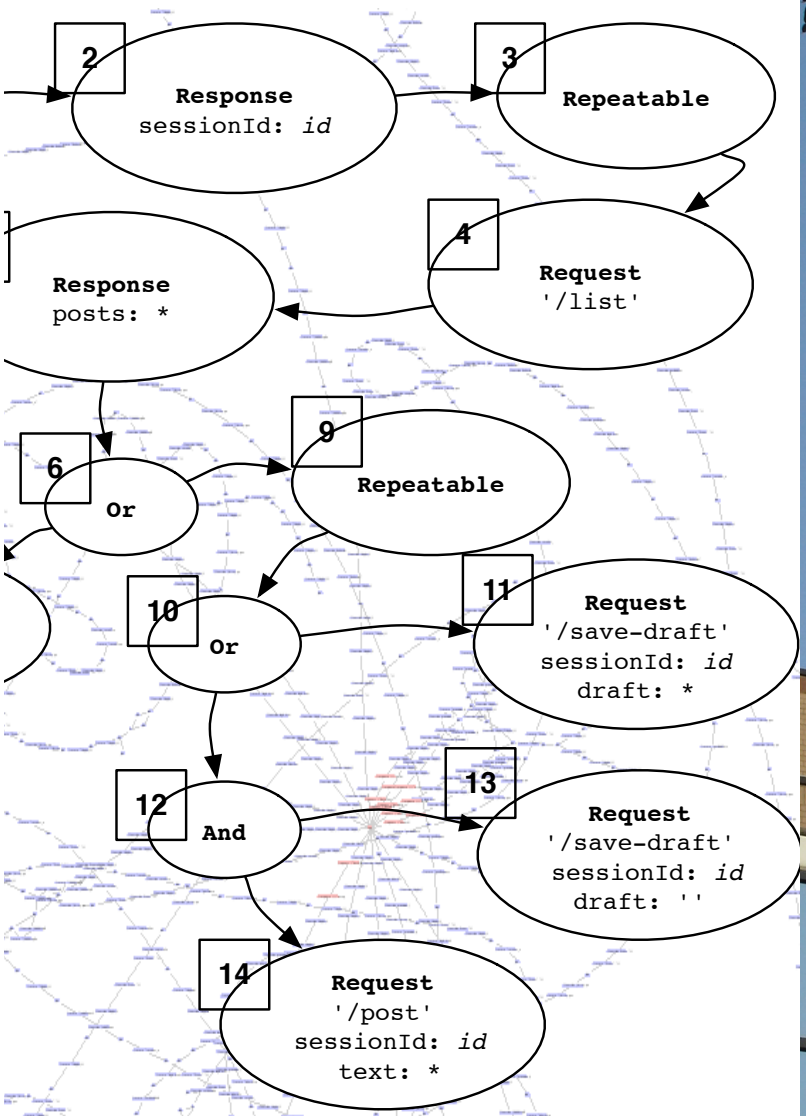
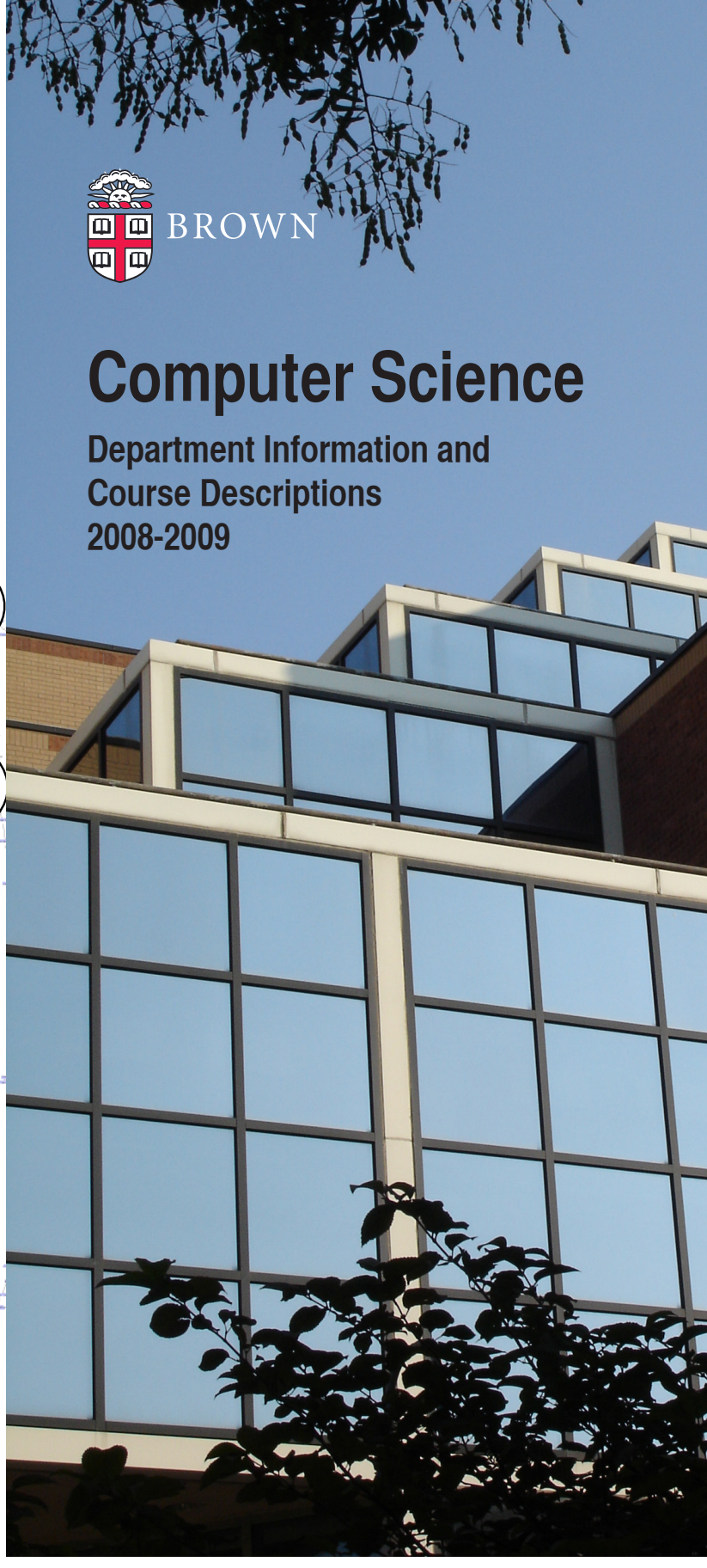


Brown University
Department of Computer Science
115 Waterman Street, Box 1910
Providence, RI 02912
www.cs.brown.edu

BROWN

Computer Science

Department Information and Course Descriptions 2008-2009



CONTENTS

This document details **ALL** courses taught by the department, not just those taught this year. For the latest information on each semester's teaching schedule, please check: **www.cs.brown.edu/courses**.

COMPUTER SCIENCE

Computer science combines the intellectual challenge of a young discipline with the excitement of an innovative and rapidly expanding technology. It has been an active area at Brown for 40 years and has been a department for nearly 30 years. The department resides in Brown's Center for Information Technology, a striking building housing many of the University's computing activities. Faculty, staff and students are provided state-of-the-art computing facilities managed by the department's technical staff.

The Department of Computer Science offers standard Sc.B. and A.B. concentrations, standard concentrations in mathematics and computer science, applied mathematics and computer science, economics and computer science, and computational biology, as well as a Master's program and a Ph.D. program.

The undergraduate program is designed to combine educational breadth in practical and theoretical computer science with deeper understanding of specialized areas such as analysis of algorithms, artificial intelligence, computer graphics, computer security, computer systems and networks, information management, programming languages and compilers, software engineering, and theory of computation. Undergraduates often take at least one semester of faculty-supervised independent study, working either on a project of their own choice or as members of a team on a faculty-sponsored research project.

The department also provides a wealth of opportunities for graduate research in computer science. Graduate students at Brown pursue research in a number of areas. Short descriptions of current research areas can be found starting on page 33. Our established coordination with other departments of the University provides an unusual opportunity for advanced research both in traditional "core" computer science and in areas combining computer science and such fields as applied mathematics, cognitive science, economics, engineering, and biology and medicine.

FACULTY

Michael J. Black, Professor, Ph.D., Yale University, 1992. Computer vision, optical flow estimation, human motion analysis, probabilistic modeling of natural scenes, forensic video analysis, neural engineering.

Ugur Cetintemel, Associate Professor, Ph.D., University of Maryland, College Park, 2002. Advanced data management systems.

Eugene Charniak, University Professor of Computer Science, Ph.D., Massachusetts Institute of Technology, 1972. Artificial intelligence, natural language processing.

Thomas W. Doeppner, Associate Professor (Research) and Vice Chair, Ph.D., Princeton University, 1977. Operating systems, concurrent and distributed programming, security.

Rodrigo Fonseca, Assistant Professor (starting in 2009-2010), Ph.D., University of California, Berkeley, 2008. Operating systems, distributed systems, and networking, especially large-scale Internet systems, embedded and mesh wireless networking.

Amy R. Greenwald, Associate Professor, Ph.D., Courant Institute of Mathematical Sciences, New York University, 1999. Internet agent economics, multi-agent learning in games, automated game-theoretic reasoning.

Maurice P. Herlihy, Professor, Ph.D., Massachusetts Institute of Technology, 1984. Concurrent and distributed computing.

John F. Hughes, Professor, Ph.D., University of California, Berkeley, 1982. Computer graphics systems, application of mathematics to fundamentals of computer graphics, gesture-based interfaces for modeling, art-based graphics.

Sorin Istrail, Julie Nguyen Brown Professor in Computational and Mathematical Sciences and Professor of Computer Science, Ph.D., University of Bucharest, 1979. Computational Biology: genomics, gene regulatory networks, genetics basis of complex diseases, protein folding, medical bioinformatics, programming languages for genomics; Combinatorial algorithms, graph theory, computational complexity; Statistical physics and theory of complex systems.

John Jannotti, Assistant Professor, Ph.D., Massachusetts Institute of Technology, 2002. Computer systems, broadly construed — especially operating systems, networking and mobile systems. Particularly interested in loosely coupled distributed systems enabling qualitatively new functionality.

Odest Chadwicke Jenkins, Assistant Professor, Ph.D., University of Southern California, 2003. Robotics, computer vision, computer animation, machine learning, human motion capture and analysis.

Philip N. Klein, Professor, Ph.D., Massachusetts Institute of Technology, 1988. Algorithms for combinatorial optimization, approximation algorithms, graph algorithms.

Shriram Krishnamurthi, Associate Professor, Ph.D., Rice University, 2001. Programming languages and environments, computer-aided verification, security, software engineering.

David H. Laidlaw, Professor, Ph.D., California Institute of Technology, 1995. Interdisciplinary research into robust and effective computational, modeling, and visualization tools to solve problems in biology, fluid dynamics, archaeology, medical imaging, and other disciplines.

Anna Lysyanskaya, Associate Professor, Ph.D., Massachusetts Institute of Technology, 2002. Cryptography, theory of computation, computer security, secure distributed algorithms.

Claire Mathieu, Professor, Ph.D., University of Paris-Sud (France), 1988. Approximation algorithms for combinatorial optimization, probabilistic analysis of algorithms.

