



Formal Methods

<http://www.imit.kth.se/courses/2G1516/>

Mads Dam mfd@imit.kth.se 790 4123/070 8 76 44 40
Mika Cohen mikac@imit.kth.se 790 4265

2005 Mads Dam IMIT, KTH

1

2G1516 Formal Methods

Synopsis

- Techniques and tools for formal modelling and analysis of computer systems
- Topics:
 - Modelling techniques
 - Model checking, temporal logic, automata, SPIN
 - Process algebra, CCS, CWB
 - Deductive verification, theorem proving, Floyd/Hoare logic, ESC/Java
- Theory and labs
 - Aim 1: Show this is useful stuff
 - Aim 2: Understand and reason about the theory
- Do not underestimate the workload!

2005 Mads Dam IMIT, KTH

2

2G1516 Formal Methods

Feedback Please!

- Your help is requested...
- Course committee (kursnämnd):
 - NN1: _____
 - NN2: _____
 - NN3: _____
- Very important!

2005 Mads Dam IMIT, KTH

3

2G1516 Formal Methods

Requirements, 1

- Three labs, 3 pts total
- Lab 1, SPIN
Reporting on Nov 15
 - Lab 2, ESC/Java
Reporting on Nov 23
 - Lab 3, CCS/CWB
Reporting on Dec 8

Final exam, 2 pts, determines final grade

2005 Mads Dam IMIT, KTH

4

2G1516 Formal Methods

On the Labs

No assigned lab facilities!

Deadlines must be respected

Important to get started early

Requirements for each lab determined on course web

To get help:

- Use course mailing list: formal-methods@imit.kth.se
- Ask course assistant at exercise sessions

2005 Mads Dam IMIT, KTH

5

2G1516 Formal Methods

Literature

Slides

Course book:

D. Peled: Software Reliability Methods

Price ~550SEK, available at Kårbokhandeln Campus and Kista

Handouts

Course web

2005 Mads Dam IMIT, KTH

6

2G1516 Formal Methods

Why Formal Methods?

The better you understand the problem and your proposed solution

The better the quality of the end product

Understanding → Properties
 → Modelling
 → Formalisation
 → PROOF!

Formalisation and proof = use of mathematics and logics

Embodied in usable tools

2005 Mads Dam IMIT, KTH

7

2G1516 Formal Methods

The Challenge

Real systems are huge and complex

There is no silver bullet

Problems

- Undecidability: Applies to humans as well as machines
- Distribution: Complexity explodes

Engineering:

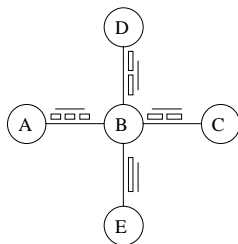
- How to get a satisfactory analysis with the resources (time, space, people, machines) available
- Satisfactory: Correct, or good enough for trust

2005 Mads Dam IMIT, KTH

8

2G1516 Formal Methods

The Sum Is More Than the Sum of the Parts



Each link uses fault free data transfer protocol
 Is the system fault free?

$Used(B) = n \cdot Size(AB) + m \cdot Size(DB)$
 $Free(B) = Mem(B) - Used(B)$

Scenario:

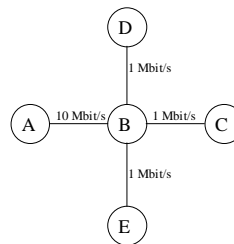
$Min(Size(AB), Size(DB)) > Free(B)$
 $n \cdot Size(AB) < Size(BC)$
 $m \cdot Size(DB) < Size(BE)$

2005 Mads Dam IMIT, KTH

9

2G1516 Formal Methods

Another Example



What happens if we upgrade link AB to 10 Mbit/s?

B permanently saturated
 AC throughput -> 1 Mbit/s
 (But AB transmits at full speed)
 Buffer release rate at B: 2 Mbit/s
 AC traffic has 91% chance of grabbing free buffer
 DE traffic gets < 0.2 Mbit/s

2005 Mads Dam IMIT, KTH

10

2G1516 Formal Methods

The Key Questions

What is the right level of abstraction?

- Physical? Circuit? Network?
- Binaries? Bytecode? Source program? Libraries?
- Architectural? Algorithmic? Process? Model?

What are the important properties?

- Type safety (e.g. Java bytecode verifier)?
- Partial/total correctness? Implementation correctness?
- Deadlocks? Starvation? Reactiveness?

There is no "correct" answer

Answers determine choice of methods and tools

2005 Mads Dam IMIT, KTH

11

2G1516 Formal Methods

Occam's Razor

Abstraction, abstraction, abstraction...

- Eliminate irrelevant details until the important features stick out
- And then eliminate some of these if necessary
- And try to eliminate the bad design decisions that made the analysis too complex - e.g. timing, language, platform dependencies

Use common, well-understood models

- Sequential systems: First-order logic (FOL), recursive functions
- Concurrency and distribution: Automata = state machines, Communicating state machines, process algebra, temporal logic

2005 Mads Dam IMIT, KTH

12

2G1516 Formal Methods

Abstraction

Assume you should verify absence of deadlock between an ATM trying to store a withdrawal and a server

ATM:

```
server!store(self,time,accountno,amount, transactionkind)
```

Server:

```
if correctChecksum(...) and
  validAmount(...) and
  allowedTransaction(...) then
  storeValue(time,accountno,amount,transactionkind);
  client!transactionOK
else
  client!transactionNotOK
endif
```

2005 Mads Dam IMIT, KTH

13

2G1516 Formal Methods

Abstraction, II

- What will the parameters be used for when checking for deadlocks?
- What can the server do which affects the possibility of deadlock?

2005 Mads Dam IMIT, KTH

14

2G1516 Formal Methods

Abstraction, III

Suppose we don't care about memory consumption, very detailed execution time analysis, actual value input/output relations...

ATM:

```
server!store(self)
```

Server:

```
if random() < 0.5 then
  storeValue(); client!transactionOK
else client!transactionNotOK endif
```

2005 Mads Dam IMIT, KTH

15

2G1516 Formal Methods

Main Schools

Theorem proving

- FOL, HOL, type theory, using formalised theories
- Problem of proof search
- Not covered in course this year

Model checking

- Automatic traversal of finite state machine graphs
- Efficient, but scalability issues

Floyd/Hoare logic

- Theorem proving for flow graphs/while programs
- Proof search, but more domain knowledge

Process algebra

- Reasoning about abstract processes
- Structured approach

2005 Mads Dam IMIT, KTH

16

2G1516 Formal Methods

The Field

Formal verification is an ambitious task, so:

- Use with discretion
- Avoid exaggerated expectations
- For engineers, not for programmers

Current industrial usage

- Common criteria EAL 5-7
- Programming language design
 - Operational semantics, type systems
 - Check out the SLAM project at Microsoft
- Processor design
 - all major chip vendors + tool providers
- Safety-critical systems
 - Railway, automotive, power industry, aerospace

2005 Mads Dam IMIT, KTH

17

2G1516 Formal Methods