

UNDERGRADUATE HANDBOOK

2009–2010 Edition



**Department of Computer Science
Yale University**

Contents

1	Introduction	1
2	Faculty	1
3	Overview of the Department	2
3.1	Artificial Intelligence	2
3.2	Computer Graphics	3
3.3	Computer Systems and Networking	3
3.4	Programming Languages	4
3.5	Scientific Computing	5
3.6	Theory of Computation	6
4	Computing Facilities	7
5	Degree Programs in Computer Science	7
5.1	B.S. and B.A. in Computer Science	8
5.2	Special Projects	11
5.3	Combined B.S./M.S. in Computer Science	15
5.4	B.S. in Computer Science and Mathematics	16
5.5	B.A. in Computer Science and Psychology	17
5.6	B.S. Electrical Engineering & Computer Science	19
6	Program in Computing and the Arts	22
7	Miscellany	25
7.1	Class Advisors and the DUS	25
7.2	Life After Yale	26
7.3	Undergraduate Research	27
7.4	DSAC	27
7.5	Undergraduate Prize	28
7.6	Additional Sources of Information	29
8	Course Listings	29
8.1	Introductory Courses	29
8.2	Intermediate Courses	32
8.3	Advanced Courses	32
8.4	Graduate Courses	40
8.5	Related Courses in Other Departments	40
9	Yale College Calendar / Deadlines	42

1 Introduction

The Department of Computer Science at Yale University was started in 1969 as a small graduate department and began offering an undergraduate major in 1972. There are now 20 regular faculty members, 8 adjunct or affiliated faculty, and 4 research scientists; more than 40 undergraduate majors (20 seniors last year); and more than 50 graduate students. The department offers more than 30 different undergraduate courses each year, all taught by faculty. Further information is available from the departmental web page <http://www.cs.yale.edu>.

2 Faculty

Department Chairman

Avi Silberschatz

Director of Undergraduate Studies

Stanley Eisenstat

Director of Graduate Studies

Vladimir Rokhlin

Professors

Dana Angluin	David Gelernter	Zhong Shao
James Aspnes	Paul Hudak	Avi Silberschatz
Julie Dorsey	Drew McDermott	Daniel Spielman
Stanley Eisenstat	Vladimir Rokhlin	Steven Zucker
Joan Feigenbaum	Holly Rushmeier	
Michael Fischer	Martin Schultz	

Associate Professors

Brian Scassellati	Yang Richard Yang
-------------------	-------------------

Assistant Professors

Daniel Abadi	Bryan Ford
--------------	------------

Adjunct Faculty

Willard Miranker

Affiliated Faculty

Kei-Hoi Cheung	Yiorgos Makris	Edmund Yeh
Ronald Coifman	Stephen Morse	
Mark Gerstein	Andreas Savvides	

3 Overview of the Department

Computer science is one of the most dynamic and progressive intellectual enterprises of our age. At Yale our focus is on the structure, design, and fundamental properties of computers and computer programs and on methods for using computers to solve significant problems. We use mathematics extensively in the design and analysis of problem-solving techniques and the exploration of fundamental properties of computation, and draw heavily on techniques from engineering and from the natural sciences as well.

There are six main areas of study: artificial intelligence, computer graphics, computer systems and networking, programming languages, scientific computing, and theory of computation. In addition there are collaborations with other disciplines, including economics, engineering, law, psychology, mathematics, medicine, and the arts.

3.1 Artificial Intelligence

Artificial Intelligence is the study of computational models of the mind. At Yale there is a wide variety of topics studied, including vision, robotics, planning, learning, and computational neuroscience.

The term “artificial intelligence” is somewhat misleading because the focus of research in the field is often on more mundane activities, such as simple visual perception, than the word “intelligence” would suggest. The field has learned over the years that the effortlessness of a skill such as vision is deceptive, that in fact the brain does a great deal of hard labor behind the scenes to allow us to see without conscious effort. It will take us years to duplicate the skills that nature evolved over eons.

In general we think it is a mistake for AI research to focus on central mental function and ignore input and output. In the long run machines will not be treated as intelligent unless they can perceive and manipulate the objects around them. Real perception and action impose stubborn constraints on thinking. Sophisticated robot planning is wasted if the robot crashes into the wall while trying to generate a predicate-calculus description of the world in front of it. Hence our focus is on real-time perceptual control of behavior, in both natural and artificial systems.

AI uses many of the same techniques as other areas of computer science application, from numerical optimization to symbolic indexing. The key to solving any problem is always the algorithm and its analysis. The goal is always to characterize precisely a set of problems and demonstrate an algorithm that solves them with reasonable efficiency. But, at least at its current state of development, AI is of necessity more exploratory than other areas. We are often forced to define a problem at the same time that we

try to solve it. It often happens that we don't know how to analyze the performance of an algorithm with existing tools, but we believe that its average-case performance is much better than its worst-case performance, and this belief must be backed up with experiments.

Faculty members working in this area are Drew McDermott, Brian Scassellati, and Steven Zucker. Michael Hines is a research scientist.

3.2 Computer Graphics

Research in computer graphics at Yale includes the areas of modeling and interacting with architectural-scale scenes, sketching and alternative design techniques, material and texture models, applications of perception to computer graphics, applications of computer graphics in cultural heritage, and recovering shape and reflectance from images.

With the proliferation of 3D graphics capabilities and the introduction of virtual reality systems, an increasing number of applications are being developed that require the interactive visualization of complex 3D scenes. However impressive the evolution of graphics hardware over the past thirty years, the goal of realistically modeling and interactively manipulating scenes of industrial complexity remains an elusive one. At Yale the research along this line is to develop (1) improved methods for the capture and editing of architectural-scale models with a mix of 3D scanning, digital photography, and novel user interfaces and (2) new algorithms to accelerate the visualization of very complex 3D scenes using novel simplification techniques and image-based impostors.

While researchers have made great strides in light transport algorithms, or rendering, simulations depend just as much on the underlying material models. Unfortunately, the models widely used in computer graphics assume that the materials are both pristine and immutable, even though real materials are neither. The research on material and texture models at Yale is to devise both new material representations and operators for generating and capturing a broad range of complex surface appearances, and to develop methods both to simulate materials and the processes that affect them, and to physically measure the input required for these models.

Faculty members working in this area are Julie Dorsey and Holly Rushmeier.

3.3 Computer Systems and Networking

Computer systems research at Yale is divided into three sub-areas (with a large degree of overlap and collaboration across these sub-areas): database systems, operating systems, and networking systems.

Database systems provide an environment for storage and retrieval of both structured and semi-structured data. Such systems were originally designed for use in business-type applications. Today, however, they are being utilized in many other application domains, including scientific computing, networking, and bioinformatics. Research topics at Yale include transaction management, data warehousing, Web-scale databases, real-time systems, multimedia systems, approximate queries, and data mining.

The role of operating systems has evolved over time, from sharing one device's resources among many users in the mainframe era, to providing convenient user interface, storage, and networking abstractions in the personal computer era. As we transition to the ubiquitous computing era, operating systems must now manage a user's information and computation across many computers and devices. Yale is developing new operating system architectures, application environments, and security frameworks to meet today's challenges across the computing spectrum, from mobile personal devices to large-scale Internet services built on grids of many-core processors.

Computer networks allow computers to communicate with one another, and of course form the backbone of the Internet. But although they have become a critical infrastructure of our information-based society, they still have not achieved the reliability of traditional telephone networks. Research at Yale concentrates on designing highly robust and efficient Internet backbone networks, by combining computer science with optimization, economics, and game theory. Peer-to-peer (P2P) is emerging as a new paradigm for network application development, as witnessed by the wide usage of P2P file-sharing and video-streaming applications. However, these applications not only generate a large volume of traffic, but also may unnecessarily spread traffic across the whole Internet, leading to inefficiency. Research at Yale focuses on designing effective architecture and algorithms to improve both application performance and Internet operation efficiency.

Faculty members in the Computer Systems and Networking area are Daniel Abadi, Bryan Ford, Yiorgos Makris (EE), Andreas Savvides (EE), Avi Silberschatz, and Yang Richard Yang.

3.4 Programming Languages

Programming languages are the main vehicle for man-machine communication. They provide a way to express an algorithm as a program and impact the way we think of a computer system. Several languages developed at or associated with the department (in particular Haskell, ML, and Linda) have achieved worldwide currency, reflecting the department's leadership in the areas of functional programming and parallel computing. Parallel languages

such as Linda are tools for building programs that do many things simultaneously; functional languages such as Haskell and ML provide a mathematical approach to programming based on a view of a program as a set of simple equations. Applications of programming language research include graphics and animation, networking, computer music, robotics, graphical user interfaces, and systems programming.

Two particular areas of study are *formal methods* and *software ensembles*. Formal methods emphasize the use of formal mathematics to ensure the correctness, reliability, and maintainability of complex software systems. At Yale the study of formal methods focuses on functional programming and related ideas such as computational logic, denotational semantics, type theory, category theory, and program transformation.

Software ensembles are programs that are built out of many separate, coordinated activities, with an emphasis on recognizing and understanding the properties that all such systems share. The search for a precise definition of “coordination language,” and the development of internet applications are new focuses of the software ensemble project.

Faculty working in this area are David Gelernter, Paul Hudak, and Zhong Shao. Nick Carriero is a research scientist.

3.5 Scientific Computing

During the past forty years computers have dramatically changed engineering, medicine, and science. It is now possible to test thousands of designs and run thousands of trials without first building a prototype for each product or conducting an elaborate experiment for each trial. The impact of this new ability, this power to simulate the real thing, is easy to imagine. Reliability, flexibility, efficiency, and often attractive cost have placed scientific computation as the keystone between theory and applications.