Franco P. Preparata, An Wang Professor, Dr. Eng., University of Rome, 1959. Computational biology, computational geometry and metrology, parallel computation, design and analysis of algorithms, probabilistic analysis of algorithms.

Benjamin Raphael, Assistant Professor, Ph.D., University of California, San Diego, 2002. Combinatorial optimization, computational biology, design and analysis of algorithms.

Steven P. Reiss, Professor, Ph.D., Yale University, 1977. Programming environments, software engineering, user interfaces, software visualization, software understanding.

John E. Savage, Professor, Ph.D., Massachusetts Institute of Technology, 1965. Applied theory of computation, parallel computation, design and analysis of algorithms, models of computation, nanocomputing.

Meinolf Sellmann, Assistant Professor, Dr. rer. nat., University of Paderborn, 2002. Combinatorial optimization and feasibility: operations research, algorithms, constraint programming.

Erik B. Sudderth, Assistant Professor (starting in 2009-2010), Ph.D., Massachusetts Institute of Technology, 2006. Machine learning, computer vision, statistical signal and image processing, probabilistic inference, Bayesian nonparametrics.

Roberto Tamassia, Professor and Chair, Ph.D., University of Illinois, 1988. Analysis, design, and implementation of algorithms, information security, cryptography, graph drawing and computational geometry.

Eli Upfal, Professor, Ph.D., Hebrew University, Jerusalem, 1983. Theory of computing, randomized algorithms, stochastic analysis of algorithms, Internet and Web modeling and algorithms, parallel and distributed computing, computational biology.

Andries van Dam, Thomas J. Watson Jr. University Professor of Technology and Education and Professor of Computer Science, Ph.D., University of Pennsylvania, 1966. Interactive computer graphics, particularly immersive virtual reality and pen-centric computing; hypermedia and electronic books, including interactive illustrations, and web-based learning.

Pascal Van Hentenryck, Professor, Ph.D., University of Namur (Belgium), 1987. Combinatorial optimization, programming languages, artificial intelligence, constraint programming, decision making under uncertainty, computational biology, software and hardware verification, numerical analysis.

Peter Wegner, Professor Emeritus, Ph.D., London University, 1968. Programming languages, software engineering.

Stanley B. Zdonik, Professor, Ph.D., Massachusetts Institute of Technology, 1983. Database management systems, stream-processing, automatic database design, scientific databases, querying uncertain data, databases and cloud computing.

VISITING, ADJUNCT AND JOINT FACULTY

Roger B. Blumberg, Adjunct Lecturer, B.A., Columbia University, 1983. Computer science education, educational software design, computing & society, and the history & philosophy of technology.

Thomas L. Dean, Adjunct Professor, Ph.D., Yale University, 1986. Artificial intelligence, probabilistic inference, machine learning, robot problem solving.

Kathi Fisler, Adjunct Associate Professor, Ph.D., Indiana University, 1996. Computer-aided verification, access-control policy analysis and authoring, diagram-based specification languages.

Dina Goldin, Visiting Scientist, Ph.D., Brown University, 1997. Models of interactive computation and constraint database algebras, computing paradigms, similarity querying, languages for programming and querying, agent-oriented computing paradigms, algorithms and computer-aided design.

Mark Johnson, Professor of Cognitive & Linguistic Science and Computer Science. Ph.D., Stanford University, 1987. Computational linguistics, natural language processing.

Joseph J. LaViola, Jr., Adjunct Assistant Professor. Ph.D., Brown University, 2005. Pen-based computing, user interfaces, human motion estimation, virtual reality, and computer graphics.

Barbara Meier, Visiting Assistant Professor, Sc.M., Brown University, 1987. Computer animation, visual effects, nonphotorealistic rendering, designing tools for artists.

Donald Stanford, Adjunct Professor, Sc.M., Brown University, 1977. Enterprise Java and component engineering, online transaction processing models, standards-based networking models (IP), wireless communications systems.

Gabriel Taubin, Associate Professor of Engineering and Computer Science. Ph.D., Brown University, 1991. Licenciado en Ciencias Matemáticas, University of Buenos Aires, 1981. Computer vision, computer graphics, geometry modeling, mesh signal processing, geometry compression, smart cameras, smart sensor networks, embedded systems.

Alan Usas, Adjunct Associate Professor, Ph.D., Stanford University, 1976. Instructional and research applications of technology, security, fault-tolerant computing, entrepreneuring.

UNDERGRADUATE PROGRAMS

Undergraduate concentrations in computer science encourage students to take both theoretical courses that develop logical and mathematical reasoning abilities and practical courses that provide experience in the construction, design, and implementation of real computing systems.

The primary source for information on majoring in computer science is the Undergraduate Concentration Information, available at **www.cs.brown.edu/ugrad/concentrations/**.

Students can determine their concentration advisor from the list at **www.cs.brown.edu/ugrad/concentrations/advisors.html**

Writing Requirement: computer programming is a methodical expression of complicated ideas. An appropriate writing course serves both to sharpen this ability and to facilitate the description of programs to others, an integral part of the programming task. Concentrators in all programs except computational biology must take an approved writing course. To qualify for approval, a course should require at least two essays and be graded in part on the quality of these essays. To use a particular course to fulfill the writing requirement, students should have the approval form at **www.cs.brown.edu/ugrad/concentrations/writing.html** signed by the course's instructor.

Computer Science A.B.

The standard A.B. concentration in computer science has two prerequisites: any math course beyond MATH0090 (except for MATH0420 or APMA0330), as well as an approved writing course. It requires six core CS courses (or, if CSCI0190 is taken, five core courses): either CSCI0150 and CSCI0160, CSCI0170 and CSCI0180, or CSCI0190; plus CSCI0220, CSCI0310, CSCI0320, and an approved introductory CS theory course (currently only CSCI0510); and three advanced courses in CS or related areas. Two of the courses must include a pair of courses with a coherent theme. A list of approved course pairs is available at the approved course pair webpage **www.cs.brown.edu/ugrad/concentrations/approvedpairs.html.**

Computer Science Sc.B.

The standard Sc.B. concentration in computer science has two prerequisites: any math course beyond MATH0090 (except for MATH0420 or APMA0330), as well as an approved writing course. It requires two courses in mathematics or applied mathematics beyond MATH0100 or MATH0170, one of which must be a linear

algebra course (CSCI053, MATH0520, or MATH0540); an approved two-course sequence in physics, chemistry, biology, engineering or geological sciences; six core CS courses (or five core courses if CSCI 0190 is taken): either CSCI0150 and CSCI0160, CSCI0170 and CSCI0180, or CSCI1090; plus CSCI0220, CSCI0310, CSCI0320, and an approved CS theory course (currently only CSCI0510); and seven advanced (1000-level) courses in computer science or related areas, including one course in each of three subfields (no course may count for more than one subfield): CS systems (CSCI1230, CSCI1260, CSCI1270, CSCI1380, CSCI1600, CSCI1610, CSCI1660, CSCI1670, CSCI1680, CSCI1730, or CSCI1900), CS theory (CSCI1490, CSCI1510, CSCI1550, CSCI1570, CSCI1590, CSCI1730, or CSCI1760), artificial intelligence (CSCI1410, CSCI1430, CSCI1460, CSCI1480, or CSCI1490), and one capstone course (a one-semester course, normally taken in the student's last undergraduate year, in which the student (or group of students) uses a significant portion of his or her undergraduate education, broadly interpreted, in studying some current topic in-depth, to produce a culminating artifact such as a paper or software project). Four of the advanced courses must be approved pairs (see **www.cs.brown.edu/ugrad/concentrations/approvedpairs.html**). Four of the advanced courses must be CS courses.

Computer Science–Economics

The joint computer science–economics concentration exposes students to both theoretical and practical connections between computer science and economics. The intent of the concentration is to prepare students for either academic careers conducting research in areas that emphasize the overlap between the two fields or professional careers that incorporate aspects of economics and computer technology.

The concentration is offered in two versions, the A.B. and the Sc.B. While the A.B. degree allows students to explore the two disciplines by taking advanced courses in both departments, its smaller number of required courses is compatible with a liberal arts education. The Sc.B. degree achieves greater depth in both computer science and economics by requiring more courses, and it offers students the opportunity to integrate both disciplines creatively through a design requirement.

Computer Science–Economics A.B.

Interested students may contact concentration advisors in either the Department of Computer Science or the Department of Economics. Prerequisites: MATH0090 and MATH0100 or MATH0170.

MATH0520 or MATH0540. ECON0110. An approved writing course.

Required courses: APMA1650. Seven CS courses are required (or six courses if CSCI0190 is taken): either CSCI0150 and CSCI0160, CSCI0170 and CSCI0180 or CSCI0190; CSCI0220, CSCI0310, and CSCI0510; and two courses from either the analytical track (CSCI1410, CSCI1490, CSCI1550, CSCI1570, CSCI1590, CSCI1760, and one of APMA1210 and APMA1660) or the information systems track (CSCI0320, CSCI1230, CSCI1260, CSCO1270, CSCO1380, CSCI1430, CSCI1480, CSCI1660, CSCI1670, CSCI1680, CSCI1730, and CSCI1900). Six additional economics courses are required: ECON1100 or ECON1130 (or ECON1110 with permission), ECON1210, ECON1630, and three other 1000-level courses, of which two must be chosen from the "mathematical economics" group (ECON1170, ECON1470, ECON1640, ECON1750, ECON1850, ECON1860, and ECON1870).

Computer Science–Economics Sc.B.

Interested students may contact concentration advisors in either the Department of Computer Science or the Department of Economics. Prerequisites: MATH0090 and MATH0100 or MATH0170. MATH0520 or MATH0540. ECON0110. An approved writing course.

Required courses:

Applied Mathematics: APMA1650.

Computer Science: CSCI0150 and CSCI0160, CSCI0170 and CSCI0180, or CSCI0190. CSCI0220, CSCI0310, CSCI0320 and CSCI0510. One of the following tracks: (1) Analytical track. Two courses from the set CSCI1410, CSCI1490, CSCI1550, CSCI1570, CSCI1590, CSCI1760, and one of APMA1210 and APMA1660. (2) Information systems track. Two courses from the set CSCI1230, CSCI1260, CSCI1270, CSCI1380, CSCI1430, CSCI1480, CSCI1660, CSCI1670, CSCI1680, CSCI1730, CSCI1900. One additional 1000-level CS course.

Economics: ECON1110 or ECON1130, ECON1210 and ECON1630, plus at least five other 1000-level economics courses. Of those five courses, at least three must be chosen from the "mathematical economics" group, comprised of ECON1170, ECON1470, ECON1640, ECON1750, ECON1850, ECON1860, and ECON1870. Capstone course: a one-semester course, normally taken in the student's last undergraduate year, in which the student (or group of students) uses a significant portion of his or her undergraduate education, broadly interpreted, in studying some current topic, preferably at the

intersection of computer science and economics, in-depth, to produce a culminating artifact such as a paper or software project.

Applied Mathematics–Computer Science Sc.B.

Interested students may contact concentration advisors in the Division of Applied Mathematics or the Department of Computer Science. Prerequisites: any math course beyond MATH0090 (except for MATH0420 or APMA0330), as well as an approved writing course.

