THE UNIVERSITY *of* EDINBURGH
informatics

Board of Studies
Course Proposal Template

PROPOSED COURSE TITLE: Formal Verification

PROPOSER(S): Paul Jackson

DATE: 9th February 2016

SUMMARY

This template contains the following sections, which should be prepared roughly in the order in which they appear (to avoid spending too much time on preparation of proposals that are unlikely to be approved):

1. Case for Support
– To be supplied by the proposer and shown to the BoS Academic Secretary prior to preparation of an in-depth course description

1a. Overall contribution to teaching portfolio

1b. Target audience and expected demand

1c. Relation to existing curriculum

1d. Resources

2. Course descriptor
- This is the official course documentation that will be published if the course is approved, ITO and the BoS Academic Secretary can assist in its preparation

3. Course materials
- These should be prepared once the Board meeting at which the proposal will be discussed has been specified

3a. Sample exam question

3b. Sample coursework specification

3c. Sample tutorial/lab sheet question

3d. Any other relevant materials

4. Course management

- This information can be compiled in parallel to the elicitation of comments for section 5.

4a. Course information and publicity

4b. Feedback

4c. Management of teaching delivery

5. Comments

- To be collected by the proposer in good time before the actual BoS meeting and included as received

5a. Year Organiser Comments

5b. Degree Programme Co-Ordinators

5c. BoS Academic Secretary

[Guidance in square brackets below each item. Please also refer to the guidance for new course proposals at http://www.inf.ed.ac.uk/student-services/committees/board-of-studies/course-proposal-guidelines. Examples of previous course proposal submissions are available on the past meetings page http://www.inf.ed.ac.uk/admin/committees/bos/meetings/.]

SECTION 1 – CASE FOR SUPPORT

[This section should summarise why the new course is needed, how it fits with the existing course portfolio, the curricula of our Degree Programmes, and delivery of teaching for the different years it would affect.]

1a. Overall contribution to teaching portfolio

[Explain what motivates the course proposal, e.g. an emergent or maturing research area, a previous course having become outdated or inappropriate in other ways, novel research activity or newly acquired expertise in the School, offerings of our competitors.]

There are several motivations for this proposal:

- Formal verification is an increasingly important verification approach in industrial digital hardware design processes. It also is becoming important in some software development processes, most notably in safety-critical application areas and when concurrency problems lead to total system failures.
- Formal verification is a showcase for computer science theory (e.g. automata theory and programming language semantics) and automated reasoning (model checking and SMT solving in particular), both areas of strength for the School of Informatics.
- The current main School offering in the formal verification area is the half of the current Automated Reasoning course which covers hardware model checking. Formal verification of software is currently only touched on briefly by the Level 11 Advances in Programming Languages course. This proposal aims to integrate the model checking half of the AR course with significant new material on software formal verification techniques.
- The 2008 Informatics Teaching Programme Review recommended increased use of formal notations and verification techniques in our curriculum. At the verbal feedback session, one of the external reviewers, Martyn Thomas, expressed surprise that a School with such strong work in the foundations of formal methods did not incorporate more applied formal methods material in its curriculum.
- As the proposal addresses the verification of hardware and parallel software, it is an attractive option for post-graduate students on the School's PPAR (Pervasive Parallelism) CDT programme

1b. Target audience and expected demand

[Describe the type of student the course would appeal to in terms of background, level of ability, and interests, and the expected class size for the course based on anticipated demand. A good justification would include some evidence, e.g. by referring to projects in an area, class sizes in similar courses, employer demand for the skills taught in the course, etc.]

The course is aimed at a broad range of students:

1. Those considering an industrial career in software engineering or digital hardware engineering. Formal verification tools are used now in some parts of industry, but uptake is limited by lack of awareness of formal verification techniques and knowledge of relevant foundation subjects, using temporal logics for specification and using logic to express desired program properties, for example.
2. PhD students undertaking research where formal verification techniques would be useful. For example, students in ICSA designing new hardware or hardware-related protocols could well use model checking to help verify new ideas.
3. Those who might be interested in pursuing research in new formal verification techniques. While most of the topics covered do not directly lead in to active research of my own or others in Informatics, an understanding of the motivation and capabilities of current formal verification techniques would provide strong motivation for research in several areas of active interest.

