PROPOSED COURSE TITLE: Automated Reasoning

PROPOSER(S): Jacques Fleuriot

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what motivates the course proposal, e.g. an emergent or maturing research area, a previous course having become outdated or inappropriate in other ways, novel research activity or newly acquired expertise in the School, offerings of our competitors.)

The current Automated Reasoning course is split into two distinct parts involving interactive theorem proving and model checking. While the students do manage to get hands-on experience with state-of-the-art tools such as Isabelle and NuSMV, the course is delivered at a fast pace and there are numerous topics that cannot be covered in much depth. This is especially problematic for the theorem proving part, where the first 5 lectures have to cover introductory material about propositional and first order logic to ensure that all students have the appropriate background. This leaves very little room when it comes to providing the students with an understanding of topics such as rewriting, higher-order unification, proof style (e.g. procedural vs declarative proofs) and the interplay between interactive and fully automatic theorem provers, which are important aspects of modern automated reasoning.

Another issue with the current AR course is that it is delivered as a 4th year course, which I believe has led to a reduction in the number of third year students selecting theorem proving/AR-related individual projects.

The aim of this proposal is to revamp the Automated Reasoning course to ensure a less-hurried delivery that

i) will enable appropriate motivation and coverage of existing topics and
ii) the introduction of new ones that provide a thorough understanding of modern, interactive theorem proving.

The model checking part would then move to a more specialised, Level 11 Formal Verification (FV) course, which is being proposed by Paul Jackson.

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 main audience will be UG3 students, but MSc students, who have an interest in formal reasoning and are planning to take the new Formal Verification course, would also benefit.

Currently the AR course attracts around 30 students, but with the proposed change I expect the number to increase to 40-50, if we include MSc students. This estimate is based on what the class size used to be when AR was a UG3 course. However, given that Informatics now attracts more MSc students, this figure should probably be viewed conservatively.

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.]

This proposal revamps the Automated Reasoning course, setting it as a Year 3 course (as it used to be in the past) to broaden the choice available to students when it comes to projects in the area. AR will also benefit from the foundations laid by courses such as Inf2D and DMMR, thereby ensuring that the time spent revisiting topics such as propositional and first order logic is kept to a minimum.

The course is best offered as a Semester 1 course to enable UG3 students to familiarise themselves with the material before they make their UG4 individual project selection. Moreover, it may be of interest to MSc students who plan to take the FV course.

Relation to existing curriculum: See 2c. Aspects such as formalized mathematics fit well with other topics in joint degrees involving maths.

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.]
Lecturer: Jacques Fleuriot (potentially shared with Paul Jackson as is the case now, with Fleuriot then teaching half of the new FV course.)

Teaching assistant (as for current AR). The TA will set the practical assignment, working in conjunction with the course lecturer(s). The assignment will require the use of the interactive theorem prover Isabelle.

Demonstrator (as for current AR). The demonstrator will be available to help in the lab with the practical assignment. The demonstrator will be expected to answer specific queries about the coursework and the system being used, as well as more general queries about logic and formal reasoning. They are also expected to reply to email queries that students may have.

Marker (as for current AR). The marker will be responsible for going through the submissions for the assignment, in order to give the appropriate feedback and marks. Marking activities will include inspecting the submitted files and checking them using the relevant automated reasoning tool. The marker will also work out a marking guide with the lecturer beforehand and may be expected to provide notes on answers for the coursework.

Tutors. These will be required for the tutorials. They will be responsible for going through the unassessed questions, which will be provided to the students at least one week in advance of the actual tutorial. Answers will be provided to the tutors ahead of time and posted on the course web page at the end of each week.

Scalability. The main limiting factor is the size of a computing lab but I don’t expect this to be an issue if the number of students is under 50. Most students tend to work on their own laptops and Isabelle is freely available under Linux, OS X and Windows so DICE is not an absolute necessity.
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.]:

Automated Reasoning

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 enrolls 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.

9

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.

3

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.

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.]

