Course Outlines

Year 1

FUNCTIONAL PROGRAMMING

This is a first course in programming. You will use of a programming language called Haskell, which allows programs to be viewed as mathematical functions. This makes the language very powerful, so that we can easily construct programs that would be difficult or very large in other languages. An important theme of the course is how to apply mathematical reasoning to programs, so as to prove that a program performs its task correctly, or to derive it by algebraic manipulation from a simpler but less efficient program for the same problem. You gain hands-on experience of programming through two lab exercises: the first one aims to make you acquainted with the mechanics of writing Haskell programs, and the second one tackles a more challenging programming task of simulating the world of an imaginary animal. This demonstration shows you more about this problem.

DESIGN AND ANALYSIS OF ALGORITHMS

In this course you will learn about the basics of algorithms, the methods of solving problems by computer, and data structures - the means by which information is represented and manipulated inside a computer. Building on the experience with Haskell from Functional Programming, the course covers the principles of algorithm design, analysis of the performance of algorithms, and the fundamental ideas in the design of efficient data structures. The emphasis is on choosing appropriate data structures for a problem domain and designing correct and effective algorithms to operate on these data structures in solving the problem. You will learn to apply these ideas in practice through a lab exercise.

IMPERATIVE PROGRAMMING
In these courses you will apply lessons you learnt in Functional Programming to the design of programs written in a more conventional way. By studying a sequence of programming examples, each a useful software tool in its own right, you will learn to construct programs in a systematic way, structuring them as a collection of modules with well-defined interfaces. These courses also cover informally the method of invariants for understanding and reasoning about programs that contain loops. You will conclude with the study of a larger programming example, such as a graphical program for finding the best route for driving between specified towns in the UK. You can further explore this problem using this demonstration. Through lab exercises, you will learn to create, debug and maintain programs of a non-trivial but moderate size.

The second part of the course introduces the ideas of Object Oriented Programming: abstract data types and data encapsulation; interfaces and polymorphism; and generics and collection classes. In lab sessions, you will work to develop and enhance your own text editor, supporting incremental search and multi-level undo.

DIGITAL SYSTEMS

On this course you will learn about the fundamentals of the electronic circuits that are used to build computers. Beginning with the functions of individual transistors, and building towards a complete implementation of a simple processor, the course explains in a structured way how complex behaviours are built up from simpler parts. You will gain an understanding of the factors that affect the performance of hardware, and how these factors alter with changes of scale, for example in the size of the data that a computer system handles.

LINEAR ALGEBRA

Another important application of computers in Mathematics is in solving problems that can be expressed as systems of simultaneous linear equations. You will have learnt at school how to solve particular sets of simultaneous equations, but this course goes further. You will study the general conditions that determine which systems of equations have solutions, and for which systems the solution is unique, as well as systematic methods for solving systems of equations. This demonstration which you might program during your 2nd year utilizes the mathematics you will learn in this course along with the 2nd year numerical computation course.

CONTINUOUS MATHEMATICS

This will be taught by the Department of Computer Science. Many areas of mathematics, and applications of mathematics in the applied sciences, are underpinned by the concepts of calculus, i.e. differentiation and integration. The first aim of this course is to provide an introduction to calculus in several variables that will form a basis for later courses. This theory will be demonstrated using examples drawn from maximising or minimising functions of one or more variables, the solution of ordinary differential equations, and the solution of simple partial differential equations (including the reduction of partial differential equations to ordinary differential equations via a change of variables in special cases). The second aim of the course is to introduce some computational techniques in calculus, for example numerical integration and the numerical solution of differential equations. These techniques lend themselves to practical implementation, allowing demonstration of the theory developed during the course.

DISCRETE MATHEMATICS

This course will provide you with the vocabulary of concepts that is needed to understand in mathematical terms the problems that computer systems are designed to solve, and to analyse proposed solutions for their correctness and efficiency. Specifications of computer systems can be expressed formally in terms of sets, relations and functions, and the correctness of programs that implement these specifications is often proved using the techniques of mathematical logic, including reasoning based on mathematical induction. This reasoning may involve the mathematical properties of permutations, orderings, and other similar concepts. Finally, analysis of the efficiency of programs may involve solution of recurrence relations, combinatorial calculations, and the use of asymptotic approximations. All these ideas will be introduced to you in this course.

INTRODUCTION TO PROOF SYSTEMS

The course is an introduction to logic and proof systems. You will develop skills of using formal logic to express and prove useful properties of mathematical structures and more generally in constructing rigorous proofs and manipulating formal notation. You will be exposed to basic logic topics, including proof systems, resolution, satisfiability, compactness, Skolemisation, and Herbrand models.

PROBABILITY

This course introduces some concepts from probability theory. You will learn about random processes, and learn to understand and apply probabilistic models such as the Binomial and Poisson distributions. Probability is useful in Computer Science not only to answer questions about the average performance that can be expected of computer systems, but also in advanced topics like Randomised Algorithms, in which some otherwise 'hard' problems can be solved faster using a program that makes random choices.

