CS6320 Formal Methods for Distributed Systems

Dr. John Herbert
Western Gateway Building, Room 1.78
j.herbert@cs.ucc.ie

Lectures: Thurs 9-11am
Tutorial/Lab: Weds 12-1pm? (from week 4)

Canvas site

Module Objective:
To introduce students to the concepts and techniques of Formal Methods and how they are applied in practice.

Module Content:
- Overview of standard mathematical logics including Predicate Calculus, Higher-Order Logic and Temporal Logic.
- Semantics and Proof.
- Soundness and Completeness.
- Formal specification of functional requirements of systems.
- Formal specification of network communication protocols.
- Safety and Liveness properties.
- Tool-based formal analysis and verification of communication protocols.
- Specification and verification of programs.

Overview

"Computer scientists can prove certain programs to be error-free with the same certainty that mathematicians prove theorems. The advances are being used to secure everything from unmanned drones to the internet. "

"The technology that repelled the hackers was a style of software programming known as formal verification. Unlike most computer code, which is written informally and evaluated based mainly on whether it works, formally verified software reads like a mathematical proof: Each statement follows logically from the preceding one. An entire program can be tested with the same certainty that mathematicians prove theorems.
"You're writing down a mathematical formula that describes the program's behavior and using some sort of proof checker that's going to check the correctness of that statement," said Bryan Parno, who does research on formal verification and security at Microsoft Research.
The aspiration to create formally verified software has existed nearly as long as the field of computer science. For a long time it seemed hopelessly out of reach, but advances over the past decade in so-called "formal methods" have inched the approach closer to mainstream practice. Today formal software verification is being explored in well-funded academic collaborations, the U.S. military and technology companies such as Microsoft and Amazon."

Formal Verification Creates Hacker-Proof Code
Quanta Magazine, September 2016

"In industry, formal methods have a reputation of requiring a huge amount of training and effort to verify a tiny piece of relatively straightforward code, so the return on investment is only justified in safety critical domains such as medical systems and avionics. Our experience with TLA+ has shown that perception to be quite wrong. So far we have used TLA+ on 10 large complex real-world systems. In every case TLA+ has added significant value, either finding subtle bugs that we are sure we would not have found by other means, or giving us enough understanding and confidence to make aggressive performance optimizations without sacrificing correctness. "

Use of Formal Methods at Amazon Web Services
Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, Michael Deardeuff, 2014

Initial Reading List:

There will be no single prescribed textbook for the course. Some topics will be based on a stated textbook, and more reading recommendations will be given when appropriate during the course. Additional material will be made available as handouts as required.

Mathematical Logic and Formal Modelling

The basics are covered in all the discrete mathematics books. Handouts will cover the basics and any specialized material.

Logic in Computer Science, *Michael Huth and Mark Ryan*

Model checking

Handouts include model checking, the SMV User Guide, SMV tutorial, Book chapter on CTL and model checking.

Background articles:

A look at its use in practice:
*Formal Methods: Practice and Experience***,** Jim Woodcock et al. , ACM Survey, 2009

Remarks by Bill Gates at the 17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages and Application, 2002

The Exterminators, A small British firm shows that software bugs aren't inevitable, IEEE Spectrum September 2005

Classic articles:

C.A.R. Hoare, The Emperor's Old Clothes. Communications of the ACM, Volume 24, No. 2, 1981, pp. 75–83. Newly edited available online.

E. Dijkstra, On the cruelty of really teaching computing science, Available at:
www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html

Some Application Case Studies

Formal Verification Creates Hacker-Proof Code
https://www.quantamagazine.org/formal-verification-creates-hacker-proof-code-20160920

Software
MS
Microsoft - RISE Formal Methods Working Group
https://www.microsoft.com/en-us/research/group/rise-working-group-on-formal-methods/

IBM
https://www.research.ibm.com/haifa/projects/verification/RB_Homepage/papers/wp_formal_verification_1.pdf

Amazon
Use of Formal Methods at Amazon Web Services Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, Michael Deardeuff, Amazon.com, September, 2014
http://lamport.azurewebsites.net/tla/formal-methods-amazon.pdf

Digital Hardware
Intel
Limor Fix. Fifteen years of formal property verification in Intel. In: Orna Grumberg and Helmut Veith (eds.), 25 Years of Model Checking, Lecture Notes in Computer Science, vol. 5000, pp. 139–144. Springer, 2008.