Research in scientific computing uses concepts and methodologies from numerical linear and nonlinear algebra and boundary value problems for differential equations. In addressing these areas scientific computing at Yale emphasizes algorithm development, theoretical analysis, systems modeling, and programming considerations. Algorithm development is concerned with finding new, fast, and/or parallel methods. Theoretical analysis evaluates such questions as rates of convergence, stability, optimality, and operation counts. Systems modeling examines the performance implications of the interactions between computationally intensive algorithms, operating systems, and multiprocessor machines. Programming considerations include coding efficiency, numerical accuracy, generality of application, data structures, and machine independence.

Faculty working in this area are Stanley Eisenstat, Vladimir Rokhlin,

and Martin Schultz. Rob Bjornson, Craig Douglas, and Diana Resasco are research scientists.

3.6 Theory of Computation

Theory of computation involves the use of powerful mathematical tools to obtain deep insights into fundamental problems of computation. Not being constrained by the current state of technology, research in this area is free to explore both “what is imaginable” as well as “what is.”

At Yale theory research is concentrated in the areas of discrete mathematics, complexity theory, algorithms, learning, security, and distributed computing. Complexity theory looks at the relation between algorithm and computing device and attempts to determine the inherent difficulty of a computational task. Algorithms involves the invention and analysis of algorithms for sequential and parallel models of computation. Security is the field of computer science that looks at the reliability, privacy, and availability properties of information systems; it is intimately connected to several areas of theory, including algorithms, complexity theory, cryptography, and randomness. Distributed computing extends the domain of computation to encompass systems of concurrent communicating asynchronous processors such as peer-to-peer networks and cluster computers. Uncertainty and failures inherent in such systems provide a special focus for research in this area.

Two concepts that are fundamental to all areas of computer science are computing devices and algorithms. A computing device may be a computer, a network of computers, a circuit, a robot, or a software simulator or interpreter. An algorithm is a precise description of how some task is to be executed by a computing device. The curriculum in theory of computation is designed to provide a solid theoretical basis for the understanding of computing devices and algorithms.

The most important tool for this kind of theoretical understanding is “appropriate abstraction.” The idea is to make a theoretical model of (for example) a computer that ignores enough of the details of any specific computer to be general, but is still specific enough that the theorems proved about it give useful insight into the capabilities of actual computers. The ability to use various levels of abstraction, from the immediately practical to the quite theoretical, is a lasting benefit of an education in computer science.

There is considerable contact with discrete mathematics, graph theory, number theory, mathematical logic, probability and statistics, operations research, economics, computational finance, and other related areas of study.

Faculty working in this area are Dana Angluin, James Aspnes, Joan Feigenbaum, Michael Fischer, and Daniel Spielman.

4 Computing Facilities

The faculty, researchers, and students within the department use a variety of computing resources, ranging from conventional PC's and scientific workstations to high-powered compute-servers and workstation clusters used as parallel computers. These systems are interconnected by an Ethernet local area network, which is in turn connected to the Internet via fiber optic technology to the campus backbone.

The educational computing facility for undergraduate majors (affectionately known as the "Zoo" and the site of regular late-night pizza parties; web site: <http://zoo.cs.yale.edu>) is located on the third floor of the Arthur K. Watson building, which houses the department. It consists of 31 Linux workstations and 7 Windows XP graphics workstations (with dual-core, dual-socket Intel Xeon 2.33 GHz processors, 4 Gigabytes of RAM, and 24 inch flat panel monitors). This facility is used for courses in computer science and unsponsored research by majors and is available via both console and remote login 24 hours a day, 365 days a year. Thus students in computer science have essentially unlimited access.

5 Degree Programs in Computer Science

The computer science curriculum offers students training in the theory and practice of computing. A major in computer science prepares one for a job in the field or for graduate study leading to teaching or research. A computer science undergraduate education followed by graduate study in law, business, or medicine is another strong combination.

The department offers both a Bachelor of Science and a Bachelor of Arts major in Computer Science (see §5.1) and a combined B.S./M.S. program (see §5.3). It also participates in joint majors with the Departments of Mathematics (see §5.4), Psychology (see §5.5), and Electrical Engineering (see §5.6).

Each of these programs provides a solid technical education yet allows students either to take the broad range of courses in other disciplines that is an essential part of a liberal education or to complete the requirements of a second major.¹ Thus the number of courses required is somewhat less than at other schools.

The programs are built around a common core of five computer science courses. The first, Computer Science 201 *Introduction to Computer Science*, is a survey that illustrates the breadth and depth of the field to

¹Roughly 25% of our students complete a second major such as Economics, Music, Political Science, or Theater Studies.

students who have already completed a one-term introductory course in programming. The others cover discrete mathematics; data structures; systems programming and computer architecture; and algorithm analysis and design. Together they include the material that every student of computer science should know.

This core is supplemented by a set of electives (and for the joint majors, a set of core courses in the other discipline). The electives give students great flexibility in tailoring the program to specialize in particular areas of computer science or to broaden their knowledge in a variety of areas.

The capstone of each program is the senior project, which lets students experience the challenges and rewards of original scientific research under the guidance of a faculty member. These projects deal with problems that cross the boundaries between courses and can involve complex and imaginative use of computers.

5.1 B.S. and B.A. in Computer Science

A student can earn either a Bachelor of Science or a Bachelor of Arts degree in Computer Science. The B.S. program is designed for students who plan to continue in computing after graduation, including technical management and consulting. The B.A. provides a solid computer science background as preparation for work in other fields.

The B.S. and B.A. degree programs both require the same five core courses

- ◇ Computer Science 201a or b *Introduction to Computer Science*
 - ◇ Computer Science 202a *Mathematical Tools for Computer Science*²
 - ◇ Computer Science 223b *Data Structures and Programming Techniques*
 - ◇ Computer Science 323a *Introduction to Systems Programming and Computer Organization*
 - ◇ Computer Science 365b *Design and Analysis of Algorithms*
- and a senior project (see §5.2), which must be taken as
- ◇ Computer Science 490a or b *Special Projects*.

In addition the B.S. program requires six intermediate or advanced computer science courses as electives, for a total of twelve courses; the B.A., four, for a total of ten. Neither Computer Science 480a or b *Directed Reading* nor Computer Science 490a or b may be counted as electives.

The prerequisite structure of the core courses is shown in Figure 1. Typical schedules beginning in the freshman and sophomore years are given in Tables 1 and 2. Students are strongly advised to complete Computer

²Students with the appropriate background are encouraged to substitute Mathematics 244a *Discrete Mathematics* for Computer Science 202a.

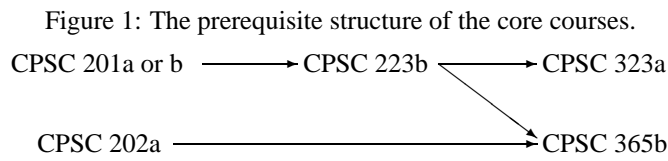


Table 1: Sample B.S. programs for a student starting in the freshman year. Omit two electives from either to get a B.A. program.

<i>Fall</i>	<i>Spring</i>		<i>Fall</i>	<i>Spring</i>
CPSC 201	CPSC 223	<i>Freshman</i>		CPSC 201
CPSC 202	CPSC 365	<i>Sophomore</i>	CPSC 202	CPSC 223
CPSC 323	Elective			
Elective	Elective	<i>Junior</i>	CPSC 323	CPSC 365
Elective	Elective		Elective	Elective
				Elective
CPSC 490	Elective	<i>Senior</i>	Elective	CPSC 490
			Elective	Elective

Table 2: Sample B.S. programs for a student starting in the sophomore year. Omit two electives from either to get a B.A. program.

<i>Fall</i>	<i>Spring</i>		<i>Fall</i>	<i>Spring</i>
CPSC 201	CPSC 223	<i>Sophomore</i>		CPSC 201
CPSC 202				CPSC 223*
CPSC 323	CPSC 365	<i>Junior</i>	CPSC 202	CPSC 365
Elective	Elective		CPSC 323	Elective
				Elective
Elective	CPSC 490	<i>Senior</i>	Elective	CPSC 490
Elective	Elective		Elective	Elective
Elective			Elective	

*Note that it is only possible to take Computer Science 201 and 223 concurrently if one has sufficient programming experience (two courses or the equivalent).

Science 201a or b and 223b by the end of their sophomore year. Otherwise the choice of electives may be somewhat limited, especially in the B.S. program.

All sophomore, junior, and senior majors should have their programs approved by their class advisor (see §7.1) or the director of undergraduate studies.

All courses counting toward the major must be taken for a letter grade.

Electives

The five core courses cover the material that every student of computer science should know; the electives give students an opportunity to specialize in particular areas of computer science.

Students considering graduate study in computer science (either immediately following graduation or after working for several years) are advised to take

- ◇ Computer Science 421a *Compilers and Interpreters*
- ◇ Computer Science 422b *Operating Systems*

and one of

- ◇ Computer Science 465a *Topics in Algorithms*
- ◇ Computer Science 468a *Computational Complexity*
- ◇ Computer Science 469b *Randomized Algorithms*,

as well as courses in their intended area of study.

Students interested in applications of computers to scientific and engineering problems are advised to take

- ◇ Computer Science 440b *Numerical Computation I*

in addition to computational courses in Applied Mathematics and Engineering and Applied Science.

To encourage study in interdisciplinary areas where computer science plays a major role, *advanced* courses³ in other departments that involve concepts from computer science and are particularly relevant to an individual program may, with permission of the class advisor or the director of undergraduate studies, be counted as electives. Generally at most two such courses may be used to satisfy the requirements for the B.S. program (one for the B.A. program).