EXAMINATIONS IN COMPUTER SCIENCE

The Computer Science degree has examinations at the end of each year. Those in the second and third years count towards the final degree result.
The first year exams consist of four papers on Functional Programming and Design and Analysis of Algorithms; Imperative Programming; Discrete Mathematics, Probability, and Continuous Mathematics; and Digital Systems, Linear Algebra and Introduction to Proof Systems.
The second year exams consist of eight 2 hour papers, made up of four Computer Science options papers and four core subjects: Concurrent Programming,Algorithms and Data Structures, Algorithms, Compilers and Models of Computation.
The third year exams consist of six papers of 2 hours each on Computer Science options, making an equivalent of five 3-hour papers. There is an optional project in year three.
The fourth year of the Computer Science course, leading to a Masters qualification, is mostly examined by take-home papers completed during the vacations. Candidates also submit a report on their project.
There is a practical element to the examinations in that the examiners receive reports on the practical work done during the year. Candidates are expected to have attended practical sessions and done the required work in order to pass the exam, but this work does not contribute to the degree class that is awarded.

Year 2
MODELS OF COMPUTATION
On this course you will gain a basic understanding of the classical mathematical models used to analyse computing processes, including finite automata, grammars, and Turing machines. These mathematical models can be used to answer questions such as what problems can be solved by computer, and whether there some problems that are intrinsically harder to solve than others.

CONCURRENT PROGRAMMING

We want to make computer software run faster and we want to improve our ability to use the facilities provided by modern multi-core processors and multi-processor supercomputers to do this. We also want to be able to build reliable distributed systems -- systems that necessarily use more than a single computer working alone. The principal aim of this course is to begin to present some of the big programming challenges that arise naturally from these requirements and to begin to introduce practical techniques for tackling them. Happily it also turns out that many of the techniques (such as message-passing) that arise naturally when thinking about these challenges can be applied profitably to the design of programs that may only ever be run on one, single-core, processor. In this case the payoff is in the improvement in conceptual clarity and maintainability of the designs.

COMPILERS
This course aims to give a simple but practical account of the programming techniques used in implementing high-level programming languages by compiling into code for stack and register-based machines. The course is based on a working implementation, written in Objective Caml, of a compiler for a language comparable to C or Pascal.

ALGORITHMS AND DATA STRUCTURES

This course presents a number of highly efficient algorithms and data structures for fundamental computational problems across a variety of areas, including amortised complexity analysis, maximum flows and applications, linear programming, approximation algorithms, and fixed parameter algorithms.

ARTIFICIAL INTELLIGENCE
This is an introductory course into the field of artificial intelligence (AI), with particular focus on search as the fundamental technique for solving AI problems.
Combinatorial Optimisation
This course is an introduction to combinatorial optimisation (with focus on algorithms). It will introduce ideas and techniques underlying many standard algorithms taught in introductory first-year and second-year algorithms courses and go beyond.
Computational Complexity
This course is an introduction to the theory of computational complexity and standard complexity classes. One of the most important insights to have emerged from Theoretical Computer Science is that computational problems can be classified according to how difficult they are to solve. This classification has shown that many computational problems are impossible to solve, and many more are impractical to solve in a reasonable amount of time. To classify problems in this way, one needs a rigorous model of computation, and a means of comparing problems of different kinds. We introduce these ideas in this course, and show how they can be used.

COMPUTER-AIDED FORMAL VERIFICATION

Computer-aided formal verification aims to improve the quality of digital systems by using logical reasoning, supported by software tools, to analyse their designs. The idea is to build a mathematical model of a system and then try to prove properties of it that validate the system's correctness - or at least help discover subtle bugs. The proofs can be millions of lines long, so specially-designed computer algorithms are used to search for and check them.

This course provides a survey of several major software-assisted verification methods, covering both theory and practical applications. The aim is to familiarise students with the mathematical principles behind current verification technologies and give them an appreciation of how these technologies are used in industrial system design today.

COMPUTER ARCHITECTURE

This course follows on from Digital Systems by showing you how hardware components can be used to design processors that achieve high performance. Central to the course is the design of pipelined architectures that execute multiple instructions simultaneously, detecting and resolving interactions between instructions dynamically. The course covers the design of instruction sets and different styles of processor implementation, followed by a brief survey of parallel machines that achieve still higher performance.

COMPUTER GRAPHICS

This is an introductory course in Computer Graphics, and covers a wide range of the field of interactive computer graphics at all levels of abstraction, and with emphasis on both theory and practise. It follows a standard textbook in the field, with additional material used to keep the course up-to-date. You will cover a range of basic graphics techniques, from the generation of graphics on raster output devices to the types of hardware used to display three-dimensional objects. It will be of benefit to anybody who uses computer graphics to display objects or data.

