School of Informatics Teaching Course Proposal Form

This version was generated **October 28, 2015**. User 'pbj@inf.ed.ac.uk' verified.

Proposal

Course Name: Formal Verification
Proposer's Name: Paul Jackson
Email Address: pbj@inf.ed.ac.uk
Course Year: 5
Names of any courses that this new course replaces :
Automated Reasoning with its current syllabus

Course Outline

Course Level: 11
Course Points: 10
Subject area: Informatics
Programme Collections:
Computer Science, Software Engineering, Artificial Intelligence.

Teaching / Assessment

Number of Lectures: 15
Number of Tutorials or Lab Sessions: 6
Identified Pre-requisite Courses: none
Identified Co-requisite Courses: none
Identified Prohibited Combinations: Current Automated Reasoning course

Assessment Weightings:
Written Examination: 60%
Assessed Coursework: 40%
Oral Presentations: 0%

Description of Nature of Assessment:
Two assessed courseworks are set, one using a model checker such as NuSMV or Spin, the other an assertion-based software formal verification tool such as Why3 with a C, Java or SPARK front end or some Boogie-related tool such as Dafny.

Course Details

Brief Course Description:
The course provides an introduction to practical automated techniques for the formal verification of hardware and software systems.

A strong emphasis of the course is on giving students hands-on experience using state-of-the-art tools, so they feel comfortable exploring further the use of these tools or similar tools in their future careers. Assessed courseworks focus on the use of tools of primary importance, and unassessed lab work is provided that introduces further tools.

Detailed list of Learning Objectives:

On completion of this course, the student will be able to:

1: Explain the meaning of CTL and LTL formulas and construct models that distinguish similar formulas

2: Construct CTL and LTL formulas to check properties expressed in natural language

3: Explain the basics of the algorithms used in model checking

4: Translate state-machine like system descriptions into the input language of a model checker

5: Use a bounded or unbounded model checker to either verify properties of interest of a provided model

6: Apply their knowledge of semantic models of imperative programming languages to calculate verification conditions for simple programs

7: Recognise the theories a verification condition makes use of in order to predict the likelihood of an SMT solver proving the verification condition

8: Write preconditions and postconditions for small programs such as found in common program libraries

9: Infer the loop invariants needed for the proof of correctness of simple loops

10: Use an assertion-based formal software formal verification tool to verify desired properties of a previously unseen small program

11: Describe formal techniques that can be used for the detection of concurrency bugs

12: Assess how likely it is that formal verification could be successful on a previously-unseen hardware or software verification problem

13: Describe the successes of current formal verification techniques

Syllabus Information:

Topics covered include:

- Formal verification in context, its current take up in industry and challenges to its wider adoption
- CTL and LTL temporal logics - their syntax and semantics
- CTL and LTL model checking techniques, including automata-based approaches and bounded model checking.
- Writing models for model checking and phrasing useful properties in CTL and LTL.
- Operational semantics of a simple imperative programming language, weakest precondition operators and verification condition generation.
- The capabilities of SMT solvers for discharging verification conditions.
- Assertion-based software verification
- Software model checking, focussing on its use for finding concurrency bugs
- Pattern-based detection of concurrency bugs

Optional topics include:

- Hardware temporal logics such as PSL and SVA
- Formal verification case studies
- Formal verification of hybrid systems
- Combining static and dynamic verification methods
- Dual use of temporal logic properties and assertions in the both static and run-time checking of hardware and software

Recommended Reading List:

- Logic in Computer Science (2nd Ed) by Huth and Ryan. Cambridge UP. 2004.
- A few case study papers and introductory papers on tools.

Any additional case for support information:

This course and a new Automated Reasoning course would replace the current Automated Reasoning course. The material for the half of this course on model checking related topics would be taken directly from the current Automated Reasoning course. The other half of the course on the formal verification of software would mostly be new material.

There is a small overlap between this new material and material taught on the Types and Semantics for Programming Languages and the Advances in Programming Languages courses, not enough though for this proposed course and either of those courses to be prohibited combinations.

though some theoretical elements have been taught previously

The half of this course material on model checking would be taken directly from the current Automated Reasoning course. The course would be suitable as an optional course for PPAR CDT students in their MSc by Research year.