Artificial Intelligence

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.]

Important: Any degree with Artificial Intelligence in its title and for MInf.
Compulsory: AI and Maths, CS and Maths.

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).]

Semester 1.

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.]

The overall aim of the course is to describe how reasoning can be modelled using computers. Its more specific aim is to provide a route into more advanced uses of theorem proving in order to solve problems in mathematics and formal verification.

Major emphases are on: how knowledge can be represented using propositional, first-order and higher-order logic; how these representations can be used as the basis for reasoning, and how these reasoning processes can be guided to a successful conclusion through a variety of means ranging from fully-automated to interactive ones. Students will develop a thorough understanding of modern, interactive theorem proving via lectures, tutorials and an assignment.
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].

The course starts with an introduction to higher order logic, theorem provers and, more specifically, Isabelle/HOL. This will set the context for the rest of the course in which Isabelle will be the framework for getting hands-on experience about the application of various theoretical concepts. Through the lectures and tutorials that incorporate practical exercises the students will gain the skills needed to get started with Isabelle and progress to more complex concepts involving both representation and reasoning.

The second part will look at representation/modelling of concepts in (higher order) logic in details. Axiomatic versus conservative extensions of theories will be covered and mechanisms such as Isabelle locales will be introduced and used. Recursive definitions and inductive notions will be covered too.

The third part of the course will focus on fundamental notions such as unification and rewriting, within both a first and higher order context. It will look at notions such as termination and use Isabelle’s simplifier as the tool for understanding many of the concepts. It will also look at the interplay between (fully) automatic and interactive proofs.

The fourth part will introduce declarative/structured proofs and using the Isar language of Isabelle show how proofs resembling pencil-and-paper ones can be formalized.

Finally the various strands will brought together through the discussion of a non-trivial case study. This may involve either formalized mathematics (e.g. looking at a geometric theory) or a formal verification example.

The assignment will be a combination of basic to intermediate representation and reasoning in Isabelle (up to 40%), more advanced proof tackling one particular domain or example (up to 40%) and a final part which, if completed successfully, will clearly demonstrate that the student has a good of the challenges that advanced interactive theorem proving entails.

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].

Informatics 2D – Reasoning and Agents (INFR08010)
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].

None

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.]


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. Use sophisticated mechanisms available in theorem provers to represent problem.
2. Write interactive proof in procedural and declarative styles.
3. Use interactive and automated methods to carry out proofs in the theorem prover.
4. Represent and reason about mathematical and other less formal knowledge using logic.
5. Understand and compare automated reasoning techniques and apply them using pen-and-paper.

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].

The course will consist of 1 practical exercise (40%), assessing learning objectives 1 to 3. Students may be asked to represent and reason about particular domains e.g. geometry or inductive proofs in the Isabelle theorem prover. There will be a formative part to the assessment that will involve the students carrying out Natural Deduction proofs in Isabelle and receiving early feedback on their effort during tutorials. For the summative part, the students will submit their files, usually in the form of mechanized theories, electronically for marking.

The examination (60%) will concentrate on assessing learning outcomes 4 and 5, which mainly involve theoretical aspects (e.g. important algorithms and techniques), representation issues, problem solving (e.g. proofs using natural deduction) and discussing broader aspects such as the capabilities and limitations of various proof techniques.

Assessment Weightings:

Written Examination: 60 %
Practical Examination: 0%
Coursework: 40%
Time spent 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 assignment will involve practical work using an interactive theorem and will form a core component of the syllabus since the course is concerned with the design, use and underlying concepts of such tools. It usually takes a fair amount of time for students to pick up enough about theorem proving so that they can accomplish something useful and worthwhile in the assignment hence the proposed coursework weighting (which would be the same as the current AR course).

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.]

See the Course Description.

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.]

