MATH43011– Computation and Complexity (Mark Kambites)

Note: This course last ran in semester 2 of 2014/15. The Level 4 and 6 versions of the course are identical (and the exam will be the same).

Unit code:	MATH43011
Credit Rating:	15
Unit level:	Level 4
Teaching period(s):	Semester 1
Offered by	School of Mathematics
Available as a free choice unit?:	N

Requisites: A general mathematics background. Basic familiarity with propositional logic and/or graph theory may be a slight advantage since these are used for some of the examples, but this is not essential as all required concepts will be covered. The course does not contain any practical computing.

Additional Requirements: Students are not permitted to take MATH43011 and MATH63011 for credit in an undergraduate programme and then a postgraduate programme.

Aims:

- to introduce the main model of computation currently being employed in the theory of computation, Turing machines;
- to introduce the key parameters quantifying computational complexity (deterministic, non-deterministic, time, space) and the relationships between them.

Overview: Quite a lot of the mathematics you have studied so far involves using algorithms to solve computational problems. For example, you have probably used Euclid's algorithm to solve the problem of finding the greatest common divisor of two integers. In this course, we abstract a level further, and study the properties of problems and algorithms themselves. The kind of questions we ask are "is there an algorithm to solve EVERY problem?" and "what problems can be solved by an EFFICIENT algorithm?".

Compared with most of mathematics, this area is in its infancy, and many important things remain unknown. The course will take you to the point where you understand the statement of one of the most important open questions in mathematics and computer science: the "P vs NP" problem, for which the Clay Mathematics Foundation is offering a $1,000,000 prize. And who knows, perhaps one day you will be the one to solve it!

Learning outcomes: On successfully completing the course students will be able to:

- define Turing machines, discuss their capabilities and limitations, and construct and analyse simple examples;
- define the key concepts of computation and complexity (including the main computability and complexity classes), discuss the relationships between them, and prove simple seen and unseen facts about them;
- recall and analyse a range of computational problems and their properties;
- state, apply and prove some of the main theorems of computation and complexity theory;
- classify and compare the computability and complexity of decision problems in simple cases.

Assessment methods: Coursework - 20%; Written exam - 80%.
Assessment Further Information: Coursework: Two pieces of take-home coursework (weighting within unit, 10% each). End of semester examination: duration 3 hours (weighting within unit, 80%).

Syllabus:
0. INTRODUCTION (approx 1 hour): outline introduction to computability and complexity; course practicalities.

1. COMPUTABILITY (approx 9 hours): problems and solutions; alphabets and languages; Turing machines; recursiveness and the Church-Turing Thesis; multitape machines; coding machines and non-recursive languages; universal computation; non-determinism.

2. COMPUTATIONAL COMPLEXITY (approx 7 hours): time and space; linear speed up and space reduction; complexity classes; lower bounds and crossing arguments; space and time hierarchy theorems; tractability and P vs NP; polynomial time reduction.

3. COMPLETENESS (approx 8 hours): NP-completeness; SAT and the Cook-Levin Theorem; NP-completeness by reduction; further examples of NP-complete languages; NP-intermediacy and Ladner's Theorem; PSpace-completeness; oracles and the Baker-Gill-Solovay Theorem.

4. SPACE COMPLEXITY (approx 3 hours): Savitch's Theorem; the Immerman-Szelepcsenyi Theorem.

Recommended reading: The course notes are self-contained and you should not need to refer to any books. But if you would like an alternative viewpoint, the following texts cover most of the course material:

Bovet and Crescenzi, *Introduction to the Theory of Complexity*, 1994

Papadimitriou, *Computational Complexity*, 1994.

Sipser, *Introduction to the Theory of Computation* (second edition), 2006.

Feedback methods: Feedback tutorials will provide an opportunity for students' work to be discussed and provide feedback on their understanding. Coursework or in-class tests (where applicable) also provide an opportunity for students to receive feedback. Students can also get feedback on their understanding directly from the lecturer, for example during the lecturer's office hour.

Study hours:
Lectures - 28 hours
Tutorials - 5 hours
Independent study hours - 117 hours

Teaching staff: Professor Mark Kambites - Unit coordinator