Even if they cannot be counted as electives, some courses in mathematics (e.g., calculus, linear algebra, probability and statistics, optimization,

³An advanced course is generally one with at least one intermediate course as a prerequisite, and an intermediate course is generally one with at least one (introductory) course as a prerequisite. One exception is Mathematics 120 *Calculus of Functions of Several Variables*, which is considered to be an introductory course.

and discrete mathematics) may be beneficial. For example, some graduate programs require calculus and linear algebra.

SUMMARY OF REQUIREMENTS

Prerequisites: None

Number of courses: *B.S. degree*—twelve term courses taken for a letter grade (including the senior project); *B.A. degree*—ten term courses taken for a letter grade (including the senior project)

Specific courses required: *B.S. and B.A. degrees*—CPSC 201a or b, 202a, 223b, 323a, and 365b

Distribution of courses: *B.S. degree*—six additional intermediate or advanced CPSC courses. *B.A. degree*—four additional intermediate or advanced CPSC courses

Substitution permitted: Advanced courses in other departments, with permission of the class advisor or the director of undergraduate studies

Senior requirement: Independent project (CPSC 490a or b)

5.2 Special Projects

Computer Science 490a or b *Special Projects* fulfills the senior project requirement of the B.S. and B.A. programs. Students select a faculty advisor to guide them in research in a subfield of computer science and submit a written report on the results. Many of these projects break new ground, and papers with Yale undergraduate authors have been published in leading computer science journals.

5.2.1 Frequently Asked Questions Regarding CPSC 490

Who is the Senior Class Advisor (SCA)?

The Senior Class Advisor (SCA), who administers the CPSC 490 projects, is the faculty member who advises and signs the course schedules of all graduating seniors (i.e., the members of the Class of 2010). This year's SCA is Professor Dana Angluin (email: dana.angluin@yale.edu).

What are the deadlines?

Senior majors enrolled in CPSC 490 must submit the CPSC 490 form (which includes a 3-page description of the project and the list of deliverables) by noon on the fourth Thursday of the term. Other students must have this form approved by the DUS *before* submitting their course schedules. Note: Joint majors must submit the CPSC 490 Joint Major form instead.

All students must complete the end-of-term requirements (see below) by noon on the last day of reading period.

Are there any specific requirements?

The precise form of the project is set in consultation with the advisor. However, all students are required to satisfy the following requirements by noon on the day that reading period ends:

- Use the script `/c/cs490/bin/abstract` to submit your name, the title of your project, your advisor's name, and a 250-to-300-word abstract. This information will be added to the on-line database of recent CPSC 490 projects (see <http://zoo.cs.yale.edu/classes/cs490/>).
- Give the SCA a paper copy of your final written report. This document will be added to the (circulating) library of recent CPSC 490 reports.
- Use the script `/c/cs490/bin/submit` to submit a set of web pages describing your project, including a copy of the description submitted with this form. These pages will become part of the on-line database of recent CPSC 490 projects (see <http://zoo.cs.yale.edu/classes/cs490/>).

Note: You must satisfy these requirements even if you plan to continue your project the next term. The only difference is that your electronic abstract, written report, and web pages should constitute an interim progress report (i.e., the level of detail must be the same as in the final versions, but the work described need not be complete).

How do I choose a project?

There are two general approaches

- *student sells project to professor:* e.g., you get an idea, write a 3-page prospectus that describes the scope of the project and includes a list of deliverables, and find a faculty member willing to supervise the work (which may require changes in the prospectus)
- *professor sells project to student:* e.g., a faculty member has a list of possible projects, and you select one (which may involve changes in the nature of the project)

and a host of possibilities in between.

What kind of project is appropriate?

The project should be more than just an extended homework assignment or final course project and should require that you learn more about some area of computer science. To give you some idea of the range of possibilities, the titles, abstracts, and web pages for recent projects are available on-line (see <http://zoo.cs.yale.edu/classes/cs490/>).

Regular courses meet $2\frac{1}{2}$ hours per week and require 2 to 3 additional hours per week for each hour of class. Using this as a guideline for what it takes to earn a course credit at Yale, the project should be something that you can complete in one semester (i.e., 14 weeks) working approximately 7 to 10 hours per week (i.e., in a total of 100–140 hours).

Note: You cannot be paid for your work on the project.

Who may advise a CPSC 490 project?

The official advisor (and thus the person who evaluates the work and assigns the grade) must be a faculty member with an appointment in the Department of Computer Science. However, the *de facto* advisor need not be, as long as the student meets with the official advisor at least once a month.

How can I learn more about projects from past semesters?

The course web pages <http://zoo.cs.yale.edu/classes/cs490/> contain the titles, abstracts, and web pages for recent projects. Copies of the written reports are kept in a circulating library managed by Linda Dobb (508a Watson).

When should I take CPSC 490?

Most students take the course during their final term of enrollment as the capstone of the program. However, students applying to graduate school should take it in the fall (or, with permission of the DUS, in the spring of their junior year) so that they can get a letter of recommendation from their advisor.

Ideally, planning for the project should begin the preceding term (at least to the extent of finding an advisor).

How often may I take CPSC 490?

While you may not count the course as an elective in any of the computer science majors, you may take it more than once for Yale credit.

May I do a two-term project?

Yes. However, you must satisfy the end-of-term requirements at the end of *each* term, and your grade for each semester will be assigned at the end of *that* semester and will reflect what you accomplished. Thus in effect a two-term project is equivalent to two one-term projects, except that the work may be incomplete at the end of the first semester and the electronic abstract, written report, and web pages for the second semester describe the entire project.

Are group projects allowed?

Yes. However, each member of the group must work on a different part of the project, and your electronic abstract, final written report, and web pages must focus on *your own* contributions.

What are the “deliverables?”

Whatever you and your advisor decide you must complete by the end of the project. Possibilities include (but are not limited to) code, theorems, simulation studies, data analysis, written reports, and oral presentations.

5.2.2 Projects, Fall 2008

Alexandra Garr-Schultz, *Texture Extension Techniques and Human Visual Perception: A Validation Study*, Advisor: Holly Rushmeier

Pavel Kamyshev, *Wikipedia Question Answering Probabilistic Model*, Advisor: Drew McDermott

Justin Thaler, *Loopy Belief Propagation for Solving Totally Unimodular Integer Programs*, Advisor: Sekhar Tatikonda (Dan Spielman)

Alexander Thomson, *Language-Based Abstractions in Frameworks for the Development of Certifying Compilers*, Advisor: Zhong Shao

5.2.3 Projects, Spring 2009

Emily Bernier, *The Similarity-Attraction Effect in Human-Robot Interaction*, Advisor: Brian Scassellati

Adam Bouland, *Multi-way Graph Partitioning using Network Flow*, Advisor: Dan Spielman

Shray Chandra, *Global Positioning of N Points by Their Local Distances*, Advisor: Vladimir Rokhlin

Bodin Civilize, *Visualizing Document Content Using Self-Organizing Maps*, Advisor: Holly Rushmeier

Christian Csar, *Examination of Cache Sensitive B^+ Trees for String Indexing*, Advisor: Daniel Abadi

David Golub, *Kinematic Learning for Humanoid Robots*, Advisor: Brian Scassellati

Kyle Gong, *GPSShare: A Location-Sharing Mobile Network*, Advisor: Andreas Savvides

Nathaniel Granor, *Usability of Image/Concept Authentication For Communication Over An Insecure Channel*, Advisor: Michael Fischer

Yagmur Ilgen, *Using MEL to Create a More Detailed House*, Advisor: Holly Rushmeier

Maria Langat, *Spatial and Temporal Data Analysis for Assisted Living Application using Wireless Sensor Networks*, Advisor: Andreas Savvides

Alexander Lemon, *Function Approximation Using Neural Networks with an Application to Character Recognition*, Advisor: Brian Scassellati

John O'Connor, *Yet Another Lisp Interpreting Experiment*, Advisor: Paul Hudak

Graham Radman, *The Importance of Perceptual Resolution in Naïve Gesture Emulation*, Advisor: Brian Scassellati

Kersi Shroff, *The Makings of a Blockbuster—Data Mining Box Office Success*, Advisor: Lisha Chen (Dana Angluin)

Aarlo Stone Fish, *A Limited JavaScript Compiler for Mozilla Firefox 3*, Advisor: Zhong Shao

Justin Thaler, *A New Characterization of Scaled Diagonally Dominant Matrices, with Applications to Belief Propagation for Quadratic Optimization*, Advisor: Sekhar Tatikonda (Dan Spielman)

Dilaver Velioglu, *Cloud Computing: A Survey*, Advisor: Avi Silberschatz

5.3 Combined B.S./M.S. in Computer Science

Exceptionally able and well-prepared students may complete a course of study leading to the simultaneous award of the Bachelor of Science and Master of Science degrees after eight terms of enrollment. The requirements are as follows:

1. Candidates must satisfy the Yale College requirements for the B.S. degree in Computer Science.
2. In fulfilling these requirements, students must complete eight graduate courses from the approved list, up to two of which may, with the permission of the director of undergraduate studies and the director of graduate studies, also be applied toward completion of the B.S. degree. Since the student will be taking Computer Science 490a or b *Special Projects*, at most one of these courses may be Computer Science 690a, 691b, or 692a or b *Independent Project*.

Graduate work must not be entirely concentrated in the final two terms of study, and students must take at least six term courses outside computer science during their last four terms at Yale and at least two undergraduate courses during their last two terms.

Students must apply for admission to this program through the director of undergraduate studies no later than the first day of classes of their third-to-last term in Yale College. Applicants must have achieved at least two-thirds A, A-, or Honors grades in all of their course credits as well as in all course credits directly relating to computer science.