- Introduction to higher order logic and Isabelle/HOL.
- Natural Deduction.
- Representation/Modelling in logic.
- Procedural proofs in Isabelle.
- First order and higher order unification.
- Rewriting (simplification).
- Proof automation via automatic theorem proving.
- Representing and reasoning about recursive datatypes and functions.
- Inductive definitions and proofs.
- Declarative/structured proofs in Isabelle.
- Modelling Case Study: Formalized mathematics or verification.
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.]

| Artificial Intelligence |

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 ].

| Broken link. |

Reading List: See 3d.

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

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: 16 hours
Seminar/Tutorial Hours: 8 hours (weekly)
Supervise practical/Workshop/Studio hours: 0 hours
Summative assessment hours: 2 hours
Feedback/Feedforward hours: 1 hour
Directed Learning and Independent Learning hours: 73 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.]

Automated Reasoning  
Theorem Proving  
Formal proof

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 exam questions will be similar in style to those asked for Part 2 of the current
Automated Reasoning paper. This will be a mixture of theory and actual proofs/logical
computations.

See 2014, Question 2, for example. This covers aspects such as Natural Deduction proofs,
rewriting and unification.
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.]

Part 1: Natural Deduction [25%]

In this section, you will get some practice with natural deduction by proving some theorems from propositional and first-order logic. Each of these theorems could be solved directly with Isabelle’s automatic tactics, but here, you are asked to use only the basic introduction and elimination rules.

<table>
<thead>
<tr>
<th>Lemma</th>
<th>Description</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemma shows $P \rightarrow P$</td>
<td>oops</td>
<td>(1 mark)</td>
</tr>
<tr>
<td>lemma shows $P \land Q \rightarrow Q \land P$</td>
<td>oops</td>
<td>(1 mark)</td>
</tr>
<tr>
<td>lemma shows $P \lor Q \rightarrow Q \lor P$</td>
<td>oops</td>
<td>(1 mark)</td>
</tr>
<tr>
<td>lemma shows $(Q \land R) \land P \rightarrow (P \land R) \land Q$</td>
<td>oops</td>
<td>(2 marks)</td>
</tr>
<tr>
<td>lemma shows $A \rightarrow (B \land C) \land D \rightarrow E \rightarrow F \rightarrow D \land A \land C$</td>
<td>oops</td>
<td>(2 marks)</td>
</tr>
<tr>
<td>lemma shows $(A \rightarrow B) \land (C \rightarrow D) \land (A \lor C) \rightarrow B \lor D$</td>
<td>oops</td>
<td>(2 marks)</td>
</tr>
<tr>
<td>lemma shows $(Q \lor R) \land P \rightarrow \neg P \rightarrow Q$</td>
<td>oops</td>
<td>(3 marks)</td>
</tr>
<tr>
<td>lemma shows $(\neg P \rightarrow Q) \rightarrow (\neg P \rightarrow \neg Q) \rightarrow P$</td>
<td>oops</td>
<td>(3 marks)</td>
</tr>
<tr>
<td>lemma shows $\neg(P \land Q) \rightarrow \neg P \lor \neg Q$</td>
<td>oops</td>
<td>(4 marks)</td>
</tr>
</tbody>
</table>

This is a relatively straightforward part to ensure that all the students can do some basic to intermediate level of reasoning in Isabelle.

Marking Criteria: Complete proofs are given for the lemmas. No use of any automatic tools and ensuring that the appropriate tactics/methods are used with the various Natural Deduction rules.
Part 2: Formalizing a geometry of simple curves [55%]

In some recent work, Kulik and Eschenbach present a formal axiomatic framework dealing with oriented curves [1]. This work introduces points (denoted by $P$, $Q$, $P'$, ...), curves (denoted by $c$, $c_1$, ...), and oriented curves (denoted by $o$, $o_1$, ...) as primitive geometric entities (structures) and two primitive relations, namely, incidence and precedence. A number of basic definitions and axioms are then given to characterize the relationships between these various primitive entities.

In this assignment, your task is to formalize part of this axiomatic framework (which only involves simple, non-oriented curves and the incidence relation) and mechanically prove a number of theorems as given in paper [1]. Your work will thus provide a rigorous, mechanical verification of some of the claims made by Kulik and Eschenbach.