COMPUTER NETWORKS

This course covers the core theory of Computer Networks in order for students to understand the science underpinning computer communications, such as basic architectural principles of computer networking, including how the Internet works today and applications of theory in current technology. The course will cover the problems of Computer Networks and the standard ways to approach and resolve these problems, including relevant real-world, state-of-the-art examples. The practicals for the course will allow students to apply theory to real-world examples.

COMPUTER SECURITY
Security aspects of computer systems have far-reaching implications in an increasingly networked world. In this course we will cover the principles underlying computer security, the problems that must be solved, and some of the solutions that are used in implementing secure systems. The course includes the use of formal models based on logic to model security problems and to verify proposed solutions.

CONCURRENCY

Computer networks, multiprocessors and parallel algorithms, though radically different, all provide examples of processes acting in parallel to achieve some goal. All benefit from the efficiency of concurrency yet require careful design to ensure that they function correctly. The concurrency course introduces the fundamental concepts of concurrency using the notation of Communicating Sequential Processes. By introducing communication, parallelism, deadlock, live-lock, etc., it shows how CSP represents, and can be used to reason about, concurrent systems. Students are taught how to design interactive processes and how to modularise them using synchronisation. One important feature of the module is its use of both algebraic laws and semantic models to reason about reactive and concurrent designs. Another is its use of FDR to animate designs and verify that they meet their specifications.

DATA VISUALISATION

Well-designed visualisations capitalise on human facilities for processing visual information and thereby improve comprehension, memory, inference, and decision making. In addition, the advent of the so-call “explosion of Big Data” has increased the needs of effective visualisation systems for data analysis and communication. In this course we will study techniques and algorithms for creating effective visualisations based on principles from graphic design, visual art, perceptual psychology, and cognitive science. The course is targeted both towards students interested in using visualisation in their own work, as well as students interested in building better visualisation tools and systems.

DATABASES

Databases are at the heart of modern commercial application development. Their use extends beyond this to many applications and environments where large amounts of data must be stored for efficient update and retrieval. Their principles and fundamental techniques are being extended today to the Web and to novel kinds of data, like XML. The purpose of this course is to provide you with an introduction to the principles of database systems. You will begin by studying database design, covering the entity relationship model, and will then cover the relational data model, relational algebra and SQL. As a newer model, XML and some XML query languages will be covered. You will take a deeper look at database query languages and their connection with logic and with complexity classes. In the second half, we will discuss some database implementation issues such as storage, indexes, query processing, and query optimization.

DEEP LEARNING IN HEALTHCARE

Neuroscience research has inspired deep learning, and with the increasing digitisation of the medical domain, deep learning is set to advance healthcare. Deep learning has the potential to transform healthcare in areas ranging from medical imaging to electronic health records. This course aims to introduce students to recent advances in deep learning in healthcare, and the application of deep learning algorithms to medical data, including practical considerations for adapting models to the diversity of healthcare data. We will first introduce the biological basis for neural networks, the foundations of neural networks, and how the numerical operations are based on concepts from linear algebra, continuous mathematics, and probability, and how they are applied to train supervised models. We will then cover the computational components (e.g. convolutional neural networks) that form the backbone of powerful tools in deep learning, and how these can be deployed in the context of a variety of medical data (e.g. images). We will then cover networks that exploit the sequential (or temporal) structures in the data. Throughout the course, we will cover the practicalities of training neural networks, focusing particularly on applications in the healthcare domain, including discussion of optimisation, scalability, privacy, and fairness.

GEOMETRIC MODELLING

This is an introductory course in modelling techniques for 3D objects. It covers a wide range of different ways of representing the geometry of real objects, depending on their functionality and application. The emphasis in this course will be on the theory and basic principles of constructing models; hence the practicals will not use CAD-specific software, but rather a programming environment suitable for reasoning about the mathematics of models.

KNOWLEDGE REPRESENTATION & REASONING

Knowledge Representation and Reasoning (KRR) is the field of Artificial Intelligence concerned with the encoding of knowledge in a machine-understandable way. This knowledge can then be processed automatically by suitable computer programs. KRR is at the core of so-called Semantic Technologies which are being widely deployed in applications.

This course is an up-to-date survey of selected topics in KRR. The course starts with a review of the basics of Propositional and First-Order logic and their use in the context of KRR. The second part of the course describes formal languages for KRR that are based on fragments of first-order logic and surveys the main reasoning techniques typically implemented for those fragments. Finally, the last part of the course surveys the important topic of non-monotonic reasoning. The course discusses also several applications were KRR-based technologies are currently being used.

LAMBDA CALCULUS AND TYPES

The lambda calculus is the prototypical functional programming language. Its basic theory is extremely rich, allowing us to explore many important phenomena in practical computer science in an elegant mathematical setting. Topics covered include the equational theory, term rewriting and reduction strategies, combinatory logic, Turing completeness and type systems.