Interested students are advised to consult with their class advisor, the director of undergraduate studies, and the director of graduate studies by the start of their junior year.

5.4 B.S. in Computer Science and Mathematics

The joint major in Computer Science and Mathematics is intended for students who are interested in computational mathematics, the use of computers in mathematics, mathematical aspects of algorithm design and analysis, and theoretical foundations of computing.

The major requires fourteen term courses as well as a senior project: six required courses (which include the core of the computer science major)

- ◇ Computer Science 201a or b *Introduction to Computer Science*
- ◇ Computer Science 223b *Data Structures and Programming Techniques*
- ◇ Computer Science 323a *Introduction to Systems Programming and Computer Organization*
- ◇ Computer Science 365b *Design and Analysis of Algorithms*
- ◇ Mathematics 120a or b *Calculus of Functions of Several Variables*
- ◇ Mathematics 244a *Discrete Mathematics*;

a course in linear algebra, one of

- ◇ Mathematics 222a or b *Linear Algebra with Applications*
- ◇ Mathematics 225a or b *Linear Algebra and Matrix Theory*;

an advanced course in mathematical computer science, one of

- ◇ Computer Science 440b *Numerical Computation I*
- ◇ Computer Science 462a *Graphs and Networks*
- ◇ Computer Science 465b *Topics in Algorithms*
- ◇ Computer Science 468a *Computational Complexity*
- ◇ Computer Science 469b *Randomized Algorithms*

one additional advanced course in Computer Science; and five additional advanced courses in Mathematics (“advanced” courses are those that have as a prerequisite Mathematics 120a or b, 222a or b, 225a or b or some higher-level course).

Students may substitute Mathematics 230 *Vector Calculus and Linear Algebra* for Mathematics 120a or b and Mathematics 222a or b or 225a or b. Neither Computer Science 480a or b *Directed Reading* nor Computer Science 490a or b *Special Projects* nor Mathematics 470a or b *Individual Studies* may be used as an elective.

The senior requirement is a project or a paper on a topic acceptable to *both* departments. Students must submit a written report (including an electronic abstract and web page(s)) to the Computer Science department, and present an oral report on the mathematical aspects of the project to the Mathematics faculty. If taken for course credit as Computer Science 490a or b or Mathematics 470a or b, the senior project course is in addition to the fourteen required courses.

The entire program of a student majoring in Computer Science and

Mathematics must be approved by the directors of undergraduate studies⁴ in *both* departments.

All courses counting toward the major must be taken for a letter grade.

SUMMARY OF REQUIREMENTS

Prerequisites: None

Number of courses: Fourteen term courses taken for a letter grade (not including the senior project)

Specific courses required: CPSC 201a or b, 223b, 323a, 365b, one of 440b, 462a, 465b, 468a, or 469b; MATH 120a or b, either 222a or b or 225a or b, 244a

Distribution of courses: One additional advanced course in computer science; five additional advanced courses in mathematics

Substitution permitted: MATH 230 for MATH 120a or b and 222a or b or 225a or b

Senior requirement: Senior project or senior essay on topic acceptable to Computer Science and Mathematics departments; written report on project to Computer Science department; oral report on mathematical aspects of project to Mathematics faculty

5.5 B.A. in Computer Science and Psychology

The joint major in Computer Science and Psychology is intended for students interested in integrating work in these two fields.⁵ Each area provides tools and theories that can be applied to problems in the other. Examples of this interaction include cognitive science, artificial intelligence, neural models of computation, and biological perception.

The only formal prerequisite for the major is

- ◇ Psychology 110a or b *Introduction to Psychology*.

Beyond the prerequisite the major requires fourteen term courses as well as a senior project.

Eight of the fourteen courses must be in computer science, including the core of the computer science major

- ◇ Computer Science 201a or b *Introduction to Computer Science*
- ◇ Computer Science 202a *Mathematical Tools for Computer Science*
- ◇ Computer Science 223b *Data Structures and Programming Techniques*

⁴In Computer Science the class advisor acts as the DUS (see §7.1).

⁵Students whose interests are less squarely focused on computer science and psychology and extend to philosophy, linguistics, or neuroscience may wish to consider the major in Cognitive Science.

- ◇ Computer Science 323a *Introduction to Systems Programming and Computer Organization*
 - ◇ Computer Science 365b *Design and Analysis of Algorithms*;
- and three advanced courses in artificial intelligence or neural computing. Students may substitute Mathematics 244a *Discrete Mathematics* for Computer Science 202a. Neither Computer Science 480a or b *Directed Reading* nor Computer Science 490a or b *Special Projects* may be used as electives.

The remaining six courses must be in psychology, including

- ◇ Psychology 200b *Statistics*

at least one course on data collection (Psychology 210–299); at least two courses from the social science point of view

List A:

- ◇ Psychology 123a, 125a, 126a, 127a, 128b, 140a, 150b, 180b, 194a, 330b, 341b, 342a, 355;

and at least one course in cognitive psychology or cognitive science listed under Psychology, e.g.,

List C:

- ◇ Psychology 120a *Brain and Thought: An Introduction to the Human Brain*
- ◇ Psychology 130a *Introduction to Cognitive Science*
- ◇ Psychology 140a *Developmental Psychology*
- ◇ Psychology 171b *Sex, Evolution, and Human Nature*
- ◇ Psychology 407b *Cognitive Science of Causality*
- ◇ Psychology 454b *Sensory Information Processing*
- ◇ Psychology 488b *Learning Theory*

Neither Psychology 490a or 491b *Directed Reading* nor Psychology 492a or 493b *Directed Research* may be used as electives.

A second course in cognitive psychology or cognitive science may substitute for one of the courses in artificial intelligence or neural computing. An additional course in psychology may substitute for Psychology 200b if the student has sufficient background in statistics to pass an examination arranged with the instructor.

The senior requirement is a project that is acceptable to *both* departments. Students must submit a written report (including an electronic abstract and web page(s)) to the Computer Science department. If taken for course credit as Computer Science 490a or b or Psychology 492a or 493b, the senior project course is in addition to the fourteen required courses.

The entire program of a student majoring in Computer Science and Psychology must be approved by the directors of undergraduate studies⁶ in *both*

⁶In Computer Science the class advisor acts as the DUS (see §7.1).

departments.

No courses in Computer Science and at most one course in Psychology may be taken on a Credit/D/Fail basis and count for the major.

SUMMARY OF REQUIREMENTS

Prerequisite: PSYC 110a or b

Number of courses: Fourteen term courses beyond prerequisite taken for a letter grade (not including the senior project; one PSYC course may be taken Cr/D/F)

Specific courses required: CPSC 201a or b, 202a, 223b, 323a, 365b, PSYC 200b

Distribution of courses: Eight courses in computer science, with three advanced courses in AI or neural computing; six courses in psychology, with at least one from PSYC 210–299, at least two from List A, and at least one from List C

Substitution permitted: For CPSC 202a, MATH 244a; for one course in AI or neural computing, one course in cognitive psychology or cognitive science; for PSYC 200b, one additional course in psychology and an examination arranged with the instructor

Senior requirement: Senior project acceptable to Computer Science and Psychology departments

5.6 B.S. Electrical Engineering & Computer Science

The joint major in Electrical Engineering and Computer Science is intended for students who want to integrate work in these two fields. It covers discrete and continuous mathematics; algorithm analysis and design; digital and analog circuits; signals and systems; systems programming; and computer engineering. It provides coherence in its core program, but allows flexibility to pursue technical electives.

The prerequisites for the major are

- ◇ Computer Science 112a or b *Introduction to Programming*
- ◇ Mathematics 112a or b *Calculus of Functions of One Variable I*
- ◇ Mathematics 115a or b *Calculus of Functions of One Variable II*
- ◇ Mathematics 120a or b *Calculus of Functions of Several Variables*
- ◇ Physics 180a and 181b *Advanced General Physics*.

Students who must take Computer Science 112a or b should do so during the fall of their freshman year to avoid the time conflict between Computer Science 112b and Physics 181b.

Students may substitute Engineering & Applied Science 151a *Multivariable Calculus for Engineers* or the first term of Mathematics 230 *Vector*

Calculus and Linear Algebra for Mathematics 120a or b; and Physics 200a and 201b *Fundamentals of Physics I and II* for Physics 180a and 181b. Students who must take Mathematics 112a or b may substitute Physics 150a and 151b *General Physics* for Physics 180a and 181b.

Fifteen term courses are required beyond the prerequisites: ten required courses (which include the core of the computer science major)

- ◇ Computer Science 201a or b *Introduction to Computer Science*
- ◇ Computer Science 202a *Mathematical Tools for Computer Science*
- ◇ Computer Science 223b *Data Structures and Programming Techniques*
- ◇ Computer Science 323a *Introduction to Systems Programming and Computer Organization*
- ◇ Computer Science 365b *Design and Analysis of Algorithms*
- ◇ Electrical Engineering 200a *Introduction to Electronics*
- ◇ Electrical Engineering 201b *Introduction to Computer Engineering*
- ◇ Electrical Engineering 202a *Communications, Computation and Control*
- ◇ Electrical Engineering 203b *Circuits and Systems Design*
- ◇ Either Mathematics 222a or b *Linear Algebra* or Statistics 241a *Probability Theory*;

four advanced electives, two in Computer Science, two in Electrical Engineering; and a senior project.

Students are encouraged to substitute Mathematics 244a *Discrete Mathematics* for Computer Science 202a. Students may substitute Mathematics 225a or b *Linear Algebra and Matrix Theory* or the second term of Mathematics 230 for Mathematics 222a or b.