Mechanizing the Basic Definitions of Simple Curves [15%]

To help you formalise the framework in Isabelle, the two primitive objects have been declared as new types:

```plaintext
typedec pt
typedec curve
```

Note that these types are not defined, so nothing is known about them except that they are nonempty. A binary incidence relation has also been declared for you:

```plaintext
consts incident :: "[pt, curve] ⇒ bool"
```

The predicate takes two arguments and represents the notion of a point lying on a curve. It has been declared as an infix predicate, so you can express that a point $P$ lies on the curve $c$ by $P$ incident $c$. In the template file msexprical.thy, you have been provided with the declared, but not yet defined, predicates isPartOf (representing the $\sqsubseteq$ given by Kulik and Eschenbach [1]), isEndPoint, isInnerPoint, MeetAt, isSumOf, isClosedCurve and isOpenCurve. These are slightly more readable names – using infix notations whenever possible – for the predicates used by Kulik and Eschenbach (e.g. we can specify that a point $P$ is an inner point of a curve $c$ as $P$ isInnerPoint $c$ instead of $\text{ipt}(P,c)$).

Note that the sum operation is defined using an $\Rightarrow$ symbol by Kulik and Eschenbach. Although this is a convenient notational device for the paper, this is not advisable in our formalization as (among many other issues) the $=$ is already defined in Isabelle. So, we explicitly define a new relation isSumOf such that $c$ is the sum of $c_1$ and $c_2$ ($c_1 \sqcup c_2$) is denoted by $c$ isSumOf $c_1$ $c_2$. Note also that one often has to make such representational choices (e.g. relational vs. functional) when dealing with the mechanization of a (pen-and-paper) framework.
Your tasks are to:

1. Formalize Definitions (2.1)-(2.4) from Kulik and Eschenbach’s paper [1] in Isabelle.

\[ \text{Mechanizing the Axioms for Simple Curve Geometry [10%]} \]

\[ \text{Mechanizing the Basic Consequences of the Axioms [30%]} \]

In Part 2, the students are asked to work on a more demanding formalization that goes beyond Natural Deduction. It could be a formalized mathematics task (as above) or a formal verification one (e.g. about properties of some algorithm). They will be given enough background information to carry out the task e.g. various definitions and even lemmas will usually be given to them in a theory file. They will not be expected to do any formalization from scratch.

Marking Criteria: Correct and appropriate representations of various concepts. Completed proofs, with credit for partial proofs.

**Part 3: Challenge Proof: Theorem 2.13 [20%]**

In this part, you should:

1. Attempt a mechanical proof of Theorem 2.13, which states that:

   Two distinct points on an open curve uniquely determine the sub-curve connecting these points.

Note that this part of the assignment is challenging and may involve a lot of effort without a proper mechanization plan. You are strongly advised to follow the proof outlined in the paper. In particular, you should consider Step 1 and Step 2 separately. You may find it helpful to prove various sub-lemmas.

Note also that credit will be given for an unsuccessful or incomplete mechanization attempt that proved significant lemmas/theorems demonstrating progress towards a final proof.

There will usually be a final, challenging part to the assignment that aims to really test the students and indicate whether they have understood some of the intricacies and subtleties of interactive theorem proving. This part may come with an existing pen-and-paper proof (as for the above) or may involve working out the proof from scratch.

