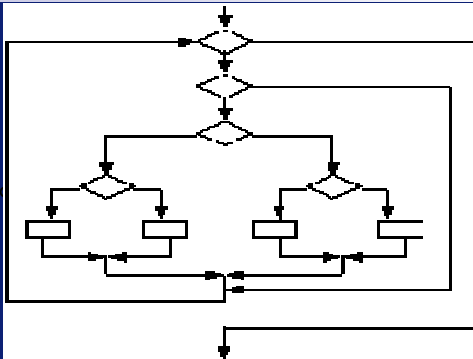


Spring 2009



Program correctness

Overview and motivation

Marcello Bonsangue



Leiden Institute of Advanced Computer Science
Research & Education

Course Information

- Marcello Bonsangue (marcello@liacs.nl)
 - My office: 155a
- Alexandra Silva (ams@cwi.nl)
- Michiel Helvensteijn (mhelvens@liacs.nl)
- All important information on www.liacs.nl/~marcello/penc.html
 - Schedule
 - Old examinations

Visit it regularly



Lectures

- Where: room W1412
- When: Friday (11:15 - 13:00)

February	6	13	20	27
March	6	13	20	27
April	3	17	24	
May	8	15		

Class participation is important



Practice

- Where: room W1412
- When: Friday (13:45 - 15:30)

February	8	13	20	27
March	6	13	20	27
April	3	17	24	
May	8	15		

Class participation is essential



Grading

- This course is worth 5 ECTS
- Evaluation by written examination + 2 home assignments
- Written examination
 - when: **Wednesday 9 June** from 14:00 to 17:00
 - where: room ???

and also

- when: **Thursday 27 Augustus** from 10:00 to 13:00
- where: room ???



Reading

*Logic in Computer Science:
Modelling and Reasoning about Systems*

Michael R. A. Huth and Mark D. Ryan

Cambridge University Press, 2004

ISBN 0 521 54310 X paperback



Expected Background

- Propositional logic
- Predicate logic
- Sets and functions
- Induction
- Recursion

- Imperative programming



Course Organization

- The course cover **model** and **proof-based** techniques for proving programs correct

- The course combines
 - theory (logics)
 - practice (program and system modeling)

- Course goals:
 - introduction to fundamental concepts of formal methods
 - usage of formal methods in software engineering



Formal Methods

- Formal methods includes all applications of mathematics to software engineering problems.
 - type checking
 - model checking

 - program correctness
 - semantics



Formal Methods

- We consider formal methods for **verifying** the **correctness** of computer systems (hardware and/or software)
- Logics provide a mean for mechanizing verification details

computer aided verification

- fully automated (e.g. model checking)
- interactive (e.g. program correctness)



Why?

- **Avoid loss of life**

Therac 25, a computer-controlled radiation therapy machine made by Atomic Energy of Canada killed 6 people by radiation overdoses between 1985 and 1987 because of a timing problem on a data entry:

“An operator mistake could be fixed within 8 seconds, but even though the monitor reflected the operator change, the change did not affected a part of the program”



Why?

- **Save costs**

In 1994, 2 million Intel Pentium V had a bug in the FDIV operation. It could be detected by the following MS-Excel operation:

$$(4195835/3145727) \times 3145727 - 4195835 = 512 !!!$$

- Cost to Intel: \$475 million
- From 1994 Intel applies formal verification techniques to its products



Why?

- **Guarantee security**

In 1998 several e-mail systems did not check for the length of e-mail addresses, and allowed their buffers to overflow causing the applications to crash.

Hostile hackers used this fault to trick the operating system into running a malicious program in its place.



Do you trust your system?

The real wonder is that the system works
as well as it does

(Peterson, 1996)

but remember that software systems provide the infrastructure in virtually all industries today:

- air traffic control
- water level management
- energy production and distribution
- ...



Why Formal Methods?

- Testing and simulation techniques are never exhaustive
- Formal verification proves that a system works based on:
 - mathematical principles
 - exhaustive verification techniques
 - mathematical model structures



Warning

- The use of formal methods does not solve all these problems
 - proof: hand-checked or machine supported?
 - modelling task: difficult and yet crucial!

- Formal methods should be part of a methodology together with
 - Reviews (of requirements, design, and code)
 - Testing (of software units and their integration)