Electives must be either 300- or 400-level courses in the Departments of Computer Science and Electrical Engineering or approved by the directors of undergraduate studies in *both* departments. Cross-listed classes may be counted as being in either department. Computer Science 480a or b *Directed Reading* and Computer Science 490a or b *Special Projects* may not be used as electives.

The senior project must be taken as Computer Science 490a or b or Electrical Engineering 471a or 472b, depending upon the advisor's department, and must be acceptable to *both* departments. Students must submit a written report (including an electronic abstract and web page(s)) to the Department of Computer Science.

A typical program for a student who has had only one term of calculus is shown in Table 4. A typical program for a student who has taken the equivalent of one year of calculus in high school and has the equivalent of one term of programming experience is shown in Table 3.

The entire program of a student majoring in Electrical Engineering and Computer Science must be approved by the directors of undergraduate stud-

Table 3: Sample program for a student with some of the prerequisites.

	<i>Fall</i>	<i>Spring</i>
<i>Freshman</i>	EENG 200 PHYS 180 ENAS 151	EENG 201 PHYS 181
<i>Sophomore</i>	CPSC 201 EENG 202	CPSC 223 EENG 203 MATH 222
<i>Junior</i>	CPSC 202 CPSC 323	CPSC 365 EE elective
<i>Senior</i>	CS elective Senior project	CS elective EE elective

Students with no or little programming experience should take CPSC 112 in the *fall* of the freshman year, either postponing EENG 200 until the sophomore year or taking MATH 120 in the spring instead of ENAS 151 in the fall.

Table 4: Sample program for a student with only one term of calculus.

	<i>Fall</i>	<i>Spring</i>
<i>Freshman</i>	MATH 115 PHYS 180 CPSC 112	MATH 120 PHYS 181 EENG 201
<i>Sophomore</i>	CPSC 201 EENG 200 EENG 202	CPSC 223 EENG 203
<i>Junior</i>	CPSC 202 CPSC 323 STAT 241	CPSC 365 EE elective
<i>Senior</i>	CS elective EE elective	CS elective Senior project

ies⁷ in *both* departments.

All courses counting toward the major must be taken for a letter grade.

SUMMARY OF REQUIREMENTS

Prerequisites: CPSC 112a or b; MATH 112a or b, 115a or b, and 120a or b; PHYS 180a, 181b or 200a, 201b

Number of courses: Fifteen term courses beyond the prerequisites taken for a letter grade (including the senior project)

Specific courses required: CPSC 201a or b, 202a, 223b, 323a, and 365b; EENG 200a, 201b, 202a, and 203b; MATH 222a or b or 225a or b or STAT 241a

Distribution of courses: Four additional 300- or 400-level electives, two in computer science, two in electrical engineering

Substitution permitted: MATH 244a for CPSC 202a; advanced courses in other departments, with permission of both departments

Senior requirement: Independent project (CPSC 490a or b or EENG 471a or 472b) acceptable to both departments

6 Program in Computing and the Arts

The Bachelor of Arts in Computing and the Arts is an interdepartmental major designed for students who wish to integrate work in computing with work in one of the arts disciplines: Art, History of Art, Music, and Theater Studies.

For students with a computing perspective, issues in these disciplines present interesting and substantive problems: How musicians use computers to compose; the limitations of current software tools used by artists; the types of analyses done by art historians; challenges in designing and using virtual sets in the theater; ways that virtual worlds might help to envision new forms of artistic expression; lessons that can be learned from trying to create a robotic conductor or performer.

For students with an artistic perspective, computing methods offer a systematic approach to achieving their vision. A foundation in computer science allows artists to understand existing computing tools more comprehensively and to use them more effectively. Furthermore, it gives them insight into what fundamentally can and cannot be done with computers, so they can anticipate the future development of new tools for computing in their field.

⁷In Computer Science the class advisor acts as the DUS (see §7.1).

Students choose a track in art, history of art, music, or theater studies. The prerequisite for all tracks is

◇ CPSC 112a or b *Introduction to Programming*.

(Students with little programming experience are advised to complete this course during the freshman year.) The additional prerequisites for the *art* track are

◇ ART 111a or b *Visual Thinking*

◇ ART 114a or b *Basic Drawing*.

There are no additional prerequisites for the *history of art* track. The additional prerequisite for the *music* track is

◇ MUSI 210a or b *Elementary Studies in Analysis and Composition I*,

as determined by the Music Theory Placement Test. (Students who do not place into or out of MUSI 210 a or b may have to take a lower-level course first.) The additional prerequisites for the *theater studies* track are

◇ THST 110a and 111b *Survey of Theater and Drama*.

There is no required favorable review of studio work for admission to any track.

Twelve term courses are required beyond the prerequisite(s), not including the two-term senior project. Three Computer Science courses are common to all tracks:

◇ CPSC 201a or b *Introduction to Computer Science*

◇ CPSC 202a *Mathematical Tools for Computer Science*

◇ CPSC 223b *Data Structures and Programming Techniques*.

Students are advised to complete these courses by the end of the sophomore year. Students may substitute MATH 244a *Discrete Mathematics* for CPSC 202a.

The remaining nine courses are track-specific, as specified below. All requirements for a *single* track must be satisfied. The *art* track requires the following courses:

a) three 100-level courses beyond ART 111a or b and 114a or b, such as

◇ ART 132a or b *Introductory Graphic Design*

◇ ART 138a *Digital Photography*

◇ ART 145a *Introduction to Digital Video*;

b) two Art courses at the 200- or 300-level;

c) one Art course at the 400-level;

d) two Computer Science courses chosen from

◇ CPSC 475b *Computational Vision & Biological Perception*

◇ CPSC 478b *Computer Graphics*

◇ CPSC 479a *Advanced Topics in Computer Graphics*;

- e) one additional intermediate or advanced Computer Science course (excluding CPSC 490a or b *Special Projects*).

The *history of art* track requires the following courses:

- a) one introductory History of Art course chosen from
- ◇ HSAR 112a *Introduction to the History of Art: Prehistory to the Renaissance*
 - ◇ HSAR 115b *History of Western Art from the Renaissance to the Present*;
- b) two History of Art courses (representing different areas) at the 200–, 300–, or 400–level;
- c) HSAR 401a or b *Critical Approaches to Art History*;
- d) one 400–level History of Art seminar;
- e) one studio art course (which may have prerequisites);
- f) CPSC 478b *Computer Graphics*;
- g) one Computer Science course chosen from
- ◇ CPSC 437a *Introduction to Databases*
 - ◇ CPSC 475b *Computational Vision and Biological Perception*
 - ◇ CPSC 479a *Advanced Topics in Computer Graphics*;
- h) one additional intermediate or advanced Computer Science course (excluding CPSC 490a or b *Special Projects*).

The *music* track requires the following courses:

- a) two computer music courses from a Music perspective:
- ◇ MUSI 325a *Fundamentals of Music, Multimedia Art and Technology*;
- b) five Music courses chosen from
- ◇ MUSI 312a and 313b *Composition Seminar I*
 - ◇ MUSI 343a *Music Cognition*
 - ◇ MUSI 395a *Compositional Applications in Music, Multimedia Art and Technology*
 - ◇ MUSI 412a and 413b *Composition Seminar II*
 - ◇ MUSI 450b *Special Topics in Music, Multimedia Art and Technology*
 - ◇ MUSI 466b *Music and Multimedia Art*
 - ◇ MUSI 471a or 472b *Individual Study*;
- c) two computer music courses from a Computer Science perspective:
- ◇ CPSC 431a *Computer Music—Algorithmic and Heuristic Composition*;
 - ◇ CPSC 432a *Computer Music—Sound Representation and Synthesis*;
- d) one additional intermediate or advanced Computer Science course (excluding CPSC 490a or b *Special Projects*).

The *theater studies* track requires the following courses:

- a) THST 210a *Introduction to Performance Concepts*;
- b) three courses in dramatic literature or theater history;
- c) two upper-level Theater Studies production seminars in design, directing, or playwriting;
- d) one computer music course chosen from
 - ◇ CPSC 431a *Computer Music—Algorithmic and Heuristic Composition*
 - ◇ CPSC 432a *Computer Music—Sound Representation and Synthesis*;
- e) one graphics course chosen from
 - ◇ CPSC 478b *Computer Graphics*
 - ◇ CPSC 479a *Advanced Topics in Computer Graphics*;
- f) one additional intermediate or advanced Computer Science course (excluding CPSC 490a or b *Special Projects*).

For all tracks the senior requirement is a two-term project supervised by faculty members from both Computer Science and the arts department and approved by the director of undergraduate studies. Students must submit a written report, including an electronic abstract and web page(s). The project is taken as one term of

◇ CPAR 491a or b *Senior Project for Computing and the Arts*

and one term of

◇ ART 495a or b *Senior Project*

◇ HSAR 499a or b *The Senior Essay*

◇ MUSI 490 *Senior Essay for Intensive Majors in the History, Theory, or Composition of Music*

◇ THST 491a or b *Senior Project in Theater Studies*,

depending on the track chosen.

The entire program of each student majoring in Computing and the Arts must be approved by the director of undergraduate studies. Courses taken Credit/D/Fail may not be counted toward the major.

For more information visit the program web site at

<http://graphics.cs.yale.edu/comparts.html>

7 Miscellany

7.1 Class Advisors and the DUS

We have designated a computer science faculty member to serve as the advisor for all members of your Class. Your class advisor will meet with you at the start of each term to discuss your selection of courses and sign your schedule. Your advisor will also be available throughout the year to

answer questions about the major; sign petitions to double-major or change majors; and so forth. Moreover, to provide some continuity, you will usually have the same advisor in both your junior and senior years.