I would hope to attract at least 20% of the 4th year and MSc cohorts, say 60 students. I expect it to have broader appeal than the current Automated Reasoning course which has 50 students this year.

1c. Relation to existing curriculum

[This section should describe how the proposed course relates to existing courses, programmes, years of study, and specialisms. Every new course should make an important contribution to the delivery of our Degree Programmes, which are described at http://www.drps.ed.ac.uk/15-16/dpt/drps_inf.htm.

Please name the Programmes the course will contribute to, and justify its contribution in relation to courses already available within those programmes. For courses available to MSc students, describe which specialism(s) the course should be listed under (see http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2015/programme-guide/specialist-areas), and what its significance for the specialism would be. Comment on the fit of the proposed course with the structure of academic years for which it should be offered. This is described in the Year Guides linked from http://web.inf.ed.ac.uk/infweb/student-services/ito/students.]

BSc Honours, BEng Honours, MInf Undergraduate Masters in Informatics: this proposed course is strongly relevant to students in their 4th and 5th year on virtually all these programmes, as we expect a large fraction of our students to be going on to software engineering jobs of one kind or another.

Taught MSc: for similar reasons, this course is strongly relevant to MSc students on our *Informatics*, *Computer Science* and *Artificial Intelligence* programmes taking a number of the available specialisms: most particularly *Computer Systems, Software Engineering & High-Performance Computing*, and also *Cyber-Security and Privacy*, *Agents, Knowledge and Data*, and *Theoretical Computer Science.*

1d. Resources

[While course approvals do not anticipate the School's decision that a course will actually be taught in any given year, it is important to describe what resources would be required if it were run. Please describe how much lecturing, tutoring, exam preparation and marking effort will be required in steady state, and any additional resources that will be required to set the course up for the first time. Please make sure that you provide estimates relative to class size if there are natural limits to its scalability (e.g. due to equipment or space requirements). Describe the profile of the course team, including lecturer, tutors, markers, and their required background. Where possible, identify a set of specific lecturers who have confirmed that they would either like to teach this course apart from the proposer, or who could teach the course in principle. It is useful to include ideas and suggestions for potential teaching duty re-allocation (e.g. through course sharing, discontinuation of an existing course, voluntary teaching over and above normal teaching duties) to be taken into account when resourcing decisions are made.]

Annual time budgets for course-related activities, both steady-state and first year:

- **Lecturing (Lecturer)**: *Steady state*: 6 hrs/lecture to review, prepare updates, and deliver. With 15 lectures, this is 90 hrs.
  First year: 6 lectures can be brought in from the current AR course. 8 new lectures needed, each needing perhaps 20 hours to prepare and deliver.
  Total hours = 6*6hrs + 8*20 hrs = 196 hrs.
- **Formative coursework preparation (Lecturer + TA)**: *Steady state*: a key issue with this course will be keeping awareness of the best exemplars of formal verification tools up to date and continually evolving the materials guiding the students in the use of the tools.
  20 + 20 hrs.
  First year: Current hardware model checking coursework for AR could easily be adapted. New courseworks needed for software verification. I anticipate these can be adapted from existing tutorial materials that accompany the selected verification tools. 40 + 20 hrs.
- **Coursework support (Lecturer + Demonstrator)**: *Steady state and first year*: 10 + 20 hrs for 60 students. Scale this with the number of students. Activities here would include providing guidance during lab sessions, informal formative assessment of student work in booked time-slots, introduction of courseworks in lecture slots, presentation and discussion of sample solutions in lecture slots.
- **Exam preparation (Lecturer)**: *Steady state*: 30 hrs. *First two years*: 40 hrs.
- **Exam marking (Lecturer)**: *Steady state and first year*: Allowing 0.5hrs/student, for 60 students, 30 hrs.