LOGIC AND PROOF

Logic plays an important role in many disciplines, including Philosophy and Mathematics, but it is particularly central to Computer Science. This course emphasises the computational aspects of logic, including applications to databases, constraint solving, programming and automated verification, among many others. We also highlight algorithmic problems in logic, such as SAT-solving, model checking and automated theorem proving.

The course relates to a number of third-year and fourth-year options.Propositional and predicate logic are central to Complexity Theory, Knowledge Representation and Reasoning, and Theory of Data and Knowledge Bases. They are also used extensively in Computer-Aided Formal Verification, Probabilistic Model Checking and Software Verification.

MACHINE LEARNING

Machine learning techniques enable us to automatically extract features from data so as to solve predictive tasks, such as speech recognition, object recognition, machine translation, question-answering, anomaly detection, medical diagnosis and prognosis, automatic algorithm configuration, personalisation, robot control, time series forecasting, and much more. Learning systems adapt so that they can solve new tasks, related to previously encountered tasks, more efficiently.

This course will introduce the field of machine learning, in particular focusing on the core concepts of supervised and unsupervised learning. In supervised learning we will discuss algorithms which are trained on input data labelled with a desired output, for instance an image of a face and the name of the person whose face it is, and learn a function mapping from the input to the output. Unsupervised learning aims to discover latent structure in an input signal where no output labels are available, an example of which is grouping web-pages based on the topics they discuss. Students will learn the algorithms which underpin many popular machine learning techniques, as well as developing an understanding of the theoretical relationships between these algorithms. The practicals will concern the application of machine learning to a range of real-world problems.

PHYSICS INFORMED NEURAL NETWORKS

Mathematical models have been used to investigate physical phenomena for centuries. Modelling is embedded in numerous scientific and non-scientific areas, including the fields of chemistry, physics, economics and social sciences, to name but a few. As the scientific world and ideas expanded, the systems have become more and more complex, moving from simple relationships between user inputs and observed outputs, to idealised differential equations and onto huge systems of highly non-linear partial differential equations. The major aim of this course is to present the concept of physics informed neural network approaches to approximate solutions systems of partial differential equations.

PRINCIPLES OF PROGRAMMING LANGUAGES
This course uses interpreters written in Haskell as a vehicle for exploring various kinds of programming languages.

PROBABILITY

The first half of the course takes further the probability theory that was developed in the first year. The aim is to build up a range of techniques that will be useful in dealing with mathematical models involving uncertainty. The second half of the course is concerned with Markov chains in discrete time and Poisson processes in one dimension, both with developing the relevant theory and giving examples of applications.

QUANTUM INFORMATION

The course is designed for computer science undergraduates and focuses on the general theory of quantum information, independently of the physical realizations.

REQUIREMENTS

Many software and hardware development projects go through a phase called 'Requirements Capture and Analysis' which tries to determine the properties a system should have in order to succeed in the environment in which it will be used. This can be a very difficult task, and typical requirements documents contain errors, some of which are very difficult to detect, as well as very expensive to correct later on. Experience shows that many errors arise from social, political and cultural factors and recent research has focused on the problem of reconciling such factors with traditional concerns about the more technical aspects of system development.

This course takes a unique stance to the discussion of requirements in that it acknowledges the involvement of both the social and technical concerns. The course surveys a wide range of different approaches to the problem of determining requirements and aims to provide students with a set of techniques and skills that may be tailored to address a wide range of requirements problems.

SCIENTIFIC COMPUTING

The overarching aim of this course is to present the theory for a range of techniques used in scientific computing in a manner that allows problems to be solved efficiently and robustly through: (i) the choice of a suitable numerical algorithm for a specific problem; and (ii) understanding of how to utilise an implementation of the numerical algorithm efficiently.

The first component of the course will present the theory that underpins two commonly used iterative methods for the iterative solution of linear systems, including preconditioning techniques that may be used to accelerate these iterative methods. Aside from black box preconditioners, the potential to exploit the structure of the matrix will be demonstrated and related to the error analysis for iterative methods. This material will then be shown to have application when solving constrained minimisation problems using Newton's method. The second component of the course will focus on practical methods for the numerical solution of initial value ordinary differential equations to within a specified tolerance. We will then exploit probabilistic methods to explain how a stochastic simulation of chemical reactions between a large number of molecules may be reduced to a system of initial value ordinary differential equations. We conclude with a brief introduction to Bayesian inference, explaining how this may be used as an alternative to classical optimisation techniques when attempting to estimate parameters from experimental data that is subject to experimental error.

A STEGANOGRAPHY APP FOR FACEBOOK

Steganography means hiding a hidden payload within an apparently-innocent cover, usually an item of digital media. Facebook is the ideal platform for the transmission of payload-carrying images, since images are so commonly uploaded and shared. And Facebook does not permit encrypted communications between users, who might seek an alternative way to preserve their privacy. This project is to develop a steganography app for Facebook, which allows messages to be sent and received inside pictures.