The current class advisors are:

Class	Advisor	Office	E-mail
2010	Dana Angluin	AKW 414	dana.angluin@yale.edu
2011	Stanley Eisenstat	AKW 208	stanley.eisenstat@yale.edu
2012	Stanley Eisenstat	AKW 208	stanley.eisenstat@yale.edu

If your class advisor should be unavailable for an extended period of time, then the Director of Undergraduate Studies, Stanley Eisenstat (AKW 208, email: stanley.eisenstat@yale.edu) can answer your questions and sign your course schedule.

7.2 Life After Yale

Where Do Students Go?

Yale computer science majors are in high demand, both by employers and by graduate and professional schools. For example, last year's seniors went to the following places:

Microsoft	4	Consulting	2
Google	1	Financial services	2
Software companies	1	Graduate/professional	4
Startups	1	Other and Unknown	5

Letters of Recommendation

Prospective employers and graduate/professional schools often ask students to submit letters of recommendation from faculty. Instructors in

- project courses (where you work closely with your advisor),
- small courses (where you are more visible),
- advanced courses (where the demands are greater),
- courses taken by graduate students (with whom you can be compared),
- courses related to the area in which you propose to work or study, and
- courses in which you did well

are generally good choices. However, do not be reluctant to ask any instructor for a recommendation.

The best time to request letters is immediately after completing a course, when memories of you and your performance are freshest. Your college dean's office has standardized forms for letter writers to use and return. These letters are kept on file so that you can have copies sent when needed.

Job Search

Many companies interview at Yale for full-time and summer positions in the

computing field. Check with Undergraduate Career Services for details, and watch the cs-majors mailing list for additional opportunities (see §7.6).

Graduate and Professional School

Many computer science majors go to graduate or professional (i.e., law, business, or medical) school, either immediately after graduation or after working for a few years. In either case it is prudent to have letters of recommendation on file and to have taken any entrance examinations (e.g., the GRE, LSAT, GMAT, or MCAT) before leaving Yale.

Ph.D. programs in computer science generally offer research or teaching assistantships that include tuition and a stipend. You can also apply for fellowships from the National Science Foundation and other organizations. In contrast M.S. programs and professional schools typically do not offer any financial support.

Students interested in graduate school are advised to discuss their plans with their class advisor, the director of undergraduate studies, and the director of graduate studies, preferably no later than the spring of their junior year.

7.3 Undergraduate Research

For a general overview of undergraduate research opportunities at Yale, see the YSER (Yale Science and Engineering Research Program) web site, <http://www.yale.edu/yscr>.

There is no organized program within Computer Science. However, students wanting to do research (other than that done to satisfy the senior requirement) can

- Take one or more terms of Computer Science 290a or b *Directed Research* under the direction of any of our faculty (which earns Yale credit but does not count toward the requirements of the major).
- Work in a research group during the summer or during the academic year. Such paid positions are not common and are arranged directly between a student and a faculty member.

How soon students can begin research depends on their background and area of interest.

7.4 DSAC

The Departmental Student Advisory Committee (DSAC) represents undergraduate computer science majors and provides a liaison between the faculty and undergraduates in matters pertaining to the computer science curriculum

and the majors' use of departmental resources. DSAC also helps with the planning and operation of the Zoo, the undergraduate computing facility for the department (see §4).

As defined by Yale College, DSAC's charter is to review aspects of the department's undergraduate curriculum as it affects both majors and non-majors, and to serve as a channel through which solicited or unsolicited opinions of other students can be expressed. It advises the department on such matters as ideas for new courses and programs; the effectiveness of the curriculum; the scope and sequence of course offerings; the requirements for the major; the role of the senior project; proposals for the improvement of instruction and advising; and the usefulness and interest of specific courses to non-majors.

Feel free to contact DSAC (dsac@cs.yale.edu) if you have suggestions about the curriculum, want help using the Zoo, or have general questions about the Department. DSAC also plans several pizza parties throughout the year for majors and other students interested in computer science.

The members of DSAC for 2009–2010 and their e-mail addresses are

Ahmet Aktay	ahmet.aktay@yale.edu
Christian Csar	christian.csar@yale.edu
Harley Trung	harley.trung
May Liu	may.liu@yale.edu
Michelle Vu	michelle.vu@yale.edu

The DSAC web page is <http://zoo.cs.yale.edu/dsac>.

7.5 Undergraduate Prize

The department awards a prize to the graduating Senior majoring in computer science who, in the judgment of the Computer Science faculty, ranks highest in scholarship.

The ranking is based on grade point average in courses that count toward the major and were taken at Yale. To be eligible, a student must have taken at least 12 such courses. The GPA is computed using the formula on which General Honors are based, but using the list of courses that determines eligibility for Distinction in the Major. Thus 100-level courses; 200-level courses other than CPSC 201, 202, and 223; CPSC 480; all but the first term of CPSC 490, and courses taken only for the M.S. part of the B.S./M.S. program are excluded.

The previous winners were

2009	Justin Thaler
2008	Andrew Smith
2007	Samarth Keshava
2006	Semih Salihoglu
2005	Jian Yuan
2004	Collin Jackson
2003	Manfred Lau
2002	Ameet Talwalkar
2001	Yichen Xie
2000	Michael Bernstein and Samuel Jeong (cowinners)
1999	Kenji Obata

7.6 Additional Sources of Information

The `cs-majors` mailing list contains postings of interest to undergraduates majoring in computer science or taking courses in the subject, including announcements of new courses and faculty, colloquia and other departmental events, recruiting visits and employment/internship opportunities, as well as messages from the director of undergraduate studies. The mailing list can be accessed on the web via

<http://mailman.cs.yale.edu/mailman/listinfo/cs-majors>

It contains all the archived posts as well as instructions on how to subscribe to or un-subscribe from the mailing list.

8 Course Listings

8.1 Introductory Courses

CPSC 079b *Digital Photorealism*. Julie Dorsey
TTh 1:00–2:15

Examination of basic methods used to define shapes, materials, and lighting when creating computer-generated images. Topics include mathematical models for shape, texture models, and lighting techniques. Principles are applied through use of modeling/rendering software. The term project will be the production of a short animated video with rich visual effects. *Proficiency in high-school-level mathematics is assumed. No previous experience with computers necessary.* (Formerly Computer Science 179b.)

CPSC 101b *Great Ideas of Computer Science.*

(Not taught in 2009–2010)

An introduction for nonmajors to some of the most important ideas in computer science: What the computer is; how it works; what it can do and what it cannot do, now and in the future. Topics include algorithms, elementary programming, hardware, language interpretation, software engineering, complexity, models of computation, and artificial intelligence. *No previous programming experience required.*

CPSC 112a or b *Introduction to Programming.*

CPSC 112a. Daniel Abadi MWF 10:30–11:20

CPSC 112b. Drew McDermott MWF 11:35–12:25

An introductory course designed to teach students majoring in any subject how to program computers. The language taught is either C# or JAVA. The focus is on the development of programming skills, problem-solving methods, and selected applications. Topics include data types, control structures, basic algorithms, object-oriented programming, graphical user interfaces, and some advanced programming concepts. *No previous experience with computers necessary.*

CPSC 150a *Computer Science and the Modern Intellectual Agenda.*

David Gelernter

MW 11:35–12:50

An introduction to the basic ideas of computer science (computability, algorithm, virtual machine, symbol processing system) and of several ongoing relationships between computer science and other fields, including philosophy of mind, classical cognitivism, connectionism, and artificial life. *No previous experience with computers necessary.* (Satisfies the WR and HU requirements. Enrollment limited to 25.)

CPSC 151b *The Graphical User Interface: DOS to Windows to What?.*

David Gelernter

MW 11:35–12:50

The role of Graphical User Interfaces (such as the Desktop, with its overlapping windows, icons, menus and pointer device—as embodied in Mac OS, Microsoft Windows etc), on standard platforms such as desktop PCs, laptops, small-screen devices etc. Why did GUIs develop in the way they did? Why have they evolved so little since the Desktop of the 1970s? How will changing hardware and user requirements reshape them in the future? *Prerequisite: Have used a desktop or laptop computer.* (Satisfies the WR and HU requirements. Enrollment limited to 25.)

CPSC 178a *Visualization: Data, Pixels, and Ideas.*

(Not taught in 2009–2010)

An introduction to the use of computer graphics as a medium for communication and discovery. Topics include computer graphics primitives and their association with data, relationships, and concepts to generate an image; real-time interactions with images; and the application of visualization to a variety of application domains, from science and engineering to business and the arts. *No previous experience with computers necessary.*

CPSC 201a or b *Introduction to Computer Science.*

CPSC 201a. Dana Angluin MWF 10:30–11:20

CPSC 201b. Holly Rushmeier MWF 11:35–12:25

An introduction to the concepts, techniques, and applications of computer science for potential majors. Topics include computer systems (the design of computers and their languages); theoretical foundations of computing (computability, complexity, algorithm design); and artificial intelligence (the organization of knowledge and its representation for efficient search). Examples stress the importance of different problem-solving methods. *After Computer Science 112a or b or equivalent.*

CPSC 202a *Mathematical Tools for Computer Science.* Joan Feigenbaum

TTh 1:00–2:15

Introduction to formal methods for reasoning and to mathematical techniques basic to computer science. Topics include propositional logic, discrete mathematics, and linear algebra. Emphasis on applications to computer science: recurrences, sorting, graph traversal, Gaussian elimination.