The TA and/or demonstrator (possibly the same person) will need to be someone who comes to the course already with some familiarity with formal verification techniques, perhaps through taking the course in some previous year.

Possible alternate lecturers who could teach the course in principle (* = confirmed):

Stuart Anderson. *David Aspinall. Julian Bradfield. James Cheney. Jacques Fleuriot. Michael Fourman. Stephen Gilmore. Richard Mayr. *Ajitha Rajan. *Donald Sannella. *Ian Stark. *Perdita Stevens.

If this course were to go ahead, then resources used by the current Automated Reasoning course (e.g. my lecturing on half the course) would be freed.

SECTION 2 – COURSE DESCRIPTOR

[This is the official course descriptor that will be published by the University and serves as the authoritative source of information about the course for student via DRPS and PATH. Current course descriptions in the EUCLID Course Catalogue are available at www.euclid.ed.ac.uk under 'DPTs and Courses', searching for courses beginning 'INFR'*]*

2a. Course Title *[Name of the course.]* :

Formal Verification

2b. SCQF Credit Points:

[The Scottish Credit and Qualifications Framework specifies where each training component provided by educational institutions fits into the national education system. Credit points per course are normally 10 or 20, and a student normally enrols for 60 credits per semester. For those familiar with the ECTS system, one ECTS credit is equivalent to 2 SCQF credits. See also http://www.scqf.org.uk/The%20Framework/Credit%20Points.]

10

SCQF Credit Level:

[These levels correspond to different levels of skills and outcomes, see http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf. At University level, Year 1/2 courses are normally level 8, Year 3 can be level 9 or 10, Year 4 10 or 11, and Year 5/MSc have to be level 11. MSc programmes may permit a small number (up to 30 credits overall) of level 9 or 10 courses.]

11

Normal Year Taken: 1/2/3/4/5/MSc

[While a course may be available for more than one year, this should specify when it is normally taken by a student. "5" here indicates the fifth year of undergraduate Masters programmes such as MInf.]

4

Also available in years: 1/2/3/4/5/MSc

Different options are possible depending on the choice of SCQF Credit Level above: for level 9, you should specify if the course is for 3rd year undergraduates only, or also open to MSc students (default); for level 10, you should specify if the course is available to 3rd year and 4th year undergraduates (default), 4th year undergraduates only, and whether it should be open to MSc students; for level 11, a course can be available to 4th and 5th year undergraduates and MSc students (default), to 5th year undergraduates and MSc students, or to MSc students only]

5 & MSc

2c. Subject Area and Specialism Classification:

[Any combination of Computer Science, Artificial Intelligence, Software Engineering and/or Cognitive Science as appropriate. For courses available to MSc students, please also specify the relevant MSc specialist area (to be found in the online MSc Year Guide at http://web.inf.ed.ac.uk/infweb/student-services/ito/students/taught-msc-2015/programme-guide/specialist-areas), distinguishing between whether the course should be considered as "core" or "optional" for the respective specialist area.]

UG: *Computer Science, Artificial Intelligence, Software Engineering*.

MSc: Optional course for specialist areas: *Computer Systems, Software Engineering & High-Performance Computing*, *Cyber-Security and Privacy*, *Agents, Knowledge and Data*, *Theoretical Computer Science*

Appropriate/Important for the Following Degree Programmes:

[Please check against programmes from http://www.drps.ed.ac.uk/15-16/dpt/drps_inf.htm to determine any specific programmes for which the course would be relevant (in many cases, information about the Subject Area classification above will be sufficient and specific programmes do not have to be specified). Some courses may be specifically designed for non-Informatics students or with students with a specific profile as a potential audience, please describe this here if appropriate.]