Required courses: MATH0180 or MATH0350, and MATH0520 or MATH0540; APMA0350, APMA0360, and either APMA1170 or APMA1180; CSCI0150 and CSCI0160, or CSCI0170 and CSCI0180, or CSCI0190, plus CSCI0220, and two of: CSCI0310, CSCI0320 or CSCI0510. (In some cases, substitutions of equivalent courses are permitted.) In addition, students must complete three 1000-level courses in applied math and three 1000-level courses in computer science and a capstone course.

The three computer science courses must include a pair of courses with a coherent theme. Approved course pairs can be found on the approved pair webpage
http://cs.brown.edu/ugrad/concentrations/approvedpairs.html

Of the 1000-level applied mathematics courses, at least two should constitute a standard sequence or address a common theme. For example, either of the pairs APMA1200–APMA1210 or APMA1650–APMA1660 is suitable. Capstone course: a one-semester course, normally taken in the student's last undergraduate year, in which the student (or group of students) use a significant portion of their undergraduate education, broadly interpreted, in studying some current topic in-depth, to produce a culminating artifact such as a paper or software project.

Computational Biology Sc.B.

Interested students may contact concentration advisors in the Department of Computer Science, the Division of Biology and Medicine, the Division of Applied Mathematics, or the Department of Chemistry. Prerequisites: MATH0100 or MATH0170, and BIOL0200, or equivalents.

Sixteen courses are required beyond the prerequisites. General core course requirements: these include one course in organic chemistry (CHEM0330); two courses in biology (BIOL0470, and BIOL0280 or BIOL0500); and two courses in computer science (CSCI0150 and CSCI0160), or (CSCI0170 and CSCI0180) or (CSCI0190), plus all

students are required to take (CSCI0220), one course in probability and statistics (APMA 1650). Computational biology core course requirements: (CSCI1810 and APMA1080). A minimum of one semester of independent study is required (such as BIOL1950 or CSCI1970). In addition, students must complete a research project in their senior year under faculty supervision.

Students must take six courses within one of the following four tracks. Computational Genomics track (for those interested in the development of algorithms and high-quality software [tools and systems] for biological applications); Molecular Modeling track (for those interested in molecular modeling and drug design); Biological Sciences track (for those interested in biological questions); Applied Mathematics and Statistical Genomics track (for those students whose interest focuses on extracting information from genomic and molecular biology data, and modeling the dynamics of these systems). Interested students should consult advisors in the computer science, chemistry, biology and applied mathematics departments respectively.

Mathematics–Computer Science Sc.B.

The standard Sc.B. program in mathematics–computer science has as prerequisites three semesters of calculus, through MATH0180 or equivalent; one semester of linear algebra, i.e. CSCI0530, MATH0520, or MATH0540; either passing or placement out of one of three language courses: FREN0200, GRMN0120, or RUSS0200, or passing an approved writing course.

Required courses: MATH1530; three additional 1000-level mathematics courses; either CSCI0150 and CSCI0160, CSCI0170 and CSCI0180 or CSCI0190; two of CSCI0310, CSCI0320 or CSCI0510; three 1000-level CS courses, of which two form an approved pair (as above under the CS A.B. program); three additional 1000-level courses chosen from mathematics, computer science, applied mathematics, or related areas, and approved by the concentration advisor; a capstone course, as is described previously under the CS Sc.B. program, and should involve an area in which mathematics and computer science are clearly related, e.g. computer graphics analysis of mathematical phenomena, mathematical models used in artificial intelligence, mathematical analysis of algorithms, or theoretical models of computation. The student is expected to produce a final project relating mathematics and computer science.

Independent Study

An undergraduate may undertake a thesis or project, typically in the senior year, under the supervision of a faculty member. It is up to the

interested student to get in touch with a faculty member and convince her or him to supervise the work. Such an effort is more likely to be successful if the student's project relates to the research interests of the faculty member.

A student engaged in a thesis or project under faculty supervision may get academic credit by registering for CSCI1970 ("independent study"). Under ordinary circumstances, a student can receive at most two semesters of credit for independent study. Note that students being paid to do research cannot also receive academic credit for it. Independent study counts toward the advanced computer science courses required for the A.B. and Sc.B. degrees.

Earning Honors

To be considered a candidate for honors in computer science, a student must achieve an outstanding record in the computer science concentration: grades in CS classes should be mostly As or the equivalent. Furthermore, the student must complete a thesis under the supervision of a committee of two faculty members, one of whom must be in the Department of Computer Science, and the committee must deem the thesis worthy of honors.

A student should choose a thesis advisor (who will be the chair of the student's committee) and begin work on the project leading to the thesis no later than the end of the first month of her or his penultimate semester. By the end of the third week of the student's final semester, he or she must have chosen a second committee member, have prepared a one-page proposal for the thesis work, and have presented this proposal to the committee. If the committee approves the proposal and if the student continues to have an appropriately outstanding record in computer science courses, then the student is considered an honors candidate. The committee should notify the Director of Undergraduate Studies of the student's status.

By the day before the Registrar's deadline for honors theses (normally the first Friday of May for students graduating in the spring), the student must submit the completed thesis to the committee and defend it at a public presentation arranged by the Director of Undergraduate Studies. No specific format is required for thesis write-ups. Students doing theses or projects may enroll in CSCI1970 to get academic credit for the research. It is also permissible for a student to do a thesis by expanding a project done in a class. However, the thesis must significantly exceed the requirements of the class (as judged by the professor for that class, who in this case should typically serve as the head of the committee). Honors theses are normally the result of two semesters of work.

GRADUATE PROGRAMS

Brown University offers two graduate degrees in computer science: an Sc.M. for those who wish to improve their professional competence in computer science or to prepare for further graduate study, and a Ph.D. Requirements are outlined below.

Master of Science

The course requirements for the Sc.M. degree consist of a basic and an advanced component. All courses must be at the 1000-level or higher and must be completed with a grade of B or better.

The basic component consists of six courses. At least two courses must be at the 2000-level. Two courses must be computer science courses that form a coherent major; one course must be a computer science course that complements the major; three additional courses must be in computer science or related areas. Examples of majors and complementary courses are available on the course pairs and complements webpage: **www.cs.brown.edu/grad/masters/reqs/ScM_Courses.pdf**. The advanced component requires the student to complete two additional 2000-level courses as part of completing *one* of the four following options:

- A thesis (typically taken as two reading and research courses).
- A project (typically taken as two reading and research courses).
- A project (typically taken as two reading and research courses) and an internship that complements the project
- Two additional 2000-level courses that demonstrate depth in some area of computer science or a closely related discipline. This pair of courses must be approved by the Director of Master's Studies.

This will result in a total of eight courses (two of which may be reading and research).

Concurrent ScB and ScM in Computational Biology

The School of Computing at National University of Singapore and The Department of Computer Science at Brown have established a concurrent Bachelor's and Master's degree program in Computational Biology. After having first completed four years of undergraduate study at National University of Singapore, qualified students will attend Brown University to complete their fifth and

final year of study in computational biology. After the successful completion of requirements set forth by both universities, the students will simultaneously earn both their Sc.B. and Sc.M. degrees.

Doctor of Philosophy

Ph.D. students must satisfy the requirements for admission to candidacy, fulfill major and minor course requirements, do a thesis proposal, complete a thesis that embodies the results of original research and gives evidence of high scholarship, and obtain 24 tuition credits. This is a formal requirement of the graduate school and does not necessarily imply that 24 courses must be taken.

The requirements for admission to candidacy include a course requirement that encourages breadth of study in diverse areas of Computer Science, a programming assignment that tests programming ability, and a research project that tests ability to do research. These requirements should normally be completed by the end of the student's second year. Upon satisfying these requirements, the student will be formally admitted to candidacy for the Ph.D. degree in Computer Science.

The student must complete one major and two minor course requirements. Each requirement is normally met by the satisfactory completion of two approved one-semester courses. The minor requirements are normally one inside and one outside the field of Computer Science. The major and minor course requirements are normally completed by the end of the student's third year in residence.

The student's thesis research is normally done under the supervision of a member of the Computer Science faculty. The thesis is read by the thesis supervisor and two readers appointed by the graduate committee upon the recommendation of the thesis supervisor. It is presented at a meeting open to students, faculty, and the public. Its adequacy is judged by the thesis supervisor, the readers, and the faculty from the Department of Computer Science attending the oral presentation.

Additionally, a new PhD program in Computational Biology and Computer Science, sponsored by the Center for Computational Molecular Biology and the Department of Computer Science, is being planned and is expected to start in the 2009-2010 academic year.

COURSES OFFERED

Primarily for Undergraduates

CSCI0020. Concepts and Challenges of Computer Science

Removes the mystery surrounding computers and the ever-growing digital world. Introduces a range of topics including the Internet and multimedia, along with the underlying digital technology and its relevance to our society. Other topics include artificial intelligence, IT security, ethics and economics of computing, and its pervasiveness in today's world. Analytic skills are developed through HTML and Python assignments. CSCI0020 is a good introduction to a wide range of CS topics that have broad relevance in our society. No prerequisites.

CSCI0040. Introduction to Scientific Computing and Problem Solving

An introduction to computer programming and software design in a high-level language. Emphasizes fundamental techniques and strategies for solving scientific problems with computers. Abstract concepts are illustrated by a wide range of applications from engineering, the sciences and the humanities. Intended primarily for students not concentrating in computer science who want a single application-oriented programming course. No prerequisites.

CSCI0090-A. Building a Web Application (First-year seminar) (Not offered 08/09)

Computer applications involving web-based interfaces interacting with back-end databases are becoming common. These range from e-commerce sites (such as amazon.com) to Banner, the system through which students view course information online at Brown. In this course, we develop a web-based solution for a specific application. In doing so, we study issues related to software engineering, software development and the design, structure and implementation of web-based applications. While the course does not involve substantial programming, students' backgrounds should include some programming and web-page development. Limited enrollment; permission of instructor required.

CSCI0150. Introduction to Object-Oriented Programming and Computer Science

Emphasizes object-oriented design and programming in Java, an effective modern technique for producing modular, reusable, internet-aware programs. Also introduces interactive computer graphics, user

interface design, and some fundamental data structures and algorithms. A sequence of successively more complex graphics programs, including Tetris, helps provide a serious introduction to the field intended for both potential concentrators and those who may take only a single course. No prerequisites.

CSCI0160. Introduction to Algorithms and Data Structures

This course introduces fundamental techniques for problem solving by computer that are relevant to most areas of computer science, both theoretical and applied. Algorithms and data structures for sorting, searching, graph problems and geometric problems are covered. Programming assignments conform to the object-oriented methodology introduced in CSCI0150. Prerequisite: CSCI0150 or written permission.

CSCI0170. Computer Science: An Integrated Introduction

Although students are taught to use programming languages such as Scheme and ML as tools, the goal of CSCI0170 is not merely to teach programming. On the contrary, the goal is to convey to students that computer science is much more than programming! All of the following fundamental computer science techniques are integrated into the course material: algorithms, data structures, analysis, problem solving, abstract reasoning, and collaboration. Concrete examples are drawn from different subareas of computer science: from arbitrary-precision arithmetic, natural language processing, databases and strategic games. Requires no previous programming experience. Indeed, few high school students are exposed to functional programming; even students with previous programming experience often find this course to be an invaluable part of their education.

