COMP30025 Theory of Computation

Credit Points:	12.50
Level:	3 (Undergraduate)
Dates & Locations:	This subject is not offered in 2014.
Time Commitment:	Contact Hours: 48 hours, comprising of two 1-hour lectures and one 2-hour workshop per week Total Time Commitment: 170 hours
Prerequisites:	25 points of university-level mathematics.
Corequisites:	None
Recommended Background Knowledge:	Basic proficiency in discrete mathematics and propositional logic.
Non Allowed Subjects:	**Subject** / **Study Period Commencement:** / **Credit Points:** COMP30021 Theoretical Computer Science / Not offered 2014 / 12.50 433-330 Theory of Computation
Core Participation Requirements:	<p>For the purposes of considering request for Reasonable Adjustments under the Disability Standards for Education (Cwth 2005), and Student Support and Engagement Policy, academic requirements for this subject are articulated in the Subject Overview, Learning Outcomes, Assessment and Generic Skills sections of this entry.</p> <p>It is University policy to take all reasonable steps to minimise the impact of disability upon academic study, and reasonable adjustments will be made to enhance a student's participation in the University's programs. Students who feel their disability may impact on meeting the requirements of this subject are encouraged to discuss this matter with a Faculty Student Adviser and Student Equity and Disability Support: <a href="http://services.unimelb.edu.au/disability">http://services.unimelb.edu.au/disability</a></p>
Contact:	email: **awirth@unimelb.edu.au (mailto:awirth@unimelb.edu.au)**
Subject Overview:	**AIMS** At the heart of theoretical computer science are questions of both philosophical and practical importance. What does it mean for a problem to be solvable by computer? What are the limits of computability? Which types of problems can be solved efficiently? What are our options in the face of intractability? This subject covers such questions in the content of a wide-ranging exploration of the nexus between logic, complexity and algorithms, and examines many important (and sometimes surprising) results about the nature of computing. **INDICATIVE CONTENT** # Automata theory: Finite-state machines, pushdown automata, grammars and recognisers for regular and context-free languages # Computability: Turing machines, the Church-Turing thesis, decidability, reducibility # Complexity theory: The classes P and NP, NP-complete problems, space complexity including sub-linear space, approximation algorithms, probabilistic complexity classes, use in cryptography # Additional topics may include descriptive complexity and interactive proof Example of assignment # Proving the equivalence of a variant of a standard machine to the original version # Demonstrating the regularity, or otherwise, of a language # Describing an NP-hardness reduction

Learning Outcomes:	**INTENDED LEARNING OUTCOMES (ILO)** On completion of this subject the student is expected to: 1 Design, manipulate, and reason about formal computational models, such as automata and Turing machines 2 Describe the limitations of different types of computing devices 3 Identify relations between classes of computational problems, formal languages, and computational models 4 Account for the inherent complexity of many computational problems of practical importance 5 Conduct formal reasoning about machines, problems and algorithms, including reduction-based proof
Assessment:	Two individual assignments involving mathematical proof and possibly some programming, due in approximately weeks 6 and 10 Written assignments during semester, expected to take about 36 hours (30%) A 3-hour end-of-semester written examination (70%) Hurdle requirement: To pass the subject, students must obtain at least: 50% overall 15/30 in project work And 35/70 in the written examination Assessment addresses all Intended Learning Outcomes (ILOs)
Prescribed Texts:	Michael Sipser, "Introduction to the Theory of Computation", 3rd Edition.
Breadth Options:	This subject potentially can be taken as a breadth subject component for the following courses: # **Bachelor of Arts (https://handbook.unimelb.edu.au/view/2014/B-ARTS)** # **Bachelor of Commerce (https://handbook.unimelb.edu.au/view/2014/B-COM)** # **Bachelor of Music (https://handbook.unimelb.edu.au/view/2014/B-MUS)** You should visit **learn more about breadth subjects (http://breadth.unimelb.edu.au/breadth/info/index.html)** and read the breadth requirements for your degree, and should discuss your choice with your student adviser, before deciding on your subjects.
Fees Information:	Subject EFTSL, Level, Discipline & Census Date, http://enrolment.unimelb.edu.au/fees
Generic Skills:	On completion of this subject students should have developed the following skills: # Analytical skills # Reasoning and problem solving skills # Ability to apply knowledge of basic science and engineering fundamentals # Capacity for creativity and innovation # Ability to undertake problem identification, formulation and solution # Capacity for lifelong learning and professional development # Communication skills, in particular, ability to communicate precise formal arguments and proofs
Notes:	**LEARNING AND TEACHING METHODS** The subject involves two 1-hour lectures per week followed by a 2-hour workshop. Weekly tutorial problems are assigned. Both lectures and workshops are designed to be highly interactive, and the written assignments are designed to be challenging, so as to generate discussion. Although written assignments are submitted by students individually, in-plenum discussion of the problems is allowed, and encouraged. **INDICATIVE KEY LEARNING RESOURCES** The subject uses a state-of-the-art textbook. It offers access to visualisation tools (the JFLAP suite), an online discussion forum, and advance access to all teaching materials, including slides used in lectures. **CAREERS / INDUSTRY LINKS** A solid understanding of computability and complexity is essential for any research-oriented work in the computing industry and academia. A grounding in theory provides important

	conceptual tools that are used by theoreticians, computer scientists, and software engineering practitioners alike.
Related Majors/Minors/ Specialisations:	Computing and Software Systems Master of Engineering (Software with Business)