BODY LANGUAGE

The Microsoft Kinect is a device with a depth camera that measures the distance out into a scene. It has software to detect the "skeleton" of people standing in front of it and to extract their "gestures". The idea of this project is to use a Kinect to build a gesture-based control system for some electro-mechanical device. This might be a robot, or some other device, like a toy or an iPod player. The challenge is to devise and implement an easy to use interface to control the device by movement.

BUILDING AN ANIMATION OR SIMULATION SYSTEM

The simulation of objects and beings within a computer has become a hot topic over the last few years, with applications to virtual reality, the design of safe stadia, synthetic actors and environments for films, and computer games, amongst others. This project would look at the design and construction of a relatively simple animation or simulation system, that would probably sacrifice ease of animation in order to focus on the modelling of interactions between the actors and their environments.

COMPUTATIONAL LINGUISTICS: ELLIPSIS INTERPRETATION

Sentences like: 'John likes Chopin, but Mary doesn't', or 'John likes films, but not plays' contain ellipsis - 'missing' components that have to be filled in from earlier parts of the sentence: John likes Chopin, but Mary doesn't like Chopin', 'John likes films, but John doesn't like plays'. If you run the first pair of sentences through a parser like the Stanford parser: you will see that it gets the constituent structure right, but doesn't attempt to capture the interpretation. This project will attempt to develop and test a post-processing module for the Stanford parser which will interpret some common cases of ellipsis.

ELECTRONIC PET

Microsoft .NET Gadgeteer is an open-source toolkit for building small electronic devices. We have a whole bunch of electronic bits and pieces from this toolkit in a box - your challenge is to bring them to life, and make an electronic pet. The CS Department doesn't have a cat, but by the end of this project it could have a mouse, or a beetle, or a furby - can you create one? Will it make blood-curdling noises? Scuttle into a dark corner? Demand to be stroked? React to your mood? Or chase off burglars?

FRIEND FINDING BY PHONE

There are smartphone apps to help you find friends (foursquare, blendr, grindr) but they only find people who have already signed up. How can you use a phone to find interesting people nearby who haven't signed up to your app? By their tweets? Any other ideas? We have Windows and Bada smartphone handsets available - can you design and build an app to find interesting people around you by whatever signals they send out?

GEOMLAB AND MINDSTORMS

Geomlab has a turtle graphics feature, but the pictures are drawn only on the screen. It should be possible to make a turtle out of Lego Mindstorms, then control it with an instance of Geomlab running on a host computer, with communication over Bluetooth.

MINE4NUGGETS. SEARCHING FOR OBJECTS INTO IMAGES

Most of the information on commercial websites (e.g., Amazon, Rightmove, and Autotrader) is represented by (semi-)structured pages where the interesting data is structured into structured or visual patterns. However, some critical information is often represented with an image or a chart. As an example, on many estate agents' websites, data about the energy efficiency of the property is represented by an EPC Chart, while the actual size of the rooms is represented through a floor-plan Have you ever tried to search for an energy-efficient property or one with a large bedroom? The answer is: you can't. This project is concerned with the study of semantic image analysis techniques that can be used to locate and extract interesting objects from pictures and charts on the web. The goal is to enhance current search technology with object-search capabilities for images, a field that offers challenging research and potential commercial opportunities.

PARALLEL LINEAR ALGEBRA

The most common large-scale parallel architecture for scientific computing is a distributed memory computer. When using this architecture each processor has access to data stored in its own memory, while data stored by memory associated with other processors may only be accessed by communication across a network. For reasons of computational efficiency, many parallel scientific computing algorithms are implemented by partitioning the data stored so that communication across the network between processors is minimized. This project will investigate the most appropriate partitioning for linear algebra calculations.

ROBOT PATH PLANNING

A robot cannot find its way by looking at a map designed for humans; rather, it must make a complex series of geometric calculations. Many algorithms have been developed for robot path planning, and the purpose of this project is to investigate (and improve upon) some of these. The 'robot' might be an arbitrary convex polygon that can slide in a world composed of polygonal obstacles; an industrial robot arm; or a human trying to find their way across a strange city. Which algorithm(s) you investigate is largely up to you: they range from pragmatic, 'suck-it-and-see' methods to mathematically-complicated methods that are guaranteed to succeed (after a lot of CPU time!). There is also considerable scope to think of novel robots. For example, previous groups have considered: a planetary exploration vehicle; a smart torpedo; and a model railway track.

RESILIENT AND RAPID RASPBERRIES

In today's world of high frequency trading, understanding the rapidly changing quotes from a myriad of trading algorithms is a colossal data processing problem. With tens of thousands of updates per second in a single market, and limited space and power constraints, it is important to be able to maximise throughput and resiliency with the hardware available. Given a day of actual stock exchange data, your challenge is to create a cluster to process and aggregate the data. Doing this with real hardware is more fun - we cannot give you our data centre, but we can supply you with a Raspberry Pi cluster.