CSCI0180. Computer Science: An Integrated Introduction

A continuation of CSCI0170. Students learn to program in Java while continuing to develop their algorithmic and analytic skills. Object-oriented design of programs is a principal focus. Examples are drawn from such areas as strategy games, databases, discrete-event simulation, window managers, web client/server programming, route-finding, and data compression. Lab work done with the assistance of TAs. Prerequisite: CSCI0170.

CSCI0190. Programming with Data Structures and Algorithms

This course is a one-semester version of CSCI0150 and CSCI0160 for students with prior programming background. It covers data structures including stacks, lists, queues, trees, heaps, and graphs;

algorithmic methods such as divide and conquer and dynamic programming; and basic analysis techniques. Prerequisite: CSCI0040, score of 4 or 5 on the CS AP exam, or permission of the instructor.

CSCI0220. Introduction to Discrete Structures and Probability

The objective of the course is to place on solid foundations the most common structures of computer science, to illustrate proof techniques, to provide the background for an introductory course in computational theory and to introduce basic concepts of probability theory. It introduces Boolean algebras, logic, set theory, elements of algebraic structures, graph theory, combinatorics and probability. No prerequisites.

CSCI0240. Visual Thinking/Visual Computing *(Not offered 08/09)*

This interdisciplinary course provides a systematic grounding in both technical and theoretical areas of visual research and communication, with a focus on the key role of computer graphics. In addition to reading and writing, assignments include visual projects to be completed with custom-made and commercial software packages.

CSCI0310. Introduction to Computer Systems

This course covers the basic principles behind the organization of modern computers. It starts with machine representation of data types and logic design, then explores the architecture and operations of computer systems, including I/O, pipelining, and memory hierarchies. Uses assembly language as an intermediate abstraction to study introductory operating system and compiler concepts. Prerequisite: CSCI0150, CSCI0180 or CSCI0190.

CSCI0320. Introduction to Software Engineering

Advanced programming techniques including Java, threads, web applications, user interfaces and XML. Covers software design including object-oriented design, systems design, web application design and user interface design. Software engineering including modeling, analysis, testing, debugger reuse, the software life cycle, tools, and project management. Prerequisite: CSCI0160, CSCI0180 or CSCI0190; CSCI0220 is recommended.

CSCI0510. Models of Computation

This course introduces basic models of computation including languages, finite-state automata and Turing machines. Proves fundamental limits on computation (incomputability, the halting problem). Provides the tools to compare the hardness of

computational problems (reductions). Introduces computational complexity classes (P, NP, PSPACE and others). Prerequisite: CSCI0220.

CSCI0530. Directions: The Matrix in Computer Science

The aim of this course is to provide students interested in computer science an introduction to vectors and matrices and their use in modeling and data analysis. The course will be driven by applications from areas chosen from among: combinatorial optimization, computer vision, cryptography, game theory, graphics, information retrieval and web search, machine learning and scientific visualization. For example, students will learn Google's PageRank method for ranking web pages. This course satisfies the linear algebra requirement for the Computer Science Sc.B. Prerequisite: no formal prerequisites but students are expected to be comfortable with mathematics and computing.

For Undergraduates and Graduates

CSCI1230. Introduction to Computer Graphics

This course offers an in-depth exploration of fundamental concepts in 2D and 3D computer graphics. It introduces 2D raster graphics techniques, including scan conversion, simple image processing, interaction techniques and user interface design. The bulk of the course is devoted to 3D modeling, geometric transformations, and 3D viewing and rendering. A sequence of assignments in C++ culminates in a simple geometric modeler and ray tracer. Prerequisite: CSCI0160 or CSCI0180; CSCI0320 is strongly recommended. Students who do not know C++ should take a mini-course on it offered during the first week of the semester. One of CSCI0530 or MATH0520 is strongly recommended.

CSCI1250. Introduction to Computer Animation

Introduction to 3D computer animation production including story writing, production planning, modeling, shading, animation, lighting and compositing. The first part of the course leads students through a series of exercises that build on each other to teach basic skills in 2D and 3D animation. At each step, student work is evaluated for expressiveness, technical correctness and aesthetic qualities. Students then work in groups of two to four to create a polished short animation. The class format includes lecture, demonstration, and viewing animations. The emphasis is on in-class critique of ongoing work, which is essential to the cycle of visually evaluating work in progress, determining improvements, and implementing them for further evaluation. Prerequisite: consent of instructor.

CSCI1260. Introductory Compiler Construction *(Not offered 08/09)*

Lexical analysis, syntactic analysis, semantic analysis, code generation, code optimization, translator writing systems. Prerequisites: CSCI0220 and CSCI0320; CSCI0510 is recommended.

CSCI1270. Database Management Systems

Introduction to database structure, organization, languages and implementation. Relational and object-relational models. Query languages, query processing, query optimization, normalization, file structures, concurrency control and recovery algorithms, and distributed databases. Studies of actual systems. While database management system usage is covered, emphasis is on the systems-building aspects of these large, complex systems. We also relate the material to modern applications such as the web. Prerequisites: CSCI0220 and CSCI0310.

CSCI1280. Intermediate 3D Computer Animation

This course continues work begun in CCSCI1250 with deeper exploration of the core technical and artistic aspects of 3D computer animation. In the first portion of the course, students complete a series of tutorials and animation assignments in which they learn more complex modeling, character rigging, animation, shading, and lighting techniques. In the second portion of the course, students independently explore one area in more depth and then finally create portfolio-quality demonstrations alone or in pairs. We read and discuss technical texts as well as works on artistic motivation and view related animated films. The emphasis of class time will be on critiquing ongoing student work. Prerequisite: CSCI1250. Enrollment limited to 20. Written permission required.

CSCI1340. Innovating Game Development *(Not Offered 08/09)*

What technologies will shape the next generation of video games? This project-centered course focuses on computational innovations for game development. Students examine innovative game technology through case studies of existing games and talks by industrial and academic game professionals. In teams, students propose and implement a project demonstrating a novel technology for gaming. Recommended: strong computational or engineering background.

CSCI1370. Virtual Reality Design for Science

Explores the visual and human-computer interaction design process for scientific applications in Brown’s immersive virtual reality Cave.

Joint with RISD. Computer science and design students learn how to work effectively together; study the process of design, learn about scientific problems, create designs for scientific applications; critique, evaluate, realize and iterate designs, and demonstrate final projects. Prerequisite: permission of the instructor.

CSCI1380. Distributed Computer Systems

Explores the fundamental principles and practice underlying networked information systems. We first cover basic distributed computing mechanisms (e.g., naming, replication, fault tolerance, security) and enabling middleware technologies. We then discuss how these mechanisms and technologies fit together to realize distributed databases and file systems, web-based and mobile information systems. Prerequisite: CSCI0320.

CSCI1410. Introduction to Artificial Intelligence

Theoretical and practical approaches to designing intelligent systems. Example tasks range from game playing to hardware verification. Core topics include knowledge representation, search and optimization and automated reasoning. Application areas include natural language processing, machine vision, machine learning, and robotics. Prerequisites: CSCI0160, CSCI0180 or CSCI0190. Strongly recommended: CSCI0220.

CSCI1430. Introduction to Computer Vision

How can computers understand the visual world of humans? This course treats vision as a process of inference from noisy and uncertain data and emphasizes probabilistic and statistical approaches. Topics may include perception of 3D scene structure from stereo, motion, and shading; image filtering, smoothing, edge detection; segmentation and grouping; texture analysis; learning, recognition and search; tracking and motion estimation. Prerequisites: basic linear algebra, basic calculus and exposure to probability.

CSCI1480. Building Intelligent Robots

How do robots function autonomously in dynamic, unpredictable environments? This course focuses on programming mobile robots, such as the iRobot Roomba, to perceive and act autonomously in real-world environments. The major paradigms for autonomous control and robot perception are examined and compared with robotic notions in science fiction. Prerequisite: CSCI0150, CSCI0170 or CSCI0190. Recommended: CSCI1410 or CSCI1230.

CSCI1490. Introduction to Combinatorial Optimization

This course covers the algorithmic aspects of optimizing decisions in fully observable, non-changing environments. Students are introduced to state-of-the-art optimization methods such as linear programming, integer programming, local search, and constraint programming. Prerequisite: CSCI0160, CSCI0180, or CSCI0190. Strongly recommended: CSCI0310 and CSCI0510; APMA0340, MATH0520 or MATH0540.

CSCI1510. Introduction to Cryptography and Computer Security

This course studies the tools for guaranteeing safe communication and computation in an adversarial setting. We develop notions of security and give provably secure constructions for such cryptographic objects as cryptosystems, signature schemes and pseudorandom generators. We also review the principles of secure system design. Prerequisites: CSCI0220 and CSCI0510.

CSCI1550. Probabilistic Methods in Computer Science

Introduction to the applications of probability theory in computer science, in particular to randomized algorithms and probabilistic analysis of algorithms. The course introduces basic probability theory and presents applications of randomized and probabilistic analysis techniques in areas such as combinatorial optimization, data structures, communication and parallel computation. No prior knowledge of probability theory is assumed. Prerequisite: CSCI0220 or equivalent; CSCI1570 is recommended but not required.

CSCI1570. Design and Analysis of Algorithms

A single algorithmic improvement can have a greater impact on our ability to solve a problem than ten years of incremental improvements in CPU speed. We study techniques for designing and analyzing algorithms. Typical problem areas addressed include numerical computing, hashing, searching, dynamic programming, graph algorithms, network flow, and string parsing and matching. Prerequisites: CSCI0160, CSCI0180, or CSCI0190, and CCI0220.

CSCI1590. Introduction to Computational Complexity *(Not Offered 08/09)*

Introduction to the following topics: serial and parallel models of computation, serial and parallel space and time complexity classes, circuit complexity measures, and space-time, area-time and I/O-time tradeoffs. Prerequisite: CSCI0510.

CSCI1600. Introduction to Embedded Real-Time Software *(Not Offered 08/09)*

Comprehensive introduction to the design and implementation of software for programmable embedded computing systems, those enclosed in devices such as cellular phones, game consoles, and car engines. Includes the overall embedded real-time software design and development processes, as well as aspects of embedded hardware and real-time, small-footprint operating systems. Major project component. Prerequisite: CSCI0320. Enrollment limited to 30.

CSCI1610. Building High-Performance Servers *(Not offered 08/09)*

In-depth study of modern server design. Considers architectures for building high-performance, robust, scalable and secure servers. We consider all aspects of “mission-critical” servers. Topics include threaded and non-blocking programming paradigms, high-performance I/O (network and disk), secure programming techniques, database access, performance profiling, security, and redundancy. Teams will build significant projects. Prerequisite: CSCI0320. Recommended: CSCI1670 and CSCI1680.

CSCI1660. Introduction to Computer Systems Security

This course teaches principles of computer security from an applied viewpoint and provides hands-on experience on security threats and countermeasures. Topics include code execution vulnerabilities (buffer overflow, sandboxing, mobile code), malware (trojans, viruses, and worms), access control (users, roles, policies), cryptosystems (hashing, signatures, certificates), network security (firewalls, TLS, intrusion detection, VPN), and human and social issues (usability, social engineering, digital rights management). Prerequisite: CSCI0160, CSCI0180, or CSCI0190.