Course is appropriate for

- all Undergraduate Degree Programmes apart from *Cognitive Science*,
- Postgraduate Taught Programmes: *Artificial Intelligence, Computer Science* and *Informatics*,
- MSc year of *Pervasive Parallelism* Postgraduate Research Programme.

Timetabling Information:

[Provide details on the semester the course should be offered in, specifying any timetabling constraints to be considered (e.g. overlap of popular combinations, other specialism courses, external courses etc).]

The course does not depend on other courses, so it could run in either semester. The material nicely complements that in the new Automated Reasoning course proposed by Jacques Fleuriot. The new AR course focuses on interactive theorem proving techniques, which also can be used for formal verification. To cater for students who might wish to take both, it might be good to schedule this FV course and the AR course in different semesters: say AR Semester 1 and FV Semester 2.

2d. Summary Course Description:

[Provide a brief official description of the course, around 100 words. This should be worded in a student-friendly way, it is the part of the descriptor a student is most likely to read.]

Formal verification is the use of mathematical techniques to verify the correctness of various kinds of engineering systems: software systems and digital hardware systems, for example. Formal verification techniques are exhaustive and provide much stronger guarantees of correctness than testing or simulation-based approaches. They are particularly useful for safety and security critical systems and for when system behaviour is highly complex. The course focuses on automated techniques that are currently used in industry. It gives practical exposure to current formal verification tools, explaining the input languages used and introducing the underlying mathematical techniques and algorithms used for automation. [98 words]

Course Description:

[Provide an academic description, an outline of the content covered by the course and a description of the learning experience students can expect to get. See guidance notes at: http://www.studentsystems.is.ed.ac.uk/staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html#AcademicDescription].

In recent years there have been highly noteworthy cases of the adoption of formal verification (FV) techniques in industry. For example, at Intel, FV has largely replaced simulation-based verification of their microprocessors, at Microsoft, FV is used to certify that 3rd –party drivers are free of certain kinds of concurrency bugs. As FV tools and methodologies improve, FV is expected to become more and more widely used in industry.

This course aims to familiarise students with main classes of FV techniques that are likely to become most widespread in industry in the coming years. The intent is to prepare students who might go into industry with sufficient background in FV that they would be aware of when and how they might deploy FV techniques. The course will also be of interest to students who wish to go into research developing techniques for future-generation FV tools and who might need to use FV in their research. To satisfy these aims, the course has a practical focus, giving students hands-on experience with a number of tools and explaining their input languages for specifying systems and desired system properties. The course also introduces the underlying mathematical techniques, which gives students a deeper understanding of the tools and will help them use the tools most effectively.

Topics the course covers include the following:

- Formal verification in context, its current take-up in industry and challenges to its wider adoption
- Syntax and semantics of CTL and LTL temporal logics
- CTL and LTL model checking techniques, including automata-based approaches and bounded model checking with SAT solvers
- The BDD data-structure used at the heart of many model checkers
- 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, focusing on its use for finding concurrency bugs
- Pattern-based detection of concurrency bugs

Optional topics include:

- Industrial temporal logics such as PSL and SVA used in hardware verification
- Formal verification case studies
- Formal verification of hybrid systems, system with both discrete state changes and continuous state changes governed by differential equations
- Combining formal and simulation-based verification methods
- Dual use of temporal logic properties and assertions in formal and simulation-based verification of hardware and software

The course material is primarily introduced in lectures and assessment is by a final exam. Practical exercises are provided to give students familiarity with formal verification tools and help them better understand formal verification techniques. Support for students in using the tools is provided both by demonstrators in lab sessions and through the use of an online discussion forum. In addition, demonstrators review student solutions to exercises and the lecturer both introduces the exercises and presents model solutions.

Pre-Requisite Courses:

[Specify any courses that a student must have taken to be permitted to take this course. Pre-requisites listed in this section can only be waived by special permission from the School's Curriculum Approval Officer, so they should be treated as "must-have". By default, you may assume that any student who will register for the course has taken those courses compulsory for the degree for which the course is listed in previous years. Please include the FULL course name and course code].