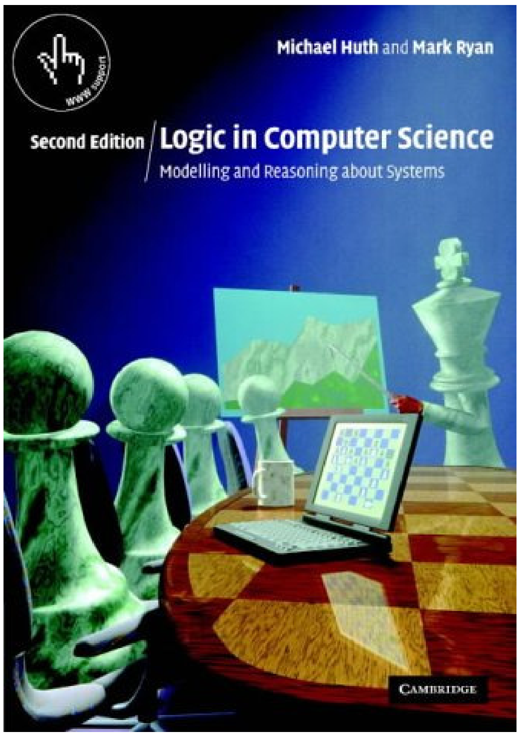
Roope Kaivola et al. Replacing testing with formal verification in Intel® Core™ i7 processor execution engine validation. In: Ahmed Bouajjani and Oded Maler (eds.), Computer Aided Verification, Lecture Notes in Computer Science, vol. 5643, pp. 414– 429. Springer, 2009.

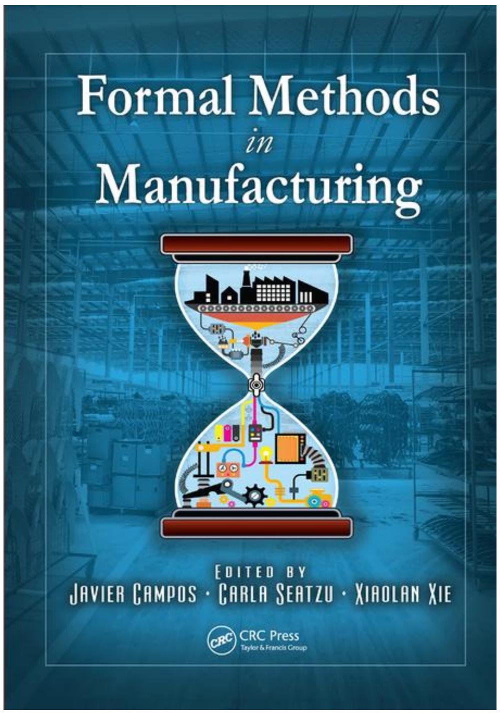
Train Safety
Development of formal method application for ensuring safety in train control system, Hyun-Jeong Jo; Jong-Gyu Hwang; Yong-Ki Yoon, Korea Railroad Research Institute. World Congress on Railway Research (WCRR) 2008, Seoul, Korea

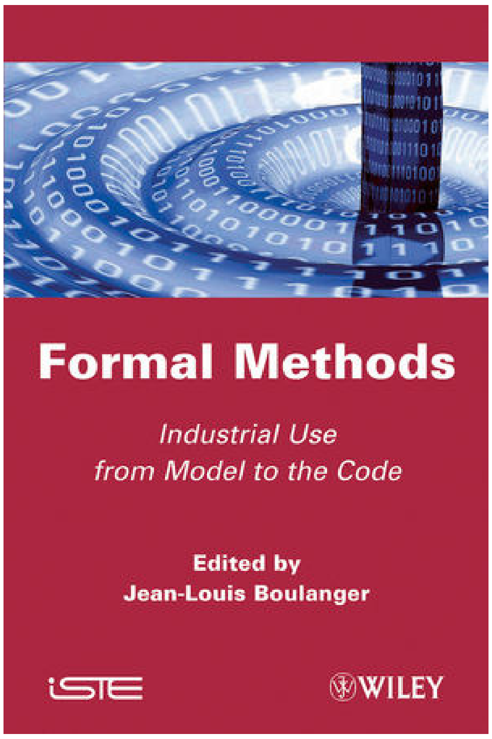
Air Traffic Control
The Use of Formal Methods on the iFACTS Air Traffic Control Project
Neil White, https://www.slideshare.net/AdaCore/white-open-do

Security
Formal methods for web security, Michele Bugliesi, Stefano Calzavara, Riccardo Focardi, Journal of Logical and Algebraic Methods in Programming, Volume 87, February 2017, Pages 110-126

http://www.cs.bham.ac.uk/research/projects/lics/


Michael Huth and Mark Ryan
www support
Second Edition
Logic in Computer Science
Modelling and Reasoning about Systems
CAMBRIDGE


Formal Methods in Manufacturing
Edited by
Javier Campos · Carla Seatzu · Xiaolan Xie
CRC Press
Taylor & Francis Group


