Contents of this Course

Software Engineering using Formal Methods

Contents of this Course

Software Engineering using Formal Methods

Contents of this Course

Software Engineering using Formal Methods

- **Teach specification language like programming language**
  No formal semantics, by example, as problem solving tool
- **Use theorem prover as black box**
  Students analyse substantial systems in lab
- **(High-level) tools and languages not quite mature enough**
  (some people claim they will never be)

Contents of this Course

Software Engineering using Formal Methods

- **Teach specification language like programming language**
  No formal semantics, by example, as problem solving tool
- **Use theorem prover as black box**
  Students analyse substantial systems in lab
- **(High-level) tools and languages not quite mature enough**
  (some people claim they will never be)

Software Engineering using Formal Methods

- **Teach formal methods like formal logic**
  (the “traditional” approach)
- **Tell what is “under the hood”, formal semantics**
- **Emphasize theory, do toy examples in labs**

A Compromise

Goals in designing this course

- **Avoid impression that FM is useless academic pastime**
- **FM can be mastered within 2SWS course**
- **Avoid overwhelming with formal details**
- **Yet enough formality to let participants know what they are doing (necessary to use FM contemporary tools effectively)**
- **Illustrate main approaches in FM today**

Teaching and Research

Teaching FM as inspiration for research and tool development
(students will become users and customers)

- **Design property specification languages as useful, clear, mature as modern programming languages**
- **Teach them as systematically as programming languages**
- **Specification patterns**
- **Design for verification**
- **Model interface design of FM tools after successful software engineering paradigms**

  Symbolic execution $\Rightarrow$ symbolic source code debugger
- **More automation, less formal stuff needed to know**

Are Formal Methods for “The Real World” ?

Formal methods do not scale up	**Verified controllers, telephone switches, even compilers**
No time for verification left	**Formal methods particularly useful in early design stages, find bugs in spec, missing requirements**
Formal methods too difficult	**Well-understood & well-designed properties & code should be verifiable; automated tools for tedious details**
Formal methods too expensive	**Formal methods can save cost when properly applied; “Pentium bug” of the car industry only matter of time with today’s practices**

Got Interested in Formal Methods?

We are looking for students who want to get involved

Selection of topics:

- **Explaining failed proof attempts**
- **Verifying non-state-based properties**
- **Combining verification and testing**

Other topics possible — contact us!