CSCI1670. Operating Systems

The basic principles of operating systems. Part I: fundamental concepts including multithreaded programming and concurrency, dynamic storage allocation and liberation, linkers and loaders, file systems and virtual memory. Covers actual systems including Solaris, Linux and Windows XP. Part II: operating-system support for distributed systems, including computer communication protocols, remote procedure call protocols, computer security, and distributed file systems. Prerequisite: CSCI0320.

CSCI1680. Computer Networks

Covers the technologies supporting the Internet, from Ethernet and Wi-Fi through the routing protocols that govern the flow of traffic

and the web technologies that are generating most of it. A major concern is understanding the protocols used on the Internet: how they work, their shortcomings, what the issues are, and what improvements are on the horizon. Prerequisite: CSCI0320 or consent of instructor.

CSCI1690. Operating Systems Laboratory

Half-credit course intended to be taken with CSCI1670. Students individually write a simple operating system in C. Reinforces the concepts learned in CSCI1670 and provides valuable experience in systems programming. Corequisite: CSCI1670.

CSCI1730. Introduction to Programming Languages

This course explores the principles of modern programming languages by implementing them. Examines linguistic features, especially control operators such as first-class functions, exceptions and continuations. This leads to a study of data and their types, including polymorphism, type inference and type soundness. The course concludes by examining compiler and run-time system topics such as continuation-passing style and garbage collection. Prerequisite: CSCI0160, CSCI0180 or CSCI0190. Recommended: CSCI0220, CSCI0310, and CSCI0510.

CSCI1760. Introduction to Multiprocessor Synchronization

This course examines the theory and practice of multiprocessor synchronization. Subjects covered include multiprocessor architecture, mutual exclusion, wait-free and lock-free synchronization, spin locks, monitors, load balancing, concurrent data structures, and transactional synchronization.

CSCI1780. Parallel and Distributed Computing *(Not offered 08/09)*

This course covers the practical aspects involved in designing, writing, tuning and debugging software designed to run on parallel and distributed systems. Topics might include client-server computation, threads, networks of workstations, message passing, shared memory, partitioning strategies, load-balancing algorithms, remote procedure call, and synchronization techniques. Prerequisites: CSCI0220 and CSCI0320. Recommended: CSCI0510.

CSCI1810. Computational Molecular Biology

Processing molecular biology data (DNA, RNA, proteins) has become central to biological research and a challenging area for computer science research. Important objectives are molecular sequence analysis, recognition of genes and regulatory elements, molecular

evolution, protein structure, comparative genomics. This course models the underlying biology in the terms of computer science and presents the most significant algorithms of molecular computational biology. Prerequisites: CSCI0160, CSCI0180 or CSCI0190, and CSCI0220, or consent of instructor.

CSCI1900. Software System Design

Students identify, design, and implement significant software applications and learn and practice techniques of project management, requirements, specification, analysis, design, coding, documentation, testing, maintenance and communication. Prerequisite: CSCI0320.

CSCI1950 Special Topics in Computer Science

Courses in various branches of computer science, including those listed below. Specific courses may be added at the beginning of each semester.

CSCI1950-C. Advanced Programming for Digital Art and Literature

This workshop explores advanced tools and techniques for the creation of innovative and expressive works of digital art. Lectures address the application of best practices from the software-design community to the context of digital media. In the first section of the course, students exercise their skills with new techniques (integrated development tools, agile and object-oriented programming, rapid debugging and prototyping, etc.) on a range of “mini-projects,” specifically the analysis, generation and digital presentation of computationally augmented literary texts. Assignments include web-data parsing, speech synthesis, context-free grammars, and statistical generation techniques. During the second half of the course, students focus on a larger work of their own design, participating in regular critical reviews throughout the development cycle. Although assignments focus on digital literature, wide-ranging media is explored including sound, image, video, 3D, and installation. Although there are no formal prerequisites, familiarity with at least one modern programming language is highly recommended.

CSCI1950-L. Algorithmic Foundations of Computational Biology *(Not Offered 08/09)*

This course is devoted to computational and statistical methods as well as software tools for DNA, RNA, and protein sequence analysis. The focus is on understanding the algorithmic and mathematical foundations of the methods, the design of associated genomics tools, as well as on their applications. The course is open to computer and

mathematical sciences students as well as biological and medical students.

CSCI1950-Z. Computational Methods for Biology

This course will introduce algorithms from machine learning and combinatorial optimization with a focus on their application to biological data. Topics will include problems in phylogenetic inference, population genetics, and biological interaction networks.

CSCI1970. Senior Seminar

Independent study in various branches of computer science, supervised by the faculty.

APMA1710. Information Theory *(Cross-listed course)*

Information theory is the study of the fundamental limits of information transmission and storage. This course offers a broad introduction to information theory and its real-world applications: Entropy and information, lossless data, compression, communication in the presence of noise, capacity, channel coding; source-channel separation; lossy data compression. Prerequisite: calculus, linear algebra, APMA1650 or equivalent.

COGS1360. Introduction to Computational Linguistics *(Cross-listed course)*

Investigates computational models of natural language comprehension and production. Focuses primarily on syntactic parsing (i.e., algorithms that determine the syntactic structure and the “logical form” of a sentence) and the relationship between different linguistic theories and algorithms that can implement them. Recommended background: CSCI0510 or equivalent, and either COGS1110 or COGS1310, or permission of the instructor.

COGS1680. Introduction to Machine Learning *(Cross-listed course)*

A systematic introduction to machine learning, covering theoretical as well as practical aspects of the use of statistical methods in artificial intelligence. Topics include linear models, decision trees, neural networks, support vector machines, regularization theory, graphical models and reinforcement learning. Application examples are taken from areas such as information retrieval, natural language processing, computer vision and computational biology. Prerequisites: CSCI0160, CSCI0180 or CSCI0190, and CSCI0220. Familiarity with probability and linear algebra are helpful.

Primarily for Graduates

CSCI2240. Interactive Computer Graphics

Important current topics in computer graphics. Course includes reading and discussing current research papers, multiple assignments and preliminary projects in which students implement recent papers, and a demanding final integrative project done in small groups. Prerequisites: CSCI1230 and CSCI0320.

CSCI2270. Topics in Database Management

In-depth treatment of advanced issues in database management systems. The focus is on current research. Topics vary from year to year and may include distributed databases, query processing, mobile data management, data warehousing, and web-based data management. Prerequisite: CSCI1270.

CSCI2310. Human Factors and User Interface Design *(Not Offered 08/09)*

Covers current research issues in the implementation, evaluation and design of user interfaces, while also providing a basic background in user-interface evaluation, programming, tools and techniques. A possible topic is programming and designing device-independent interfaces. Previous topics have included the development of pervasive Internet-based interfaces and software visualization. Prerequisite: consent of instructor.

CSCI2330. Programming Environments *(Not offered 08/09)*

This course covers programming tools, control and data integration, software understanding and debugging, environments for parallel and distributed programming, reverse engineering, configuration management and version control and debugging. Emphasis on current research areas. Prerequisite: consent of instructor.

CSCI2340. Software Engineering

Topics in design, specification, construction and validation of programs, focusing on tools to support each of these stages. We pay special attention to concerns raised by the properties of modern software systems including distribution, security, component-based decomposition and implicit control. Prerequisite: CSCI1900 or other upper-level systems coursework.

CSCI2370. Interdisciplinary Scientific Visualization *(Not Offered 08/09)*

The solution of scientific problems using computer graphics and visualization. Working in small multidisciplinary groups, students identify scientific problems, propose solutions involving computational modeling and visualization, design and implement the solutions, apply them to the problems and evaluate their success. Examples include: interactive software systems, immersive Cave applications, new applications of existing visualization methods. Prerequisites: all: programming experience; CS students: graphics experience; others: problem ideas. Interested students should contact the instructor.

CSCI2410. Statistical Models in Natural Language Understanding *(Not Offered 08/09)*

This course covers various topics in computer understanding of natural language, primarily from a statistical point of view. Topics include: hidden Markov models, word-tagging models, probabilistic context-free grammars, syntactic disambiguation, semantic word clustering, word-sense disambiguation, machine translation and lexical semantics. Prerequisite: CSCI1410.

CSCI2440. Game-Theoretic Artificial Intelligence *(Not Offered 08/09)*

This course surveys recent developments in the emerging area of game-theoretic artificial intelligence, which incorporates fundamental principles of game theory into AI. Research in this area is motivated by game-theoretic applications, such as auction design and voting, as well as AI application areas, such as multi-agent systems. Students will conduct theoretical, empirical, and experimental investigations, asking fundamental questions such as: can computational agents learn to play game-theoretic equilibria? Prerequisite: Consent of the instructor.

CSCI2500. Topics in Advanced Algorithms

CSCI2500-A. Advanced Algorithms

Typically, an algorithm solves one problem, whereas a well-designed data structure can help implement algorithms for a wide variety of problems. We will study the design, analysis and implementation of advanced data structures. Focus is on data structures that are fast, both theoretically and empirically. Prerequisite: CSCI1570 or the equivalent.

CSCI2500-B. Optimization Algorithms for Planar Graphs *(Not offered 08/09)*

Planar graphs arise in applications such as road map navigation and logistics, graph drawing, and image processing. We will study graph algorithms and data structures that exploit planarity. Our focus will be on recent research results in optimization. Prerequisite: CSCI1570 or the equivalent.

CSCI2510. Approximation Algorithms

Approximation algorithms deal with NP-hard combinatorial optimization problems by efficiently constructing a suboptimal solution with some specified quality guarantees. We study techniques such as linear programming and semi-definite programming relaxations, and apply them to problems such as facility location, scheduling, bin packing, maximum satisfiability or vertex cover. Prerequisite: CSCI1490 or CSCI1570.

CSCI2520. Computational Geometry *(Not offered 08/09)*

Algorithms and data structures for fundamental geometric problems in two and three dimensions. Topics include point location, range searching, convex hull, intersection, Voronoi diagrams and graph drawing. Applications to computer graphics, circuit layout, information visualization and computer-aided design are also discussed. Prerequisite: CSCI1570 or written permission.

CSCI2531. Internet and Web Algorithms

This advanced graduate course/seminar focuses on the mathematical foundations of algorithms for handling large amounts of data over networks. We will read and discuss recent papers in information retrieval, search engines, link analysis, probabilistic modeling of the web and social networks, and more. Prerequisite: CSCI1550 and CSCI1570, or equivalent courses.

CSCI2540. Advanced Probabilistic Methods in Computer Science *(Not offered 08/09)*

Advanced topics in applications of probabilistic methods in design and analysis of algorithms, in particular to randomized algorithms and probabilistic analysis of algorithms. Topics include the Markov chain Monte Carlo method, martingales, entropy as a measure for information and randomness, and more. Prerequisite: CSCI1550. Recommended but not required: CSCI1570.

CSCI2550. Parallel Computation: Models, Algorithms, Limits *(Not offered 08/09)*

The theoretical foundations of parallel algorithmics. Analysis of the most important models of parallel computation, such as directed acyclic computation graphs, shared memory and networks, standard data-exchange schemes (common address space and message-passing). Algorithmic techniques with numerous examples are cast mostly in the data-parallel framework. Finally, limitations to parallelizability (P-completeness) are analyzed. Course content is likely to change as technology evolves. Written permission needed for undergraduates.

CSCI2560. Applied Theory of Computation *(Not offered 08/09)*

Advanced topics in theoretical computer science are chosen from the following list: parallel computation, time and space complexity classes, circuit complexity, I/O complexity, VLSI computation and nanocomputing.