Formal Methods

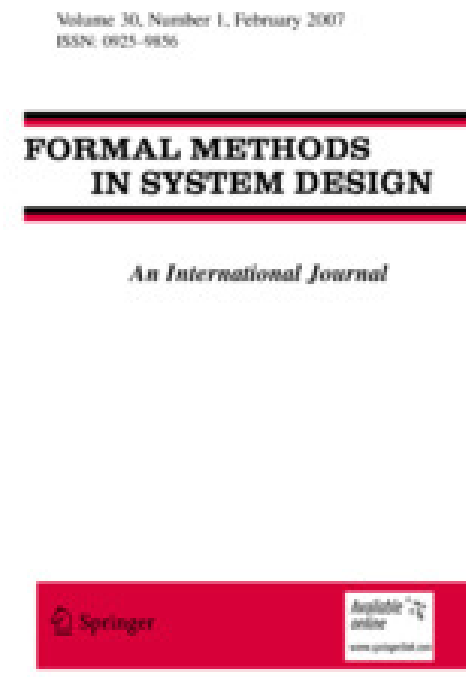
Industrial Use from Model to the Code

Edited by Jean-Louis Boulanger

iSTE WILEY

https://images.springer.com/sgw/journals/medium/10703.jpg

May 2012


Volume 30, Number 1, February 2007
ISSN: 0925-9856

FORMAL METHODS IN SYSTEM DESIGN

An International Journal

Springer

Available online
www.springerlink.com

What are Formal Methods?

Nasa Langley:

Traditional engineering disciplines rely heavily on mathematical models and calculation to make judgments about designs. For example, aeronautical engineers make extensive use of computational fluid dynamics (CFD) to calculate and predicte how particular airframe designs will behave in flight. We use the term ``formal methods'' to refer to the variety of mathematical modelling techniques that are applicable to computer system (software and hardware) design. That is, formal methods is the applied mathematics of computer system engineering, and, when properly applied, can serve a role in computer system design analogous to the role CFD serves in aeronautical design.
Formal methods may used to specify and model the behavior of a system and to mathematically verify that the system design and implementation satisfy system functional and safety properties. These specifications, models, and verifications may be done using a variety of techniques and with various degrees of rigor. The following is an imperfect, but useful, taxonomy of the degrees of rigor in formal methods:

Level-1: Formal specification of all or part of the system.

Level-2: Formal specification at two or more levels of abstraction and paper and pencil proofs that the detailed specification implies the more abstract specification.

Level-3: Formal proofs checked by a mechanical theorem prover.

Level 1 represents the use of mathematical logic or a specification language that has a formal semantics to specify the system. This can be done at several levels of abstraction. For example, one level might enumerate the required abstract properties of the system, while another level describes an implementation that is algorithmic in style.
Level 2 formal methods goes beyond Level 1 by developing pencil-and-paper proofs that the more concrete levels logically imply the more abstract-property oriented levels. This is usually done in the manner illustrated below.
Level 3 is the most rigorous application of formal methods. Here one uses a semi-automatic theorem prover to make sure that all of the proofs are valid. The *Level 3* process of *convincing* a mechanical prover is really a process of developing an argument for an ultimate skeptic who must be shown every detail.

Formal methods is not an all-or-nothing approach. The application of formal methods to only the most critical portions of a system is a pragmatic and useful strategy. Although a complete formal verification of a large complex system is impractical at this time, a great increase in confidence in the system can be obtained by the use of formal methods at key locations in the system.

CS6320 Formal Methods for Distributed Systems Computer Science

Credit Weighting: 5

Semester(s): Semester 1.

No. of Students: Min 5.

Pre-requisite(s): None

Co-requisite(s): None

Teaching Method(s): 24 x 1hr(s) Lectures; 10 x 1hr(s) Practicals (Tutorials and Laboratory

Module Co-ordinator: Dr John Herbert, Department of Computer Science.

Lecturer(s): Dr John Herbert, Department of Computer Science.

Module Objective: To introduce students to the concepts and techniques of Formal Methods and how they are applied in practice.

Module Content:
Overview of standard mathematical logics including Predicate Calculus, Higher-Order Logic and Temporal Logic.
Semantics and Proof.
Soundness and Completeness.
Formal specification of functional requirements of systems.
Formal specification of network communication protocols.
Safety and Liveness properties.
Tool-based formal analysis and verification of communication protocols.
Specification and verification of programs.

Learning Outcomes: On successful completion of this module, students should be able to:

Explain the underlying concepts of mathematical logic, correctness and proof;

Specify properties of simple systems using higher-order logic and temporal logic;

Specify a communications protocol;

Understand safety and liveness properties;

Verify, using an automated tool, basic communications protocols;

Explain how code can be specified and formally verified.

Assessment: Total Marks 100: Continuous Assessment 100 marks (1 x Mid-Term Examination 40 marks, 1 x End of Module Examination 40 marks; 2 x In-class assignments, 10 marks each).

Compulsory Elements: Continuous Assessment.

Penalties (for late submission of Course/Project Work etc.): Work which is submitted late shall be assigned a mark of zero (or a Fail Judgement in the case of Pass/Fail modules).

Pass Standard and any Special Requirements for Passing Module: 40%.

Formal Written Examination: No Formal Written Examination.

Requirements for Supplemental Examination: 1 x 1.5 hr(s) paper(s) (corresponding to Mid-Term Examination and End of Module Examination) to be taken in Autumn 2019. Marks in passed element(s) of Continuous Assessment are carried forward, Failed element(s) of Continuous Assessment must be repeated (as specified by the Module Coordinator).