SELL COMPUTER SCIENCE
Apparently "Computer Science is boring. It's all about Microsoft Word, and is only for geeks and boys." No, we don't think so either. Create something, aimed at 14-17year olds, that introduces one or more important CS concepts. (For inspiration see Manufactoria, cs4fn, yousrc, Maze or GeomLab.) Your game/tool/app should inspire young people to want to get into computing. We want something that will attract attention on the Department's prospective students website (a game? a downloadable app?) - ideally something that we can also use with groups on open days. Let your imagination run wild!

COMPUTER SECURITY VISUALISATION

Many techniques exist to detect and analyse potential cyber-attacks from network traffic data. Using computer graphics and visualisation, people can detect patterns and attacks faster than many automated analytical approaches. The purpose of this group project is to develop a number of visualisations (on a single group code-base) to improve situational awareness in an intuitive and meaningful way to a user. The core task is to address is how to make an attack distinct from standard and benign network traffic. We encourage the students to develop their own ideas.

SOME OTHER PROJECTS
Other projects on offer have included:
Creating Patient-Specific Models of the Lung to Investigate Chronic Obstructive Pulmonary Disease
Docu-Research. Object Annotation and Visualization for PDF Documents
Medical Image Analysis
Model Checking for DNA Computation
Modelling and Reasoning about Pervasive Computing Systems
Querying Blogs or Wikis
Robot Sheepdog
Robot Soccer Simulation
Smartphone Security
Further examples are available here.

Year 3

ADVANCED SECURITY
The Advanced Security course is designed to bring students towards the research boundaries in computer security, covering contemporary topics in depth. It is split into two modules:
Bootstrapping Security. Modern technology brings many opportunities for connecting devices together by means such as wifi, Bluetooth or connection to the internet. In many cases we want to make these connections secure: authenticating the identity of the device connected to, making them secret and carrying out (e.g. financial) transactions using them. In this module we examine the theory and practice of doing this using both conventional means (using a pre-existing security structure) and methods based on things such as co-location and human judgement.
Information Hiding. Steganography is the art and science of hiding a secret payload inside an innocent cover, typically hiding inside digital media such as images, movies, or mp3 audio. The course covers the definitions, practice, and theory of hidden information in way accessible to newcomers. We will cover the details of getting inside digital images in order to hide information, the different sorts of embedding operations commonly used, and how hidden information can be detected. Some linear algebra allows for more efficient hiding. Finally, we see some mathematical theory which explains how the amount of hidden data grows with the space in which is can be hidden.

AUTOMATA, LOGIC AND GAMES
To introduce the mathematical theory underpinning the Computer-Aided Verification of computing systems. The main ingredients are:
Automata (on infnite words and trees) as a computational model of state-based systems
Logical systems (such as temporal and modal logics) for specifying operational behaviour
Two-person games as a conceptual basis for understanding interactions between a system and its environment
Connections between logical reasoning over arbitrary structrures and automata over trees

BAYESIAN STATISTICAL PROBABILISTIC PROGRAMMING

Statistical probabilistic programming (SPP) is a general framework for expressing probabilistic models as computer programs. SPP systems are equipped with implementations of genericinference algorithms for answering queries about these models, such as posterior inference and marginalisation. By providing these algorithms, SPP systems enable data scientists to focuson the design of good models, utilising their domain knowledge. The task of constructing efficient generic inference engines can be left to researchers with expertise in statistical machine learning and programming languages.

CATEGORIES, PROOFS AND PROCESSES
Category Theory is a powerful mathematical formalism which has become an important tool in modern mathematics, logic and computer science. One main idea of Category Theory is to study mathematical `universes', collections of mathematical structures and their structure-preserving transformations, as mathematical structures in their own right, i.e. categories - which have their own structure-preserving transformations (functors). This is a very powerful perspective, which allows many important structural concepts of mathematics to be studied at the appropriate level of generality, and brings many common underlying structures to light, yielding new connections between apparently different situations.

COMPUTER VISION
This is an advanced course in modern computer vision and machine learning. It contains fundamental concepts from classical computer vision: filtering, matching, indexing and 3D computer vision. On top of that, a large portion of the course focuses on current computer vision methodologies and problems, which build on top of deep learning techniques: detection, segmentation, generation, and vision and language models. This course will introduce the fundamental mathematical concepts behind these tasks and how they can be integrated into modern machine-learning models. The taught material and assessment include both theoretical derivations as well as applied implementations, and students are expected to be proficient with both.