CPSC 223b *Data Structures and Programming Techniques.*

Stanley Eienstat

TTh 1:00–2:15

Topics include programming in C; data structures (arrays, stacks, queues, lists, trees, heaps, graphs); sorting and searching; storage allocation and management; data abstraction; programming style; testing and debugging; writing efficient programs. *After Computer Science 201a or b or equivalent.*

MATH 244a / AMTH 244a *Discrete Mathematics.* Adam Marcus

TTh 11:35–12:50

Basic concepts and results in discrete mathematics: graphs, trees, connectivity, Ramsey theorem, enumeration, binomial coefficients, Stirling numbers. Properties of finite set systems. *After Mathematics 115a or b or equivalent.*

CPSC 290a or b *Directed Research*. By arrangement with faculty Individual research. Requires a faculty supervisor and the permission of the director of undergraduate studies. *May be taken more than once for credit.*

8.2 Intermediate Courses

CPSC 323a *Introduction to Systems Programming and Computer*

Organization. Stanley Eisenstat

MW 1:00–2:15

Machine architecture and computer organization, systems programming in a high-level language, assembly language, issues in operating systems, software engineering, prototyping in nonprogramming languages. *After Computer Science 223b.*

CPSC 365b *Design and Analysis of Algorithms*. Daniel Spielman

TTh 2:30–3:45

Paradigms for problem solving: divide and conquer, recursion, greedy algorithms, dynamic programming, randomized and probabilistic algorithms. Techniques for analyzing the efficiency of algorithms and designing efficient algorithms and data structures. Algorithms for graph theoretic problems, network flows, and numerical linear algebra. Provides algorithmic background essential to further study of computer science. *After Computer Science 202a and 223b.*

EENG 348a *Digital Systems*. Eugenio Culurciello

TTh 2:30–3:45, lab HTBA

Development of engineering skills through the design and analysis of digital logic components and circuits. Introduction to gate-level circuit design, beginning with single gates and building up to complex systems. Hands-on experience with circuit design using computer-aided design tools and microcontroller programming. *Electrical Engineering 201b is recommended.*

8.3 Advanced Courses

CPSC 421a *Compilers and Interpreters*.

(Not taught in 2009–2010)

Compiler organization and implementation: lexical analysis, formal syntax specification, parsing techniques, execution environment, storage management, code generation and optimization, procedure linkage, and address

binding. The effect of language-design decisions on compiler construction. *After Computer Science 323a.*

CPSC 422b *Operating Systems*. Bryan Ford
MW 1:00–2:15

The design and implementation of operating systems. Topics include synchronization, deadlock, process management, storage management, file systems, security, protection, and networking. *After Computer Science 323a.*

CPSC 424b *Parallel Programming Techniques*. Andrew Sherman
TTh 2:30–3:45

CPSC 425b *Theory of Distributed Systems*. James Aspnes
MWF 11:35–12:25

Models of asynchronous distributed computing systems. Fundamental concepts of concurrency and synchronization, communication, reliability, topological and geometric constraints, time and space complexity, and distributed algorithms. *After Computer Science 323a and 365b.* (Taught in alternate years.)

CPSC 427a *Object-oriented Programming*. Michael Fischer
TTh 1:00–2:15

Object-oriented programming as a means to efficient, reliable, modular, reusable code. Use of classes, derivation, templates, name-hiding, exceptions, polymorphic functions, and other features of C++. *After Computer Science 223b.*

CPSC 428b *Language-Based Security*.
(Not taught in 2009–2010)

Basic design and implementation of language-based approaches for increasing the security and reliability of systems software. Topics include proof-carrying code; certifying compilation; typed assembly languages; runtime checking and monitoring; high-confidence embedded systems and drivers; and language support for verification of safety and liveness properties. *After Computer Science 202a and 323a and Mathematics 222a or b, or equivalent.* (Not taught every year.)

CPSC 430a *Formal Semantics*. Zhong Shao
MW 1:00–2:15

Introduction to formal approaches to programming language design and implementation. Topics include the lambda-calculus, type theory, denotational semantics, type-directed compilation, higher-order modules, and application

of formal methods to systems software and Internet programming. *After Computer Science 202a and 323a.* (Not taught every year.)

CPSC 431a *Computer Music—Algorithmic and Heuristic Composition.*

(Not taught in 2009–2010)

Beginning with high-level representations of music, various deterministic and stochastic algorithms and heuristics are studied for synthesizing music. Similarly, with suitable encodings of harmonic structure and aesthetics, automated methods for analyzing music are studied. Many of the ideas have connections to conventional music theory, but many do not, instead drawing their inspiration from mathematics, computation, and nature (such as fractals, L-systems, and stochastic methods). A key idea underlying the course is the use of a high-level functional language to express the theoretical concepts in a practical manner. Regular programming assignments lead toward a final project that is a software realization of a student-designed concept. *After Computer Science 202a and 223b. Assumes ability to read music.* (Taught in alternate years.)

CPSC 432a *Computer Music—Sound Representation and Synthesis.*

Paul Hudak

MW 2:30–3:45

Beginning with low-level representations of sound, various methods for synthesizing musical sounds are studied, including additive synthesis, subtractive synthesis, frequency modulation, granular synthesis, and physical modeling. The goal is to both simulate as accurately as possible existing musical instruments, and to create new sounds and musical soundscapes. Scales and tuning systems are also studied, as is basic acoustic signal processing (filtering, reverb, sound effects, etc). A key idea underlying the course is the use of a high-level functional language to express the theoretical concepts in a practical manner. Regular programming assignments lead toward a final project that is a software realization of a student-designed concept. *After Computer Science 202a and 223b. Assumes ability to read music.* (Taught in alternate years.)

CPSC 433a *Computer Networks.* Yang Richard Yang

MW 2:30–3:45

An introduction to the design, implementation, analysis, and evaluation of computer networks and their protocols. Topics include layered network architectures, applications, transport, congestion, routing, data link protocols, local area networks, performance analysis, multimedia networking, network security, and network management. Emphasis on protocols used in the Internet. *After Computer Science 323a.* (Taught in alternate years.)

CPSC 434b *Mobile Computing and Wireless Networking.*

(Not taught in 2009–2010)

An introduction to the principles of mobile computing and its enabling technologies. Topics include principles of mobile computing; wireless systems; information management; location-independent/dependent computing models; disconnected and weakly connected operation models; human-computer interactions; mobile applications and services; security; power management; and sensor networks. *After Computer Science 202a and 323a.* (Taught in alternate years.)

CPSC 435b *Internet-Scale Applications.* Yang Richard Yang

TTh 1:00–2:15

An introduction to the design and implementation of Internet-scale applications and services. Topics include: service-oriented software design; cloud computing paradigms; infrastructure scalability and reliability; adaptive, open clients; protocol specification; performance modeling; debugging and diagnosis; and deployment and licensing. *After Computer Science 323b.*

CPSC 436a/EENG 460a *Networked Embedded Systems and Sensor Networks.* Andreas Savvides

TTh 11:35–12:50

Introduction to the fundamental concepts of networked embedded systems and wireless sensor networks, presenting a cross-disciplinary approach to the design and implementation of smart wireless embedded systems. Topics include embedded systems programming concepts; low-power and power-aware design; radio technologies; communication protocols for ubiquitous computing systems; and mathematical foundations of sensor behavior. Laboratory work includes programming assignments on low-power wireless devices. *After Computer Science 223b or with equivalent programming experience in a high-level language.* (Open to seniors in Computer Science or Electrical Engineering only.)

CPSC 437a *Introduction to Database Systems.* Avi Silberschatz

TTh 2:30–3:45

An introduction to database systems. Data modeling. The relational model and the SQL query language. Relational database design, integrity constraints, functional dependencies, and normal forms. Object-oriented databases. Database data structures: files, B-trees, hash indexes. *After Computer Science 202a and 223b.*

CPSC 438b *Database System Implementation and Architectures.*

Daniel Abadi

MW 2:30–3:45

A study of systems programming techniques, with a focus on database systems. Half the course is spent studying the design of a traditional DBMS, supplemented by a hands-on exercise where students build various components (e.g., a catalog-manager, a buffer-manager, and a query execution engine) of a DBMS prototype. The other half is spent on non-traditional architectures (parallel databases, data warehouses, stream databases, Web databases). *After Computer Science 202a and 223b.*

CPSC 440b *Numerical Computation I.* Vladimir Rokhlin

TTh 1:00–2:15

Algorithms for numerical problems in the physical, biological, and social sciences: solution of linear and nonlinear systems of equations, interpolation and approximation of functions, numerical differentiation and integration, optimization. *After Computer Science 112a or b or an equivalent introductory programming course; Mathematics 120a or b; and Mathematics 222a or b or 225a or b or Computer Science 202a.*

CPSC 445b *Introduction to Data Mining.* Vladimir Rokhlin

TTh 1:00–2:15

A study of algorithms and systems that allow computers to find patterns and regularities in databases, to perform prediction and forecasting, and to improve their performance generally through interaction with data. *After Computer Science 202a and 223b and Mathematics 222a or b, or equivalents.*

CPSC 455a/ECON 425a *Economics and Computation.*

(Not taught in 2009–2010)

A mathematically rigorous investigation of the interplay of economic theory and computer science with an emphasis on the relationship of incentive-compatibility and algorithmic efficiency. Particular attention will be paid to the formulation and solution of mechanism-design problems that are relevant to data networking and Internet-based commerce. *Some familiarity with basic algorithmics and basic microeconomic theory will be helpful.* (Not taught every year.)

CPSC 461b *Foundations of Cryptography.*

(Not taught in 2009–2010)

Foundations of modern cryptography and their application to computer and network security. Topics include randomized models of computation, indistinguishability, computationally hard problems, one-way and trapdoor func-

tions, pseudorandom generators, zero-knowledge, secure computation, and probabilistic proofs. *After Computer Science 467a.* (Not taught every year.)

