in Undergraduate Teaching: A Status Report

Joe Kiniry
University College Dublin

Systems Research Group
School of Computer Science and Informatics
University College Dublin

Tackling Grand Challenges

- we need people, particularly young researchers and practitioners
- how healthy is the pipeline of computer scientists that know about verification?
- what is the state of instruction in the area?
- what are some of the best practices in teaching correctness-centric software?

Training the Next Generation

- get students hooked early
- remind them often of the core concepts
  - reify those concepts with concrete projects
  - help them see more than one approach
  - ensure that they use quality tools
  - measure their behavior from year to year

Systems Research Group
School of Computer Science and Informatics
University College Dublin

The State of Theory Instruction

- "theory" in the U.S. usually means complexity and recursion theory and model checking
- courses in discrete mathematics, logic, program reasoning, etc. are vanishing
- core ideas like assertions and invariants are entirely missing from curriculums
- few tools are used in theory instruction

Challenges in Practical Instruction

- tough choices in PL, style, tool use, projects
- enough with stacks and queues!
- ML and Haskell are beautiful, but you loose the students for the trees
- FM instruction without tools is like learning to program from reading a book
- doing a practical course “right” is an enormous investment

One Way to Integrate Theory and Practice

- make tough-but-logical choices:
  - a “popular” programming language (Java)
  - a de facto standard spec language (JML)
  - usable tools integrated with popular IDEs
    - CheckStyle, PMD, FindBugs, Metrics, Emma
    - JML compiler, JML-jUnit, ESC/Java2, BONc
- reuse existing IDE and course materials

Students Learning JML

- JML for undergraduate instruction
  - from “JML” universities like UCF, Concordia, UCD, KSU, and RUN to those with appreciation for correctness like Caltech, UWash, KSU, CMU, MIT, and many others in North America, the EU, and Australasia
- core concepts covered (JML level 0 and 1)
- tool use is integral to the courses

JML at UCD

- used in 1st year through 3rd year software engineering courses for past 4 years
  - cellular automaton case study (1st years)
  - self-defined projects (3rd years)
  - Thrust game case study (2nd years)
    - first time we are using BONc

Lessons Learned

- coherent concepts across courses
- consistent message from faculty
  - incremental concept and tool introduction
  - language and tool reuse across courses
  - reliable tools are critical
  - align assessment and tool use

More Information

- Mobius Program Verification Environment
  - http://mobius.ucd.ie/
- the JML Tool Suite
  - http://www.jmlspecs.org/
- ESC/Java2, BONc, and other tools
  - http://kind.ucd.ie/products/opensource/