None.

Co-Requisite Courses:

[Specify any courses that should be taken in parallel with the existing course. Note that this leads to a timetabling constraint that should be mentioned elsewhere in the proposal. Please include the FULL course name and course code].

None.

Prohibited Combinations:

[Specify any courses that should not be taken in combination with the proposed course. Please include the FULL course name and course code].

Automated Reasoning (Level 11) (INFR11074) as currently run. This clash should not be a problem as this new FV course is being proposed a part of a major reorganisation of the current AR course into this new course and a new AR course that focuses on techniques used in interactive theorem proving. The new FV and AR courses complement each other and cover mutually-exclusive material.

Other Requirements:

[Please list any further background students should have, including, for example, mathematical skills, programming ability, experimentation/lab experience, etc. It is important to consider that unless there are formal prerequisites for participation in a course, other Schools can register their students onto our courses, so it is important to be clear in this section. If you want to only permit this by special permission, a statement like "Successful completion of Year X of an Informatics Single or Combined Honours Degree, or equivalent by permission of the School." can be included.]

Incoming students are expected to be familiar with discrete maths at a level similar to that taught in the School of Informatics course *Discrete Mathematics and Mathematical Reasoning* (INFR08023). Prior exposure to predicate logic is also helpful. Programming experience in an imperative language such as Java, C or C++ is also essential for handling the material related to software verification. For the hardware verification aspects of the course, prior exposure to hardware design is not needed, but students do need to be familiar with Finite-State Automata concepts.

Available to Visiting Students: Yes/No

[Provide a justification if the answer is No.]

Yes.

2e. Summary of Intended Learning Outcomes (MAXIMUM OF 5):

[List the learning outcomes of the course, emphasising what the impact of the course will be on an individual who successfully completes it, rather than the activity that will lead to this outcome. Further guidance is available from https://canvas.instructure.com/courses/801386/files/24062695]

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

1. deploy bounded and unbounded model checking techniques to formally verify temporal logic properties of digital hardware and other finite state systems and protocols,
2. use an assertion-based software formal-verification tool to verify desired properties of computer programs,
3. describe formal techniques that can be used for the detection of concurrency bugs in software,
4. assess the pros and cons of using different automated formal verification approaches on a previously-unseen hardware or software system.

Assessment Information

[Provide a description of all types of assessment that will be used in the course (e.g. written exam, oral presentation, essay, programming practical, etc) and how each of them will assess the intended learning outcomes listed above. Where coursework involves group work, it is important to remember that every student has to be assessed individually for their contribution to any jointly produced piece of work. Please include any minimum requirements for assessment components e.g. student must pass all individual pieces of assessment as well as course overall].

Assessment is all by written exam.

As the course focuses on the application of formal verification techniques, most exam questions will involve presenting various verification scenarios and exploring how verification techniques might be deployed in those scenarios. Often questions will focus on some particular aspect of a techniques, exploring for example how well students understand the input specification languages used by techniques. Other questions might involve students stepping through the behaviour of verification algorithms on small examples.

Assessment Weightings:

Written Examination: 100%

Practical Examination: 0%

Coursework: 0%

Time spend on assignments:

[Weightings up to a 70/30 split between exam and coursework are considered standard, any higher coursework percentage requires a specific justification. The general expectation is that a 10-point course will have an 80/20 split and include the equivalent of one 20-hour coursework assignment (although this can be split into several smaller pieces of coursework. The Practical Examination category should be used for courses with programming exams. You should not expect that during term time a student will have more than 2-4 hours to spend on a single assignment for a course per week. Please note that it is possible, and in many cases desirable, to include formative assignments which are not formally assessed but submitted for feedback, often in combination with peer assessment.]

