CS6320

Formal Methods for Distributed Systems

Dr. John Herbert

1.78 Western Gateway Building
j.herbert@cs.ucc.ie

CS6320 Overview

Mathematical basics:

- Overview of standard mathematical logics including:
  - Predicate Calculus,
  - Higher-Order Logic, and
  - Temporal Logic.
- Semantics and Proof.
- Soundness and Completeness.

CS6320 Overview

Using the mathematical techniques …

- 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.
  - Design by contract

CS6320: general emphasis

- Basic concepts underlying:
  - System requirements/behaviour specification
  - Implementation models
  - Correctness of implementation wrt requirements
- Application to hardware, software, distributed systems
- Use of automated tools, e.g.
  - SMV (model checking algorithms)
  - LTSA/FSP (for concurrency)

Formal Methods

- How we think about systems and their requirements is influenced by the abstract models and notations used to describe them.
- Formal mathematical models have different flavours appropriate for different types of modelling.
- Our bias in this course is modelling rather than proof but we make look at automatic analysis

Formal Methods

- Formal methods are mathematical approaches to software and system development which support the rigorous specification, design and verification of computer systems.

Information System Engineering

- It is sometimes argued that engineering is not possible for information systems.
- A civil engineer can design a bridge, confident that it will meet its requirements when built.
- Can a software engineer be confident that their design will meet its requirements?
- Why is engineering of information systems difficult?
  - Your opinion?
  - Many aspects …

Information System Engineering

- This course introduces theories and techniques that can provide the same level of precision for information systems that is available and required for other engineering disciplines

- As well as presenting these techniques, we will also be asking the question “Are these formal models and theories practical?”.

Conventional Practice

Requirements:

- stated as natural language text / user stories / use case text

Implementation:

- Coded in appropriate language + use of libraries, frameworks

Verification: does what is built satisfy requirements?

- Testing, prototyping, QA, etc.

Limitations of practice

- Requirements stated in natural language text are not precise
- The code does not convey the abstract behaviour it is trying to achieve

And so,

- We never directly relate the code to the overall problem requirements

Need precise models and methods based on the models

Motivation

Our goal in FM is to improve the situation:

- Use formal models to represent:

  What the problem is: requirements *(specification)*

  How we solve it: algorithm, code *(implementation)*

- Use formal mathematics-based techniques
  - to analyse models
  - to relate models,

  e.g. demonstrate that the code meets requirements

Motivation

- There are many different kinds of Formal Methods
- Formal Methods can be applied at different levels of detail/precision for different designs and for different parts of the same design
- Without ever applying the techniques, it is very empowering to understand what lies behind the techniques, and ultimately what lies behind the specification, building and verification of all complex systems.