CSCI2570. Introduction to Nanocomputing

Nanoscale technologies employing materials whose smallest dimension is on the order of a few nanometers are expected to replace lithography in the design of chips. We introduce computational nanotechnologies and explore problems presented by their stochastic nature. Nanotechnologies based on the use of DNA and semiconducting materials will be explored. Prerequisite: CSCI0510.

CSCI2580. Solving Hard Problems in Combinatorial Optimization: Theory and Systems *(Not offered 08/09)*

Addresses not only the theory of combinatorial optimization but also how it is embodied in practical (industrial and logistical) systems. Explores some of the issues and obstacles encountered in implementing such systems. Emphasizes the wide variety of techniques and methodologies available, including integer programming, local search, constraint programming, and approximation algorithms. Problems addressed may include: scheduling, coloring, traveling salesman and resource allocation. Prerequisites: CSCI0320 and basic knowledge of linear algebra.

CSCI2590. Advanced Topics in Cryptography *(Not offered 08/09)*

Seminar-style course on advanced topics in cryptography. Example topics are zero-knowledge proofs, multi-party computation,

extractors in cryptography, universal composability, anonymous credentials and e-cash, interplay of cryptography and game theory. May be repeated for credit. Prerequisite: CSCI1510 or permission of the instructor.

CSCI2730. Programming Language Theory *(Not offered 08/09)*

Theoretical models for the semantics of programming languages and the verification of programs. Topics include operational semantics, denotational semantics, type theory and static analyses. Prerequisite: CSCI1730 or permission of the instructor.

CSCI2750. Topics in Parallel and Distributed Computing

CSCI2750 considers an advanced topic (to be determined) in distributed computing. May be repeated for credit. Written permission needed for undergraduates.

CSCI2950. Special Topics in Computer Science

Graduate courses in various branches of Computer Science, including those listed below. Specific courses may be added at the beginning of each semester.

CSCI2950-C. Topics in Computational Biology

This course will investigate active and emerging research areas in computational biology. Topics include cancer genomics; genome rearrangements and assembly; and protein and regulatory interaction networks. The course will be a mixture of lectures and student presentations of recent conference and journal papers.

CSCI2950-E. Stochastic Optimization *(Not offered 08/09)*

This advanced graduate course/seminar focuses on optimization under uncertainty, or optimization problems where some of the constraints include random (stochastic) components. Most practical optimization problems are stochastic (subject to future market conditions, weather, faults, etc.), and there has been substantial research (both theoretical and experimental) in efficient solution for such problems. We discuss some of the recent work in this area.

CSCI2950-G. Large-Scale Networked Systems

Explores widely distributed systems that take advantage of resources throughout the Internet. These systems leverage their large size and geographic diversity to provide bandwidth scalability, rapid responses, fault-tolerance, high-availability and diverse data collection. Topics include overlay networks, peer-to-peer systems, content

distribution networks, distributed file systems and wide-scale measurement systems.

CSCI2950-I. Computational Models of the Neocortex *(Not offered 08/09)*

This course addresses the problem of modeling the perceptual neocortex using probabilistic graphical models, including Bayesian and Markov networks, and extensions to model time and change such as hidden Markov models and dynamic Bayesian networks. The emphasis is on problems of learning, inference and attention. Sources include the literature in computational and cognitive neuroscience, machine learning, and other fields that bear on how biological and engineered systems make sense of the world. Prerequisites: basic probability theory, algorithms and statistics.

CSCI2950-J. Cognition, Human-Computer Interaction and Visual Analysis

In this graduate seminar, we will learn about models of human cognition and perception and explore potential implications of the models on how computers and humans can interact effectively when performing scientific analyses. Participants will be responsible for reading assigned materials, taking turns guiding discussions of the readings, and preparing a final paper and presentation. It is recommended that participants have some background in at least one of these areas of study.

CSCI2950-L. Algorithmic Foundations of Computational Biology II *(Not offered in 08/09)*

This course focuses on computational methods for genome-wide disease association and the HapMap Project. The following topics will be covered: basic models of population genetics, SNPs and haplotypes analysis, linkage disequilibrium (LD), LD measures, LD theory and genetic determinants of disease, empirical state of LD patterns across populations, SNP challenges to genome assembly, haplotype blocks, and block-free methods, haplotype phasing (expectation maximization (EM) algorithms, Clark algorithm, parsimony algorithms, Bayesian methods, perfect phylogeny algorithms), proofs of NP-completeness for the haplotype phasing problem (EM, parsimony, Clark-type parsimony), SNP selection and the minimum informative subset, hypothesis testing and associations, disease associations tests of significance, Sir R.A. Fisher and the likelihood, genome-wide association studies for: multiple sclerosis, cardiovascular disease, diabetes, and cancer, uses and misuses of tests of statistical significance, sample size and power calculations, haplotypes in association analysis, common disease common variant

hypothesis, coalescent theory and the ancestor recombination graph problem.

CSCI2950-O. Topics in Brain-Computer Interfaces *(Not offered 08/09)*

Introduces the mathematical and computational foundations of brain-computer interfaces. Statistical learning, Bayesian inference, dimensionality reduction, information theory and other topics are presented in the context of brain interfaces based on neural implants. Prerequisites: Knowledge of probability, statistics and linear algebra (e.g., CSCI1550, APMA0410, APMA1650, or APMA1690). Enrollment limited to 20 students.

CSCI2950-Q. Topics in Computer Vision

This course will cover current topics in computer vision with an emphasis on motion estimation, learning methods and probabilistic models. Readings will be from recent research papers. Computational techniques may include expectation maximization, hidden Markov models and belief propagation. Applications to tracking, recognition, image enhancement and human motion analysis will be considered.

CSCI2950-T. Topics in Distributed Databases and Systems

Explores data and resource management issues that arise in the design, implementation and deployment of networked systems by covering the state of the art in research and industry. Topics include sensor networks and Internet-scale information systems and services. Prerequisite: CSCI1380 or consent of the instructor.

CSCI2950-X. Topics in Programming Languages and Systems *(Not offered 08/09)*

Examines contemporary research topics in software construction from the perspectives of programming languages, software engineering and computer-aided verification. The primary goals are to understand which theory applies to which problems and to convert that theory into tools. Prerequisite: CSCI1730 or written permission of instructor.

CSCI2950-Y. Theorem Proving *(Not offered 08/09)*

This course explores computer-assisted theorem proving with the Coq Proof Assistant. The course will teach students to formally specify software and model mathematical theories. We will then study techniques for mechanically proving theorems. Prerequisite: CSCI1730 or permission of the instructor.

CSCI2950-Z. Robot Learning and Autonomy *(Not offered 08/09)*

This seminar course covers current research topics related to perceiving and acting in the real world. These topics will be pursued through independent reading, class discussion, and project implementations. Papers covered will be drawn from robotics, computer vision, animation, machine learning, and neuroscience. Special emphasis will be given to developing autonomous control from human performance. No prerequisites.

ENGN2911-T. 3D Photography and Geometry Processing *(Cross-listed course)*

In 3D photography, cameras and lights are used to capture the shape and appearance of 3D objects represented as graphical models for applications such as computer animation, game development, electronic commerce, heritage preservation, reverse engineering, and virtual reality. This course covers 3D capture techniques and systems, surface representations and data structures, as well as methods to smooth, denoise, edit, compress, transmit, simplify, and optimize very large polygonal models.

CSCI2980. Reading and Research

FACILITIES

The Department of Computer Science provides leading-edge computing technology to all its faculty and students. We have over 500 desktop systems running Linux or Windows XP.

Most of these are custom systems configured and assembled by the department's technical staff. Components include AMD Athlon Dual- or Quad-core processors with 2GB or 4GB of memory and dual 19" or single 24" flat-panel monitors. These systems are connected to the department's 1Gb/s switched Ethernet network with access to both Internet1 and Internet2 via the University's fiber-optic backbone. An 802.11g (54Mb/s) wireless network is accessible throughout the department.

The department has two electronic classrooms. One, a banked auditorium, holds seventy-three systems running Linux. This room serves as the primary computer facility for undergraduate computer science students. The other contains twenty-two systems running Microsoft Windows. The layout of this space makes it an ideal room for sections, seminars, and interactive learning. Six research labs further enrich the environment with specialized hardware and advanced workstations from a variety of vendors.

Desktop and research systems are supported by a data center with fully redundant servers that offer a wide range of services. Central file storage is provided by a clustered pair of Network Appliance filers hosting of over 16TB of RAID-6 disks and an array of Linux servers hosting nearly 9TB of RAID-5 disks. Computational servers in a Sun Grid Engine cluster, all running Linux, include 65 dual-processor, dual-core machines each with 8GB of memory, four quad-processor, dual- or quad-core systems with 16GB to 32GB each, and a number of others for a total of 91 machines with 332 cores in all. Additional compute cycles are available to the user community through a distributed processing system that allows batch jobs to make use of spare processor cycles on departmental desktop systems.

The Department is located in the Thomas J. Watson Sr. Center for Information Technology, a bright, open and inviting space. The undergraduate workstation labs share the first floor with other computer-equipped classrooms and clusters managed by the University's computer-support personnel.

RESEARCH AREAS

Artificial Intelligence

(Michael Black, Eugene Charniak, Amy Greenwald, Chad Jenkins, Mark Johnson, Meinolf Sellmann, Pascal Van Hentenryck)

Artificial intelligence is concerned with elucidating the principles behind intelligent behavior by creating artifacts (computer programs) that embody such principles. AI researchers at Brown tend to see probability and statistics as the primary mathematics in which these principles are expressed, while recognizing that various cognitive areas have quite different specifics. We also have a bias towards seeing the specific problems in which we are interested (e.g., vision, language, temporal reasoning, economic behavior, brain implants) as special instances of machine learning — that is, we believe that in a surprisingly wide variety of cases, the best way to get a program to solve some problem is to have it *learn* to solve the problem. Thus, AI researchers at Brown share not just a common set of problems but also a substantial set of tools with which to approach them.

Combinatorial Optimization

(Philip Klein, Claire Mathieu, Benjamin Raphael, Meinolf Sellmann, Eli Upfal, Pascal Van Hentenryck)

Combinatorial optimization involves finding the best (e.g. lowest cost, most valuable, smallest, largest) among a huge but finite set of candidates. Examples include job-shop scheduling and the traveling salesman problem. Research at Brown focuses on three areas:

- Solution methods for traditional optimization problems, including methods for finding exact solutions (e.g. constraint and integer programming, dynamic programming), approximation algorithms that are guaranteed to find solutions nearly as good as the best, and heuristic methods that find good solutions for most but not all possible instances.
- Stochastic and on-line optimization problems in which the data are uncertain and/or not known *a priori.*
- Modeling tools that streamline the development of solution methods.

Computational Biology

(Sorin Istrail, David Laidlaw, Franco Preparata, Benjamin Raphael, Eli Upfal, Pascal Van Hentenryck)

Computational biology is a vigorous, emerging discipline at the interface of computer science and life science and is at the core of bioinformatics, which is concerned with the acquisition, analysis, and storage of biological information. Computational biology is the development of algorithms and computer programs central to this effort, with special emphasis on nucleic acid and protein sequence applications. Subjects of active research include DNA mapping, sequence alignment, data mining in biological databases, phylogeny, spatial structures, and functional genomics and proteomics.