The course expects each student to put in 30-40hrs on the practical exercises. These exercises are formative in nature: there is no formal assessment, but demonstrators review students answers with them, and the lecturer presents and discusses model answers. In addition, students receive support while undertaking the exercises from demonstrators in scheduled labs and from interactions with demonstrators, the lecturer and each other on a online discussion forum.

Academic description:

[A more technical summary of the course aims and contents. May include terminology and technical content that might be more relevant to colleagues and administrators than to students.]

A number of details are already provided above in the above Course Description section. A few further remarks are as follows

- The model checking is likely to focus on the use of the NuSMV tool. This is a mature free tool that well illustrates the range of introductory concepts the course covers.
- For assertion-based software verification, a couple of alternative tools might be used:
    - The SPARK Ada toolset from Altran UK and AdaCore. This is the most mature assertion-based tool available. It is currently used in a number of commercial projects and both tool and thorough training materials are available for free.
    - Why3 from INRIA, France. This is at the core of the SPARK toolset. It can be used stand-alone and also has front ends for C and Java. Documentation is not as good as with the SPARK toolset, but students will need to spend less time getting to grips with the essentials of the input programming languages.
- For verification of concurrent software a prime candidate tool is CBMC.

Syllabus:

[Provide a more detailed description of the contents of the course, e.g. a list of bullet points roughly corresponding to the topics covered in each individual lecture/tutorial/coursework. The description should not exceed 500 words but should be detailed enough to allow a student to have a good idea of what material will be covered in the course. Please keep in mind that this needs to be flexible enough to allow for minor changes from year to year without requiring new course approval each time.]

The Course Description above already outlines the syllabus down to the lecture level.

Relevant QAA Computing Curriculum Sections:

[Please see http://www.qaa.ac.uk/en/Publications/Documents/Subject-benchmark-statement-Computing.aspx.pdf to check which section the course fits into.]

I assume this request relates to Appendix B of this document, which catalogues a Body of Knowledge.

Relevant topics in this appendix are:

- Artificial Intelligence: logics, reasoning.
- Computer-based systems: safety critical and other high-integrity systems.
- Computer hardware engineering: hardware description languages (which now include temporal assertion languages such as PSL and SVA), verification
- Simulation and modelling. (Formal verification is not simulation, but often is an alternative or supplement to simulation, and many issues are similar.)
- Software engineering: verification
- Theoretical computing: mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of programming, software specification, formal methods of system development.

Graduate Attributes, Personal and Professional skills:

[This field should be used to describe the contribution made to the development of a student's personal and professional attributes and skills as a result of studying this course – i.e. the generic and transferable skills beyond the subject of study itself. Reference in particular should be made to SCQF learning characteristics at the correct level http://www.sqa.org.uk/files_ccc/SCQF-LevelDescriptors.pdf].

(I focus here on characteristics 3-5 from Level 11 as these are most-obviously related to skills beyond the subject of study)

3: Generic Cognitive Skills

- Apply critical analysis, evaluation and synthesis to forefront issues, or issues that are informed by forefront developments in the subject/discipline/sector.
- Critically review, consolidate and extend knowledge, skills, practices and thinking in a subject/discipline/sector.
- Deal with complex issues and make informed judgements in situations in the absence of complete or consistent data/information.

4. Communication, ICT and numeracy

Use a wide range of routine skills and a range of advanced and specialised skills as appropriate to a subject/discipline/sector, for example:

- Communicate with peers, more senior colleagues and specialists.
- Use a wide range of ICT applications to support and enhance work at this level and adjust features to suit purpose.

Reading List:

*[Provide a list of relevant readings. See also remarks under **3d**.]*

- Logic in Computer Science (2nd Ed).why3 Huth and Ryan. Cambridge UP. 2004.
- NuSMV model checker documentation and tutorials. http://nusmv.fbk.eu
- SPARK toolset documentation and training materials. Altran UK. 2016. Overview at http://intelligent-systems.altran.com/technologies/software-engineering/spark.html Training materials accessed via SPARK Academic Programme.
- Why3 programme verification toolkit documentation and tutorials. http://why3.lri.fr
- CBMC (Bounded model checker for C and C++) documentation, http://www.cprover.org/cbmc/.