CPSC 462a/ AMTH 462a *Graphs and Networks.*

(Not taught in 2009–2010)

A mathematical examination of graphs and their applications in the sciences. Families of graphs include social networks, small-world graphs, Internet graphs, planar graphs, well-shaped meshes, power-law graphs, and classic random graphs. Phenomena include connectivity, clustering, communication, ranking, and iterative processes. *Prerequisites: Linear algebra and discrete mathematics; a course in probability is recommended.*

CPSC 463b *Introduction to Machine Learning.*

(Not taught in 2009–2010)

Paradigms and algorithms for learning classification rules and more complex behaviors from examples and other kinds of data. Topics may include version spaces, decision trees, artificial neural networks, Bayesian networks, instance-based learning, genetic algorithms, reinforcement learning, inductive logic programming, the MDL principle, the PAC model, VC dimension, sample bounds, boosting, support vector machines, queries, grammatical inference, and transductive and inductive inference. *After Computer Science 202a and 223b, or with permission of the instructor. Computer Science 365b is recommended.* (Taught in alternate years.)

CPSC 465a *Topics in Algorithms.*

(Not taught in 2009–2010)

Introduction to the fundamental tools used in approximation algorithms: linear, convex, and semi-definite programming; dynamic programming; and geometric tools. Recent progress in the design of approximation algorithms for graph problems, combinatorial problems, and other NP-hard optimization problems. Results on the hardness of approximation based on probabilistically checkable proofs. *After Computer Science 365b.* (Taught in alternate years.)

CPSC 467b *Cryptography and Computer Security.* Michael Fischer

MW 2:30–3:45

A survey of such private and public key cryptographic techniques as DES, RSA, and zero-knowledge proofs, and their application to problems of maintaining privacy and security in computer networks. Focus on technology, with consideration of such societal issues as balancing individual privacy concerns against the needs of law enforcement, vulnerability of societal institutions to electronic attack, export regulations and international compet-

itiveness, and development of secure information systems. *Some programming may be required. After Computer Science 202a and 223b.*

CPSC 468a *Computational Complexity.* Joan Feigenbaum
TTh 2:30–3:45

Introduction to the theory of computational complexity. Basic complexity classes, including Polynomial Time, Nondeterministic Polynomial Time, Probabilistic Polynomial Time, Polynomial Space, Logarithmic Space, and Nondeterministic Logarithmic Space. The roles of reductions, completeness, randomness, and interaction in the formal study of computation. *After Computer Science 365b or with permission of the instructor.*

CPSC 469b *Randomized Algorithms.*
(Not taught in 2009–2010)

Beginning with an introduction to tools from probability theory including some inequalities like Chernoff bounds, the course will cover randomized algorithms from several areas: graph algorithms, algorithms in algebra, approximate counting, probabilistically checkable proofs, and matrix algorithms. *After Computer Science 365b; a solid background in probability is desirable.* (Taught in alternate years.)

CPSC 470a *Artificial Intelligence.* Brian Scassellati
MWF 10:30–11:20

An introduction to artificial intelligence research, focusing on reasoning and perception. Topics include knowledge representation, predicate calculus, temporal reasoning, vision, robotics, planning, and learning. *After Computer Science 201a or b and 202a.*

CPSC 473b *Intelligent Robotics.* Brian Scassellati
MWF 10:30–11:20

An introduction to the construction of intelligent, autonomous systems. Sensory-motor coordination and task-based perception. Implementation techniques for behavior selection and arbitration, including behavior-based design, evolutionary design, dynamical systems, and hybrid deliberative-reactive systems. Situated learning and adaptive behavior. *After Computer Science 202a and 223b.*

CPSC 475b / EENG 475b *Computational Vision and Biological Perception.*
Steven Zucker
MW 1:00–2:15

An overview of computational vision with a biological emphasis. Suitable as an introduction to biological perception for computer science and engineer-

ing students, as well as an introduction to computational vision for mathematics, psychology, and physiology students. *After Computer Science 112a or b and Mathematics 120a or b, or with permission of the instructor.*

CPSC 477a *Neural Networks for Computing*. Willard Miranker
TTh 11:35–12:50

Artificial neural networks as a computational paradigm studied with application to problems in associative memory, learning, pattern recognition, perception, robotics, and other areas. Development of models for the dynamics of neurons and methods such as learning for designing neural networks. Concepts, designs, and methods compared and tested in software simulation. Brain and consciousness studies are optional topics. *Programming and knowledge of linear algebra and calculus required.*

CPSC 478b *Computer Graphics*. Julie Dorsey
TTh 2:30–3:45

An introduction to the basic concepts of two- and three-dimensional computer graphics. Topics include affine and projective transformations, clipping and windowing, visual perception, scene modeling and animation, algorithms for visible surface determination, reflection models, illumination algorithms, and color theory. Assumes solid C or C++ programming skills and a basic knowledge of calculus and linear algebra. *After Computer Science 202a and 223b.*

CPSC 479a *Advanced Topics in Computer Graphics*. Holly Rushmeier
MW 1:00–2:15

An in-depth study of advanced algorithms and systems for rendering, modeling, and animation in computer graphics. Topics vary and may include reflectance modeling, global illumination, subdivision surfaces, NURBS, physically-based fluids systems, and character animation. *After Computer Science 202a and 223b.*

CPSC 480a or b *Directed Reading*. By arrangement with faculty

Individual study for qualified students who wish to investigate an area of computer science not covered in regular courses. A student must be sponsored by a faculty member who sets the requirements and meets regularly with the student. Requires a written plan of study approved by the faculty advisor and the director of undergraduate studies. *May be taken more than once for credit.*

CPSC 490a or b *Special Projects*. By arrangement with faculty

Individual research. Requires a faculty supervisor and the permission of

the class advisor or the director of undergraduate studies. The student must submit a written report about the results of the project. *May be taken more than once for credit.*

8.4 Graduate Courses

Graduate courses and seminars, some of which are announced informally at the start of the term, may be open to undergraduates. Advanced undergraduates should read the departmental bulletin board and check with their class advisor and the director of undergraduate studies before making course selections.

8.5 Related Courses in Other Departments

AMTH 437a/EENG 437a *Optimization Techniques.*

BENG 445a/EENG 445a *Biomedical Image Processing and Analysis.*

EENG 201b *Introduction to Computer Engineering.*

EENG 425a *Introduction to VLSI System Design.*

EENG 444a *Digital Communication Systems.*

EENG 463b *Fault-Tolerant Computer Systems.*

MATH 222a or b/AMTH 222a or b *Linear Algebra with Applications.*

MATH 225a or b *Linear Algebra and Matrix Theory.*

MATH 270a *Set Theory.*

OPRS 235a/AMTH 235a *Optimization I.*

MUSI 485b *Methods in Computational Musicology.*

PHIL 267a *Mathematical Logic I.*

PHIL 268b *Mathematical Logic II.*

STAT 241a/MATH 241a *Probability Theory.*

STAT 242b/MATH 242b *Theory of Statistics.*

STAT 251b/MATH 251b *Stochastic Processes.*

STAT 364b/AMTH 364b/EENG 454b *Information Theory.*

8.5 *Related Courses in Other Departments*

41

STAT 365b *Data Mining and Machine Learning.*

9 Yale College Calendar / Deadlines

2009

Fall Term

2 Sep.	Wed.	Fall-term classes begin
14 Sep.	Mon.	Course schedules due for freshmen
15 Sep.	Tue.	Course schedules due for sophomores/juniors
16 Sep.	Wed.	Course schedules due for seniors
24 Sep.	Thu.	CPSC 490 forms due, Noon
23 Oct.	Fri.	Midterm
		Last day to withdraw from a course without having the course appear on the transcript
6 Nov.	Fri.	Last day to convert from CR/D/F to a letter grade
20 Nov.	Fri.	Fall recess begins, 5:20 PM
30 Nov.	Mon.	Classes resume
4 Dec.	Fri.	Classes end; reading period begins
		Last day to withdraw from a fall-term course
11 Dec.	Fri.	CPSC 490 projects due, Noon
12 Dec.	Sat.	Final examinations begin, 9:00 AM
19 Dec.	Sat.	Examinations end, 5:30 PM; winter recess begins

2010

Spring Term

11 Jan.	Mon.	Spring-term classes begin
15 Jan.	Fri.	Friday classes do not meet; Monday classes do meet
18 Jan.	Mon.	Martin Luther King Day; classes do not meet
20 Jan.	Wed.	Course schedules due for freshmen
21 Jan.	Thu.	Course schedules due for sophomores/juniors
22 Jan.	Fri.	Course schedules due for seniors
		Last day for seniors to petition for second major
4 Feb.	Thu.	CPSC 490 forms due, Noon
5 Mar.	Fri.	Midterm
		Spring-term recess begins, 5:20 PM
		Last day to withdraw from a course without having the course appear on the transcript
22 Mar.	Mon.	Classes resume
29 Mar.	Mon.	Last day to convert from CR/D/F to a letter grade
26 Apr.	Mon.	Classes end; reading period begins
		Monday classes do not meet; Friday classes do meet
		Last day to withdraw from a spring-term course
3 May	Mon.	CPSC 490 projects due, Noon
4 May	Tue.	Final examinations begin, 9:00 AM
11 May	Tue.	Examinations end, 5:30 PM
24 May	Mon.	University Commencement

Inquiries concerning the contents of this handbook may be referred to:

DIRECTOR OF UNDERGRADUATE STUDIES
DEPARTMENT OF COMPUTER SCIENCE
YALE UNIVERSITY
P.O. BOX 208285
NEW HAVEN, CT 06520-8285

Email: ugradinfo@cs.yale.edu

Phone: 203-432-1283

Web: <http://ugrad.cs.yale.edu>

Last revised August 2009.