COMPUTATIONAL BIOLOGY

This course is intended for students who want to understand modern computational (molecular/structural) biology. The course will provide an introduction to the central dogma of molecular biology and will cover fundamental methods for sequence and structure analysis as well as some essential concepts from statistical mechanics. It will then discuss algorithmic approaches for deciphering the relationship between sequence and structure and techniques to model biomolecular structures. The course will present examples for the regulation of the information flow in molecular biology, the effect of epigenetics and computational methods facilitating genome editing applications.

COMPUTATIONAL GAME THEORY

Game theory is the mathematical theory of strategic interactions between self-interested agents. Game theory provides a range of models for representing strategic interactions, and associated with these, a family of solution concepts, which attempt to characterise the rational outcomes of games. Game theory is important to computer science for several reasons: First, interaction is a fundamental topic in computer science, and if it is assumed that system components are self-interested, then the models and solution concepts of game theory seems to provide an appropriate framework with which to model such systems. Second, the problem of computing with the solution concepts proposed by game theory raises important challenges for computer science, which test the boundaries of current algorithmic techniques. This course aims to introduce the key concepts of game theory for a computer science audience, emphasising both the applicability of game theoretic concepts in a computational setting, and the role of computation in game theoretic problems. The course assumes no prior knowledge of game theory. The aims of the course are threefold: 1. to introduce the key models and solution concepts of non-cooperative and cooperative game theory; 2. to introduce the issues that arise when computing with game theoretic solution concepts, and the main approaches to overcoming these issues, and to illustrate the role that computation plays in game theory; 3. to introduce a research-level topic in computational game theory.

COMPUTATIONAL LEARNING THEORY
Machine learning studies automatic methods for identifying patterns in complex data and for making predictions based on past observations. In this course, we develop rigorous mathematical foundations of machine learning, in order to provide guarantees about the behaviour of learning algorithms and also to understand the inherent difficulty of learning problems. The course will begin by providing a statistical and computational toolkit, such as concentration inequalities, fundamental algorithms and methods to analyse learning algorithms. We will cover questions such as when can we generalise well from limited amounts of data, how can we develop algorithms that are compuationally efficient, and understand statistical and computational trade-offs in learning algorithms. We will also discuss new models designed to address relevant practical questions of the day, such as learning with limited memory, communication, privacy, and labelled and unlabelled data. In addition to core concepts from machine learning, we will make connections to principal ideas from information theory, game theory and optimisation. This is a mathematical course that with several definitions, theorems and proofs. It is expected that students will have some familiarity with probability, algorithms, and basic complexity. A formal course on Machine Learning is not required, however, students are more likely to appreciate the contents of the course if they have some familiarity with machine learning.

COMPUTATIONAL MEDICINE

This course is intended for students who want to understand modern computational medicine through a focus on computational cardiology. The course will provide an introduction to the challenges and techniques used in computational cardiology, and will cover fundamental methods for modelling and simulation of patient-specific multiscale and multi-physics human hearts in health and disease and their use for in silico testing of therapies. It will cover both functional and anatomical modelling from multimodality data, and important concepts broadly applicable to all areas of medicine in industry, academia and regulatory practice such as the digital twin vision, in silico trials for therapy evaluation and verification, validation and uncertainty quantification.

CONCURRENT ALGORITHMS AND DATA STRUCTURES

This is an advanced course on concurrent programming. The course will combine principles and practice. Principles to be studied include correctness conditions for concurrent datatypes, and the relative power of different synchronization operations. More practical topics will include how to implement concurrency primitives - such as locks and monitors - and concurrent datatypes - such as linked lists, queues, and hash tables.

This course is largely independent of the second year Concurrent Programming course, although there are clearly strong links between the two. Concurrent Programming is based upon the message-passing paradigm; much of the emphasis is on using concurrency as a way of structuring large programs. This course will be based upon low-level concurrency primitives, such as compare-and-swap; the emphasis will be on speed. MSc students could take this course in Michaelmas, followed by Concurrent Programming in Hilary.

DISTRIBUTED PROCESSES, TYPES AND PROGRAMMING

This course studies the foundations and type theory of mobile processes and programming languages for communication and distribution. Specifically, this course studies how to program communication through an introduction of a channel-based message-passing process calculus (the pi-calculus) and how to apply its type theory to practice. A course also functions as a brief introduction of message-passing programming languages such as Go language (a popular recent programming language designed by Google, which is widely used for implementing large distributed system). The course assumes a basic knowledge of programming languages. Some knowledge of concurrent programming, lambda-calculus or operational semantics would be useful but not required.

FOUNDATIONS OF SELF-PROGRAMMING AGENTS

The course will appeal to students who want to gain a better understanding of modern deep learning and will present a systematic geometric blueprint allowing them to derive popular deep neural network architectures (CNNs, GNNs, Transformers, etc) from the first principles of symmetry and invariance. The focus will be on general principles that underpin deep learning as well as concrete examples of their realisations and applications. The course will try to tie together topics in geometry, group theory and representation learning, graph theory, and machine learning into a coherent picture. It ideally targets students in CS & Math cohort or CS students with a strong mathematical background.