Breakdown of Learning and Teaching Activities:

[Total number of lecture hours and tutorial hours, with hours for coursework assignments.]

[The breakdown of learning and teaching activities should only include contact hours with the students; everything else should be accounted for in the Directed Learning and Independent Learning hours.

The total being 10 x course credits. Assume 10 weeks of lectures slots and 10 weeks of tutorials, though not all of these need to be filled with actual contact hours. As a guideline, if a 10-pt course has 20 lecture slots in principle, around 15 of these should be filled with examinable material; the rest should be used for guest lectures, revision sessions, introductions to assignments, etc. Additional categories of learning and teaching activities are available, a full list can be found at:

http://www.euclid.ed.ac.uk/Staff/Support/User_Guides/CCAM/Teaching_Learning.htm]

Lecture Hours: 17 hours (15 max on examinable material. Extras for guest lectures or lectures on material beyond syllabus.)

Seminar/Tutorial Hours: 0 hours

Supervise practical/Workshop/Studio hours: 10 hours

Summative assessment hours: 2 hours

Feedback/Feedforward hours: 4 hours

Directed Learning and Independent Learning hours: 67 hours

Total hours: 100 hours

You may also find the guidance on 'Total Contact Teaching Hours' and 'Examination & Assessment Information' at: http://www.studentsystems.ed.ac.uk/Staff/Support/User_Guides/CCAM/CCAM_Information_Captured.html

Keywords:

[A list of searchable keywords.]

Verification. Formal Verification. Hardware verification. Software verification. Model checking. Bounded model checking. Assertions. SMT (Satisfiability modulo theories). SAT (Satisfiability solvers). BDDs.

SECTION 3 - COURSE MATERIALS

3a. Sample exam question(s)

[Sample exam questions with model answers to the individual questions are required for new courses. A justification of the exam format should be provided where the suggested format non-standard. The online list of past exam papers gives an idea of what exam formats are most commonly used and which alternative formats have been http://www.inf.ed.ac.uk/teaching/exam_papers/.]

The intended exam format is one of the standard ones: 1 compulsory question and then a choice of 1 of 2 others.

The model checking material for the course is drawn from the current Automated Reasoning course. A separate document reproduces the model-checking-related questions and answers from the May 2013 exam.

3b. Sample coursework specification

[Provide a description of a possible assignment with an estimate of effort against each sub-task and a description of marking criteria.]

There is no coursework.

3c. Sample tutorial/lab sheet questions

[Provide a list of tutorial questions and answers and/or samples of lab sheets.]

For the model checking part of the course, a coursework from the current Automated Reasoning course will be adapted as a practical exercise and model answers will be made available. Please find attached the coursework instructions from 2013 as an indication of the nature of the model checking exercise. An exercise on assertion-based formal software verification might be crafted from SPARK toolset training materials and exercise on software model checking from the tutorial provided with the CBMC tool.

3d. Any other relevant materials

[Include anything else that is relevant, possibly in the form of links. If you do not want to specify a set of concrete readings for the official course descriptor, please list examples here.]

SECTION 4 - COURSE MANAGEMENT

4a. Course information and publicity

[Describe what information will be provided at the start of the academic year in which format, how and where the course will be advertised, what materials will be made available online and when they will be finalised. Please note that University and School policies require that all course information is available at the start of the academic year including all teaching materials and lecture slides.]

Full information on the course will be provided online, linked to from the course's web page.

4b. Feedback

[Provide details on feedback arrangements for the course. This includes when and how course feedback is solicited from the class and responded to, what feedback will be provided on assessment (coursework and exams) within what timeframe, and what opportunities students will be given to respond to feedback.

The University is committed to a baseline of principles regarding feedback that we have to implement at every level, these are described at http://www.docs.sasg.ed.ac.uk/AcademicServices/Policies/Feedback_Standards_Guiding_Principles.pdf.

