DD2452 Formal Methods

Concluding Lecture

Lecture Outline

1. Course summary
2. Beyond the course
3. Exam preparation
4. Course evaluation

1. Course Summary

- **Formal methods**:
  collection of formal notations and techniques (i.e. based on discrete mathematics and mathematical logic) for modelling and analysis of program behaviour. Basis for *tool* support.
- **Common goal**:
  the design of *correct* systems.

Formal Verification

- **Two possibilities**:
  - correctness by design: transformation
  - establishing correctness: *verification*
- **Three ingredients**:
  - model M M ψ
  - specification S ϕ ϕ
  - verification $\models M \approx S$ $M \models \phi$ $\models \psi \rightarrow \phi$

Approaches Considered in the Course

1. Hoare Logic & Program Verification
2. Temporal Logic & Model Checking

Hoare Logic and Program Verification

Goal:	Correctness of state transform.
Abstr.:	low-level
Models:	Source code (Java)
Specs:	Assertions (Hoare Logic, JML)
Method:	Proof tableaux; VCG + ATP
Tool:	ESC/Java2

Conclusions

+ helps in capturing transform. behaviour
+ *modular*, hence scales well
+ ESC/Java2 helps finding *logical* errors

- automatic at expense of completeness
- requires more detail than just interface
- bad at data structures

Temporal Logic and Model Checking

Goal:	Correctness of state sequences
Abstr.:	medium-level
Models:	Transition systems (Promela)
Specs:	Temporal logic (LTL, CTL)
Method:	Automata-based
Tool:	SPIN

Conclusions

+ realistic modelling of comm. protocols
+ efficient model checking
+ counter-examples as error traces

- finite-state: no unbounded data, recursion or dynamic process creation

2. Beyond the Course

- Infinite-state systems
  - recursion: pushdown automata
  - dynamic process creation: induction
- Theorem Proving
- Program Analysis
  - type systems
  - abstract interpretation

3. Exam Preparation

What do bring:

the book, lecture slides, handouts, own lecture notes taken in class

1. Hoare logic
   - Specifying programs as Hoare triples
   - Verifying programs using proof tableaux
     - Partial & total correctness
     - Concurrent programs (Owicky-Gries)

2. Temporal Logics (LTL, CTL)
   - Understanding the meaning of formulas
     - evaluation on states in models
     - formalizing properties
     - relating formulas in LTL and CTL
     - relating formulas to Bűchi automata
   - Verifying temporal properties
     - automata-based approach

4. Exam Evaluation

- Help improve the course!
- Anonymous evaluation
- How meaningful did you find the course?
- What should be added or removed?
- Other suggestions for improving the course?