GEOMETRIC DEEP LEARNING
The course will appeal to students who want to gain a better understanding of modern deep learning and will present a systematic geometric blueprint allowing them to derive popular deep neural network architectures (CNNs, GNNs, Transformers, etc) from the first principles of symmetry and invariance. The focus will be on general principles that underpin deep learning as well as concrete examples of their realisations and applications. The course will try to tie together topics in geometry, group theory and representation learning, graph theory, and machine learning into a coherent picture. It ideally targets students in CS & Math cohort or CS students with a strong mathematical background.

GRAPH REPRESENTATION LEARNING

This is an advanced course on machine learning with relational data, focusing on the recent advances in the field of graph representation learning. The goal is to provide a systematic coverage of the fundamentals and foundations of graph representation learning. The course will introduce the definitions of the relevant machine learning models (e.g., graph neural networks), discuss their mathematical underpinnings, formally study their properties (e.g., relational inductive bias, expressive power), and demonstrate ways to effectively develop and train such models.

KNOWLEDGE REPRESENTATION & REASONING

Knowledge Representation and Reasoning (KRR) is the field of Artificial Intelligence concerned with the encoding of knowledge in a machine-understandable way. This knowledge can then be processed automatically by suitable computer programs. KRR is at the core of so-called Semantic Technologies which are being widely deployed in applications.

This course is an up-to-date survey of selected topics in KRR. The course starts with a review of the basics of Propositional and First-Order logic and their use in the context of KRR. The second part of the course describes formal languages for KRR that are based on fragments of first-order logic and surveys the main reasoning techniques typically implemented for those fragments. Finally, the last part of the course surveys the important topic of non-monotonic reasoning. The course discusses also several applications were KRR-based technologies are currently being used.

LAW AND COMPUTER SCIENCE

The legal system is entering a period of profound transformation brought on by new technologies and alternative business models. Legal innovation backed by new technologies can drive down costs, make the delivery of legal services more productive, and facilitate better access to justice for citizens. As AI and digital technology permeate more of our lives, they increasingly becomes the source of legally significant events. This means that those who study and/or practice law increasingly need to understand the digital context. At the same time, those who study computer science and/or develop software increasingly need to understand potential legal consequences of design choices. Increasingly law firms are interested in hiring not just those with legal skills but also those with technical skills, and so there are exciting career opportunities for those working at the intersection of law and technology

This course, jointly offered by the Law Faculty and the Department of Computer Science, will introduce students from both backgrounds to the terrain at the boundaries of their two disciplines. The overarching theme is understanding law and computer science at their intersection.

PROBABILISTIC MODEL CHECKING

Probabilistic model checking is a formal technique for analysing systems that exhibit random behaviour. Examples include randomised algorithms, communication and security protocols, computer networks, biological signalling pathways, and many others. The course provides a detailed introduction to these techniques, covering both the underlying theory (Markov chains, Markov decision processes, temporal logics) and its practical application (using the state-of-the art probabilistic checking tool PRISM, based here in Oxford). The methods used will be illustrated through a variety of real-life case studies, e.g. the Bluetooth/FireWire protocols and algorithms for contract signing and power management.

QUANTUM PROCESSES AND COMPUTATION

Both physics and computer science have been very dominant scientific and technological disciplines in the previous century. Quantum Computer Science aims at combining both and may come to play a similarly important role in this century. Combining the existing expertise in both fields proves to be a non-trivial but very exciting interdisciplinary journey. Besides the actual issue of building a quantum computer or realising quantum protocols it involves a fascinating encounter of concepts and formal tools which arose in distinct disciplines. This course provides an interdisciplinary introduction to the emerging field of quantum computer science, explaining basic quantum mechanics (including finite dimensional Hilbert spaces and their tensor products), quantum entanglement, its structure and its physical consequences (e.g. non-locality, no-cloning principle), and introduces qubits. We give detailed discussions of some key algorithms and protocols such as Grover's search algorithm and Shor's factorisation algorithm, quantum teleportation and quantum key exchange. At the same time, this course provides an introduction to diagrammatic reasoning. As an entirely diagrammatic presentation of quantum theory and its applications, this course is the first of its kind.

UNCERTAINTY IN DEEP LEARNING

This is an advanced course in machine learning, focusing on recent advances in deep learning specifically such as Bayesian neural networks. The course will concentrate on underlying fundamental methodology as well as on applications, such as in autonomous driving, astronomy, and medical applications. Recent statistical techniques based on neural networks have achieved remarkable progress in these domains, leading to a great deal of commercial and academic interest. The course will introduce the mathematical definitions of the relevant machine learning models and derive their associated approximate inference algorithms, demonstrating the models in the various domains. The taught material and assessment include both theoretical derivations as well as applied implementations, and students are expected to be proficient with both.