Research towards the discovery of “complexity laws” governing systems in biology, physics, chemistry and economics provides a new way of thinking about the behavior of enormous systems and their interacting units. Computer science provides methods for the identification of the computational complexity roots of qualitatively similar “complex systems phenomena” occurring in physics, biology, chemistry and economics.

Computational Geometry

(Franco Preparata, Roberto Tamassia)

Theoretical research in this area addresses data structures and algorithms for fundamental geometric problems, robust geometric prim-

itives, computational metrology and tolerancing, computational topology, geometric graph theory and graph-drawing algorithms. Experimental work includes a system for matching trajectories of objects moving in space and a method for the estimation of the location of sensor nodes using power measurements of the signals received from beacon nodes.

Computational Neuroscience

(Michael Black, Tom Dean, Chad Jenkins, David Laidlaw)

As part of a larger interdisciplinary brain sciences initiative at Brown, neuroinformatics combines neuroscience and informatics in order to advance our understanding of how the brain works. Methods from various branches of computer science are used to visualize, model, and simulate neural processes. Areas of particular interests include human-computer interfaces, computational and statistical models of sensory processing and motor control, adaptivity and learning in the brain, and the use of computer graphics to enhance brain-imaging technology.

Computer Graphics

(John Hughes, Chad Jenkins, David Laidlaw, Barbara Meier, Gabriel Taubin, Andy van Dam)

The long-term research goal of the Brown University Graphics Group is to develop human-centered, powerful, and interactive 2D and 3D graphics tools for modeling, scientific visualization, and education, as well as to study the mathematical foundations of computer graphics.

The Visualization Research Group is part of the Brown University Graphics Group. An interdisciplinary team of people from a number of Brown departments develops robust and effective computer science and visualization tools and techniques for solving a range of problems and phenomena from science as well as the arts and humanities. Collaborative work with colleagues in these areas guides the research and provides a mechanism for evaluating the usefulness and robustness of results.

Computer Vision

(Michael Black, John Hughes, Chad Jenkins, Erik Sudderth, Gabriel Taubin)

Problems in vision reside at the interface of the physical world, computational machines, and the human brain. Our computational models of the physical world are, by necessity, incomplete, which means that when we observe or act upon the world we do so with inaccurate, uncertain, and ambiguous information.

Research on vision at Brown focuses on this problem of forming and testing hypotheses about a world of which we are uncertain, often from data that are inaccurate, noisy, or inconsistent. Our research includes video motion analysis, tracking, event recognition, object detection and recognition, segmentation and visual scene analysis, human motion understanding, and applications of vision in computer graphics. Our work spans many disciplines and involves collaborations with researchers in engineering, neuroscience, cognitive and linguistic sciences, mathematics, applied mathematics, and biomechanics. Our facilities include multiple Vicon motion-capture systems as well as multi-camera video capture systems.

Cryptography

(Anna Lysyanskaya, Roberto Tamassia)

Cryptography is about solving impossible problems. For example, consider the problem of accurate, verifiable and private electronic voting. At first glance, it seems that it is impossible to have a protocol for voting that would be guaranteed to tally the votes correctly, do so in public so everyone could see it was tallied correctly, and yet, still, somehow hide all information about how each individual cast his or her vote. Yet, it turns out that it is possible, and what's more, there are practical and provably secure algorithms for it, and much, much more!

The cryptography group at Brown successfully works on similarly impossible problems. For example, we work on practical and provably secure solutions to the problem of authentication without identification, where one can prove that one is an authorized user without revealing one's identity. We are developing extremely efficient algorithms for authenticating high volumes of data in non-trusted distributed environments. While firmly rooted in theory, our research is intended to be useful in practice, and we collaborate extensively with both theoreticians and systems-builders.

Database Systems

(Ugur Cetintemel, Stanley Zdonik)

Historically, we have been leaders in object-oriented database systems and advanced query languages and query processing. We have a strong interest in data management for networked systems, including the intelligent use of resources for broadcast and dissemination-based systems. We focus on the emerging area of data-stream management systems (DSMS) with an emphasis on real-time processing, quality of service maintenance, and approximate answers. Our work also involves developing data-centric abstractions

and systems for high-level tasking of wireless sensor-actuator networks.

We are interested in scaling database systems and data warehouses through the use of massively parallel grids and virtual environments or clouds. We have also been working on various automatic physical database design algorithms that can make effective use of these complex new environments. An important application for this kind of service is scientific data management, and we are developing special purpose architectures for this area. Scientific data is by its nature naturally uncertain and query processing techniques must be able to deal with this.

Design and Analysis of Algorithms

(Sorin Istrail, Philip Klein, Claire Mathieu, Franco Preparata, Benjamin Raphael, John Savage, Meinolf Sellmann, Roberto Tamassia, Eli Upfal)

We study applications of probability theory to the design and analysis of algorithms. Randomness comes up in two aspects of the study of algorithms: randomized algorithms and probabilistic analysis of algorithms. In randomized algorithms, we are particularly interested in algorithms for communication and distributed applications, as well as online combinatorial problems. In probabilistic analysis, our work focuses on the long-term steady-state performance of dynamic processes.

We also study combinatorial algorithms, which treat computations on finite, discrete mathematical structures. Topics include searching, sorting, and enumeration, data structures, graph algorithms, external-memory algorithms, combinatorial optimization, approximation algorithms for NP-hard problems, online algorithms, and algorithm engineering.

Educational Technology

(Shriram Krishnamurthi, Roberto Tamassia, Andy van Dam)

The department has a strong record of accomplishment of research in interactive educational technology and tools. We continue to work on educational applets for teaching concepts in computer science. Furthermore, most of the applications we do in our research in pen-centric and multi-touch computing have strong educational applications. We have developed a secure laboratory environment allowing the students of our computer security course to safely experiment with virus propagation and other attacks to computer networks. In addition, we are ongoing participants in the design of DrScheme, one of the leading teaching-motivated and student-friendly programming environments.

Geometric Shape Representation and Interfaces for Modeling

(John Hughes, Gabriel Taubin)

We develop interfaces for naturally creating and editing 3D shapes and the associated shape representations that enable such interfaces. These interfaces include sketch-based interfaces for both standard geometric models (cuboids, generalized cylinders, etc.) and an interface for creating a smooth shape from a drawing of its visible contour. The shape representations include various manifold representations, mesh approximations to Laplacian representations, and other approaches drawn from differential geometry and topology.

Intelligent Agents

(Amy Greenwald)

The Intelligent Agents group at Brown is designing and implementing systems that automate decision-theoretic and game-theoretic reasoning. One of our key objectives is to develop faithful representations of user preferences, thereby facilitating human-computer interaction. Given a single user's preferences, we are applying statistical techniques to the design of agent-based information retrieval systems that automatically annotate, classify, filter, retrieve, and deliver web content. Given multiple users' preferences, we are implementing intelligent agents that learn to reason strategically in e-commerce settings, such as Internet auctions and other dynamic pricing games.

Machine Learning

(Michael Black, Eugene Charniak,, Amy Greenwald, Chad Jenkins, Mark Johnson, Erik Sudderth, Eli Upfal)

It is difficult to separate machine learning from artificial intelligence at Brown. Most of the AI faculty utilize statistical machine learning techniques in their work whether they focus on machine vision, game theory, robotics, or computational linguistics. Hidden Markov models, stochastic context-free grammars, graphical models (Bayesian networks and Markov random fields), and Markov decision processes are common mathematical tools used in developing new algorithms and applications. Related research topics include the design and analysis of variational inference algorithms, Markov chain Monte Carlo (MCMC) methods, discriminative learning, and nonparametric Bayesian statistics. There are active reading groups and a good deal of collaboration among faculty in computer science and with colleagues in the Departments of Applied

Math, Cognitive and Linguistic Sciences, Engineering, and Neuroscience.

Mobile and Ubiquitous Computing

(Ugur Cetintemel, Rodrigo Fonseca, John Jannotti, Stanley Zdonik)

Recent years have witnessed a dramatic trend towards ubiquitous computing, whereby very large numbers of casually accessible, mobile or embedded computing devices are connected to an increasingly ubiquitous networking infrastructure. In this context, our research explores a host of data and resource management challenges in newly emerging networked systems, such as sensor networks, mobile ad hoc networks, and Internet-scale information systems. Specific projects include profile-based data management; power profiling and optimization for embedded networking devices; broadcast disks; data recharging; mobile computers (with ad hoc networking) as note-taking, annotation, and collaboration tools; decentralized replication for mobile and weakly connected environments; and adaptive data dissemination and routing in wireless sensor networks.

Another research area related to ubiquitous computing is the area of visual sensor networks (VSNs), which are composed of very large numbers of audio/visual sensors--smart cameras--distributed over a geographic area. The smart cameras coordinate their sensing, communication, and computation in order to acquire relevant information about their environment and to collaborate on high-level tasks. Applications of wired and wireless VSNs include surveillance and security in large, public places, human-computer interaction, and smart living environments. Problems we are addressing in this area include: implementation of VSNs using low-cost off-the-shelf components; the design and fabrication of smart cameras with system-on-chip (SoC) embedded processors, reconfigurable hardware, and open operating system software; the design of optimized low-power real-time data-processing algorithms at the hardware level; software support for low-level image processing and networking operations at the embedded operating system level, including new image-based routing protocols; new collaborative real-time audio/visual processing algorithms; and the development of software models and supporting infrastructure to let users specify how sensor data should be processed, while letting our intelligent algorithms optimize where such processing occurs.

Nanocomputing

(John Savage)

Nanocomputing, the use of nanometer-scale technologies for computation, presents important new algorithmic challenges. Self-as-

sembly will replace traditional photolithography as a means to place circuit designs on chips. As a result, nanochips will have highly regular macrostructures but exhibit randomness at the lowest level. Among the new challenges are a) discovery of methods to assemble and control nanoscale devices, b) development of architectures and algorithms to make the best use of the I/O limitations that arise due to the disparity in wire sizes at the nano and microlevels, c) the design of new algorithms for the efficient use of highly structured, faulty nanoscale structures, and d) methods for containing the high fault rates associated with these very small devices.

Natural Language Processing

(Eugene Charniak, Mark Johnson)

Computational linguistics is the study of computational processes that can comprehend, produce and/or acquire natural language. At Brown, we view computational linguistics as lying at the intersection of artificial intelligence and linguistics, and most of our work relies on statistical and machine-learning techniques that are very similar to those used in other areas of AI at Brown. More specifically, we are interested in developing sophisticated statistical models that describe the hidden linguistic (and non-linguistic!) dependencies in sentences and larger units. We use various models of this kind for parsing (i.e., identifying the syntactic structure of a sentence), speech recognition (especially modeling disfluencies in spontaneous speech), machine translation, question-answering and document retrieval.