Further guidance is available from http://www.enhancingfeedback.ed.ac.uk/staff.html.]

An online discussion forum will provide one channel for feedback on all aspects of the course, including replies to questions concerning lectures, practical exercises and previous exams.

Lectures will include feedforward introductory presentations on the practical exercises and walkthroughs and discussion of model answers to the exercises.

Demonstrators in labs will be available for providing feedback on students work on the practical exercises on a as-needed basis. In addition, students will be able to book slots, maybe 15 mins in length, during which a demonstrator can systematically walk through students' work and compare it with model solutions.

We will consider recommending students work in pairs on the exercises, so they have the benefit of peer feedback.   Students could also meet with demonstrators in these pairs, which would enable demonstrator resources to be stretched further and possibly could enable longer demonstrator meeting times.

4c. Management of teaching delivery

[Provide details on responsibilities of each course staff member, how the lecturer will recruit, train, and supervise other course staff, what forms of communication with the class will be used, how required equipment will be procured and maintained. Include information about what support will be required for this from other parties, e.g. colleagues or the Informatics Teaching Organisation.]

The demonstrator(s) and TA will likely be recruited from current PhD students in LFCS or CISA.

Communication with students will be through lectures, class emails, scheduled lab sessions and an online discussion forum.

All software used on the course is freely available and runs on Linux machines. Informatics Computing Support will be asked to have the software available on all DICE machines.

If we use the Ada/SPARK verification toolset from Altran UK, Altran UK will be happy to make available some form of their SPARK training materials that we could use or adapt to our purposes.

SECTION 5 - COMMENTS

[This section summarises comments received from relevant individuals prior to proposing the course. If you have not discussed this proposal with others please note this].

A preliminary proposal was submitted for discussion at the Board of Studies meeting on 4th November 2015. The proposal received strong support from several present.

As noted in the minutes, the proposal was approved, subject to

1. *A reduction of Learning Outcomes to maximum 5*. Originally there were 14. Now there are 4.
2. *A reduction of summative feedback to 1*. Originally there were two summative courseworks. Now there are none; all coursework is formative. This is seen as having several benefits. It eliminates the need to prepare new coursework case studies each year and enables courseworks to be improved from year to year. It also enables the provision of more detailed help to students during coursework periods and the discussion of model solutions after these periods.
3. *Compliance with new workload assessment guidelines*. The elimination of summative courseworks and the provision of at least one formative coursework fits with these guidelines. Several feedback channels are discussed in this proposal for providing timely high-quality feedback to students on their formative coursework. The teaching support estimates for TA and demonstrator time are within the normal limits set by the School's Teaching Support Staff Policy.

5a. Year Organiser Comments

[Year Organisers are responsible for maintaining the official Year Guides for every year of study, which, among other things, provide guidance on available course choices and specialist areas. The Year Organisers of all years for which the course will be offered should be consulted on the appropriateness and relevance on the course. Issues to consider here include balance of course offerings across semesters, subject areas, and credit levels, timetabling implications, fit into the administrative structures used in delivering that year.]

Year 4 Organiser (Mary Cryan)

Comments being sought

Year 5 Organiser (Mahesh Marina)

Comments being sought

Taught MSc Year Organiser (Paul Jackson)

This course would be a welcome addition to the portfolio of optional courses available to the MSc specialisms cited in Section 2c above.

Currently, these specialisms have slightly more options in Semester 2, so adding this course to Semester 1 would improve the balance. However, as noted in Section 2c, this course complements the proposed new AR course and it might be worth scheduling them in different semesters. If AR is put in Semester 1 (replacing the current AR course in Semester 2), then adding FV to Semester 2 would be fine.

5b. BoS Academic Secretary

[Any proposal has to be checked by the Secretary of the Board of Studies prior to discussion at the actual Board meeting. This is a placeholder for their comments, mainly on the formal quality of the content provided above.]