Note that the last part will usually account for no more than 20% of the overall marks.
3c. Sample tutorial/lab sheet questions

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

This is a sample exercise (theory file) that involves the use of Isabelle. The exercise would then be discussed in the tutorial to ensure the students understand the various points being made.

theory ex5 imports Main begin

text {* This exercise shows an example of how a bad axiomatization can destroy Isabelle's guarantee of soundness. We will add to Isabelle's existing theory the axioms of Naive Set Theory. *}

text {* First, we declare the existence of a new type 'SET', which represents the sets of our Naive Set Theory. We do this in Isabelle with the following declaration. *}
typedcl SET
text {* Now we declare the membership predicate and the axiom. The 'axiomatization' declaration first lists the new function symbols we are adding, and then lists the axioms. *}

Here, we state 'comprehension' as a single axiom, but really it is an axiom schema, representing infinitely many axioms, one for every 'P'. *}

axiomatization Member :: "SET => SET => bool"
where
  comprehension : "\exists y. \forall x. Member x y \longleftrightarrow P x"

text {* You can use the axiom comprehension in a proof by using the command: *}

apply (cut_tac P="\lambda x. XXXX" in comprehension)

where XXXX is the predicate (with free variable 'x') that you want to instantiate the axiom with. *}

text {* You can now prove the paradoxical statement: *}

lemma member_iff_not_member : "\exists y. Member y y \longleftrightarrow \neg Member y y"

(* Add your "proof" here *)
oops
text {* Using the lemma, it is now possible to "prove" 'False', showing that we have definitely introduced an inconsistency. You will have to use the rule 'excluded_middle' or one of the other axioms in order to complete the proof. It may be easier to try to work out how to prove this theorem on paper first, before attempting it on the computer. *}

(Note: Isabelle's built-in QuickCheck will attempt to show you a counterexample to this theorem, because it 'knows' that False is not provable. The QuickCheck facility doesn't take into account any axioms that have been added.) *}

theorem inconsistency : False

(* Add your "proof" here *)
oops
text {* After we have proved False, we can prove all sorts of nonsense: *}

theorem "1 = 0"

(* Use your theorem "inconsistency" above to show this *)
oops

end
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.]

Recommended reading list:


The students will also be asked to read various papers and given links to presentations and websites with materials pertaining to various theorem proving projects and repositories (e.g. The Archive of Formal Proof).
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.]

Course description. Material from previous version of AR course, together with an explanation of how the new course differs.

All slides and tutorials will be available at the beginning academic year. The assignment will be available no later than 2 weeks from the start of the course.

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.]

The students will receive feedback on their weekly tasks during the tutorials. The tasks will include both pen-and-paper and Isabelle-based exercises. Full solutions will be made available on the AR website at the end of each week.

A marking guide, with extensive notes, will be released after the marks for the assignment have been returned (as is currently the case). This will provide a breakdown of how marks were allocated, especially for the more challenging parts of the formalization. Common pitfalls will be highlighted and advice provided on how these should have been tackled.

The automated reasoning mailing list will be used (as is currently the case) to provide general feedback and any advice deemed appropriate during the course and before the exams.
**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.]

<table>
<thead>
<tr>
<th>Role</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecturer</td>
<td>Deliver the lectures.</td>
</tr>
<tr>
<td>TA</td>
<td>Help write the assignment under the close guidance of the lecturer.</td>
</tr>
<tr>
<td>Demonstrator</td>
<td>Available in the lab for 2 hours a week to help students who may be having difficulties or may need advice on Isabelle and/or the assignment.</td>
</tr>
<tr>
<td>Marker</td>
<td>Mark the formative assignment.</td>
</tr>
<tr>
<td>TA, Demonstrator, and Marker</td>
<td>Recruited by the standard Informatics mechanisms. The TA will be given relevant assignments, solutions and marking guides from previous years (or versions of AR).</td>
</tr>
<tr>
<td>Note</td>
<td>Expect the TA, Demonstrator and Marker roles to be held by only one person in many cases.</td>
</tr>
<tr>
<td>Equipment</td>
<td>Isabelle under DICE and it will need to be updated every year (as is currently the case). Expect many students will install Isabelle on their personal computers.</td>
</tr>
</tbody>
</table>
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].

I have discussed this revised course extensively with Paul Jackson and Alan Smaill and many of their comments have informed the current proposal.

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.]

"[This] looks reasonable to me.

I don't see the semester listed -- for course balance sake, I would recommend semester 1."


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.]