Operating Systems and Distributed Systems

(Ugur Cetintemel, Tom Doeppner, Rodrigo Fonseca, Maurice Herlihy, John Jannotti, Stanley Zdonik)

Brown has a long history of research in operating systems and distributed systems, ranging from early work in RPC protocols in the early ’70s to early work in multi-threaded programming in the ’80s and a fair number of research activities today that focus on many different areas. Within distributed systems, we are working on decentralized data replication and caching and adaptive distributed resource management. We are interested in technologies that apply data-management techniques to distributed environments, such as distributed caching, data broadcast, and automated data freshening. We are developing scalable data management algorithms for massively parallel, shared-nothing grids. We also address data management in sensor networks and in data dissemination systems. We are pursuing techniques for understanding the behavior of distributed systems through analysis and visualization, as well as developing environments to support the construction and maintenance of distributed systems. At the more fundamental level,

we are working in the foundations and practice of wait-free and lock-free synchronization algorithms, in the context of both shared-memory multiprocessors and distributed systems. Finally, in operating systems, we continue our work in multithreaded programming with particular emphasis on performance of shared-memory parallel systems.

In Internet computing we work on such topics as data prefetching, proxy cache management, and resource discovery. We are also working on the use of profiles in performing automatic data organization and in support of adaptive content-management networks. We are developing languages and tools to support the development, evolution, and understanding of large-scale Internet-based programs, including the semantics and verification of Web services.

Finally, in operating systems we continue our work in multi-threaded programming with particular emphasis on performance of shared-memory parallel systems.

Parallel Computing

(Maurice Herlihy, John Jannotti, Franco Preparata, John Savage, Eli Upfal, Pascal Van Hentenryck)

Research in parallel computation deals with models of execution in which sets of independent operations can be carried out concurrently. Therefore, it is concerned with formalization of models of parallelism, elucidation of problem parallelism, design of parallel algorithms, space-time tradeoffs, computing system layout, and investigation of the limits to parallelizability.

Programming Languages

(Shriram Krishnamurthi, Pascal Van Hentenryck)

Programming language research at Brown covers the design, implementation, and analysis of programming languages. We focus on three main areas. First, we study and construct different notions of modularity, with a strong nod to the principles derived from software engineering concerns. Second, we build highly declarative programming languages centered around constraints, with special emphasis on combinatorial and numerical constraints. Finally, we are interested in the principles and problems underlying popular programming methodologies such as scripting.

Robotics

(Tom Dean, Chad Jenkins)

Autonomous robotics pertains to the development of computational and engineering methods towards realizing physically embodied

systems whose functionality is of utility and usability across society. Functional robot behavior, however, must overcome many sources of uncertainty in the physical world associated with computational perception, decision making, and motor control. Further, robots will also need to coordinate and adapt their behavior to the needs of human users as well as other robots.

The Brown Robotics Group focuses on enabling robots to be collaborators in the pursuit of human activities through research into robot learning and human-robot interaction. Current research addresses robot learning from demonstration, physical manipulation of objects, multi-robot coordination, multi-modal interfaces for human-robot communication, and predictive modeling of motion dynamics. Robotics is a highly interdisciplinary field; Brown Robotics actively engages in various synergistic efforts in engineering, cognitive science, biomechanics, and neuroscience, such as toward neural control of prostheses.

Scientific Visualization and Modeling

(David Laidlaw, Andy van Dam)

Driven by scientific applications, research in this area centers around toolsmithing--creating, building, and evaluating computational and visualization tools for science. Problem areas include visualization of time-varying 3D vector- and tensor-valued data, information visualization, image processing, classification and segmentation, biological modeling, load balancing in a distributed environment, and distributed finite element mesh refinement. To attack these problems, we develop new user-interface environments and metaphors, look for inspiration from art to create new visual representations, and develop new numerical approaches to model complicated scientific data and physical phenomena computationally and visually. We investigate a range of working environments best suited to a collaborator's visualization challenges, from conventional desktop systems to Brown's fully immersive, four-wall Cave. Active collaborations with systems biologists, fluids researchers, planetary geoscientists, neurosurgeons, developmental biologists, medical imaging researchers, orthopaedic surgeons, bioengineers, perceptual psychologists, and evolutionary biologists help to motivate, guide, and evaluate the research. The many application areas help ensure that the computational and visualization tools are broadly useful. Research in this area is coupled with other areas in the department, in particular **Computer Graphics**, **Computational Neuroscience**, **Computational Biology**, and **User Interfaces and Virtual Reality**.

Security

(Tom Doeppner, Shriram Krishnamurthi, Anna Lysyanskaya, Steve Reiss, Roberto Tamassia)

Today's computer systems are vulnerable in myriads of ways. This is, perhaps, a consequence of the fact that they were not designed with the worst-case scenario in mind, and the problem of protecting them against attack came as an afterthought. How do we protect existing systems from attacks? How do we make sure that future systems are not vulnerable?

We develop practical tools for such tasks as combating denial-of-service attacks (via IP-traceback), checking the integrity of outsourced files and databases, authenticating aggregated data, and visualizing file permissions . We also tackle the policies that dictate the security of systems. We devise theories and build tools for the analysis of industrial-scale access-control policies and their interactions with programs. Using state-of-the-art cryptography, we develop systems that ensure fairness and accountability.

Software Engineering

(Shriram Krishnamurthi, Steve Reiss)

Software engineering is the study of effective ways to design, implement, validate and maintain extremely large software systems. Our research stresses the construction of software tools, especially in the context of programming environments. Our work currently focuses on the consistent evolution of software, the bridge between the structure and behavior of software systems, and on combining tools in an integrated environment. We are also part of the multi-university PLT Scheme project, which produces the DrScheme programming environment.

Theory of Computation

(Maurice Herlihy, Sorin Istrail, Philip Klein, Shriram Krishnamurthi, Anna Lysyanskaya, John Savage, Meinolf Sellmann, Roberto Tamassia, Eli Upfal)

Researchers in theoretical computer science at Brown develop serial, parallel, and distributed models of computation, establish fundamental limits on computation, identify problems that are computationally feasible and infeasible, not only classify problems by their use of computational resources, such as space, time, number of I/O operations, and chip area, but also examine tradeoffs between resources, study the expressive power of programming languages, explore the nature of proof, develop the foundations for reliable and

secure communications, and develop verifiable methods for the specification of tasks.

User Interfaces and Virtual Reality

(Michael Black, John Hughes, David Laidlaw, Steve Reiss, Andy van Dam)

Brown is a leader in the research of Post-WIMP (Windows, Icons, Menus, Pointing) user interfaces, including multi-modal interfaces. Brown's research spans a broad range of form factors, including traditional desktops, large-format stereo displays, pen and (multi-) touch systems, and fully immersive virtual reality environments. We develop new interaction metaphors and techniques, evaluate their effectiveness, and compare the different types of environments to develop an understanding of their strengths and weaknesses. In collaboration with the Rhode Island School of Design, we study how the design process can complement the development of user interfaces. Many scientific domains involve analysis and visualization of 3D and higher-dimensional data; collaboration with scientists in those domains drives our research and helps evaluate it. Similarly, we are advancing pen-centric computing by investigating how to make tasks naturally suited to pen or other gestural input as fluid on a computer as interacting with pencil and paper. Beyond productivity increases, we hope to discover new workflows and make computational assistance seamless for tasks like working with 2D notations (e.g., math, chemistry, music notation, and diagrams), note taking, and storyboarding. Facilities include a Tablet PC laboratory, several "fishtank" virtual-reality setups, a tiled (3x3) Powerwall, and one of the few Cave virtual environments in the northeast. This research area is closely coupled with Scientific Visualization and Modeling. In collaboration with scientists in the Brain Sciences Program, we are building a new generation of direct brain-machine interfaces using implanted electrode arrays to control computer displays and robotic devices.

Verification and Reliable Systems

(Shriram Krishnamurthi, Steve Reiss, Pascal Van Hentenryck)

Our main focus is on the modular verification of systems. We place particular emphasis on software systems, addressing the problems peculiar to them and exploiting the advantages they confer. Our verification work is strongly informed by our expertise in programming languages and software engineering. A second focus is the design and implementation of reliable and validated numerical algorithms, including nonlinear and differential equations, using interval reasoning. A third emphasis is on creating support for non-traditional

programming languages such as access-control policies and spreadsheets.

THE UNIVERSITY

Brown University is the seventh oldest institution of higher learning in the U.S. It was founded in 1764 as Rhode Island College in Warren, RI, and moved in 1770 to its present location on College Hill overlooking the capital city of Providence. In 1804, in recognition of a gift, it took its present name of Brown University. In 1971 Pembroke College, the women's college associated with Brown, merged with the University.

Around the middle of the past century, Brown began to speak of itself as a university college, a phrase meant to suggest that the school has retained something of the intimacy of a college even while it has become a major center for research and advanced studies. The present University consists of the undergraduate College, the Graduate School, and the School of Medicine.

See **www.brown.edu**.

THE AREA

Brown University is located in an historic residential area of Providence, a dynamic medium-sized city with a handsomely renovated riverfront, one hour from Boston and three hours from New York. The University and the nearby Rhode Island School of Design regularly present a broad collection of seminars, colloquia and other events. Many cultural resources are available, including concerts, theater, museums, art galleries and WaterFire, an enchanting multimedia fire installation symbolic of the city's renaissance. Recreational activities are numerous and include excellent University athletic facilities, both indoor and outdoor, the East Bay Bike Path, Rhode Island's many superb beaches, and New England ski areas. Miller Residence Hall, on the Pembroke Campus, offers dormitory accommodations for graduate students, and many rooms and apartments are available within walking distance of the University. See **www.providenceri.com**.

GRADUATE PROGRAM APPLICATION

Graduate School information can be obtained at **gradschool.brown.edu**. If you wish to apply for admission, either use the online application at **gradschool.brown.edu/go/admission**.

The deadline for receipt of Ph.D. applications (including GRE scores and letters of recommendation) for the fall term is **January 2**. Applications may be considered after this date, but may also be at a disadvantage in the awarding of financial support. The Sc.M. program has rolling admissions, which means applications are accepted until the available spots are filled. The deadline for receipt of Sc.M. applications for fall admission is **July 15** and for spring admission is **December 11**. Because space is limited and admission is competitive, applicants are encouraged to apply as early as possible.

The department requires both verbal and quantitative Graduate Record Examinations for Ph.D. applications and recommends, but does not require, the Computer Science GRE subject examination. Applicants are advised that the subject test provides an additional objective form of evaluation and that this information is often helpful in demonstrating an applicant's abilities.

Applicants are urged to express clearly their area or areas of academic and research interest; applicants who do not provide such information are difficult to evaluate for admission.

A number of University Fellowships, Teaching Assistantships and Research Assistantships are available for Ph.D. candidates. These positions cover the nine-month academic year and include a combination of tuition remission and stipend. Assistantships require a maximum of 20 hours/week teaching or research. In addition, some Research Assistantships cover the summer months.

FOREIGN STUDENTS

Information for foreign students regarding visas, financial policies, etc. is available from the Graduate School. Foreign students whose native language is not English cannot be admitted until the results of the TOEFL examination have been received.

FURTHER INFORMATION

Please access our website for detailed course descriptions, specific financial awards for which you may qualify, and graduate application information: **www.cs.brown.edu/grad/applications.** The graduate FAQs are updated regularly and are the best source of information.

Front cover: *photo of the Thomas J. Watson, Sr., Center for Information Technology, home to the Department of Computer Science, by Amy Tarbox.*

Back cover: *What's happening inside your Ajax Web application? Arjun Guha and Shriram Krishnamurthi are studying it using program analysis. The background represents the control-flow on the client, while the foreground is what the server sees -- all generated automatically.*