



# Graduate Handbook

**University of Utah  
Department of Computer Science  
50 S Central Campus Dr RM 3190  
Salt Lake City, UT 84112-9205**

**(801) 581-8224 (voice)**

**(801) 581-5843 (fax)**

**info@cs.utah.edu**

**<http://www.cs.utah.edu>**

**1997–1998**

Founded in 1965, the Department of Computer Science offers highly-regarded programs at both the undergraduate and graduate levels. Department faculty and students did pioneering work in interactive graphics, stack machine and dataflow architectures, digital recording, graphical user interfaces, three-dimensional rendering, asynchronous circuits, video games, computer algebra, and computer animation. Faculty and alumni have founded a number of well-known companies, including Adobe Systems, Ashlar, Atari, Cirrus Logic, Evans & Sutherland, Myricom, Netscape, Pixar, Pixel-Planes, Silicon Graphics, and Word Perfect.

Graduate students immerse themselves in the research activities of the department, which currently include:

- asynchronous VLSI systems
- compilers
- computer architecture
- computer vision
- educational computing
- high-speed GaS circuits
- human-computer interaction
- natural language processing
- operating systems
- programming languages
- scientific computing and visualization
- software development tools
- structured VLSI design
- automated knowledge acquisition
- computer-aided geometric design
- computer graphics
- databases
- formal methods for system design
- geometric modeling
- information-based complexity
- numerical analysis
- parallel and distributed computing
- robotics
- security
- software engineering
- virtual environments

These research activities are funded from a variety of federal, state, and industrial sources, including the National Science Foundation (NSF), the Defense Advanced Research Projects Agency, the Department of Energy, the Office of Naval Research, the National Institutes of Health, the Utah State Centers of Excellence Program, Silicon Graphics, Hewlett-Packard,

and IBM. The department is a partner in an NSF Science and Technology Center for "Computer Graphics and Visualization" along with Brown, Caltech, Cornell, and the University of North Carolina; it recently received an NSF Research Infrastructure award to support research activities requiring high-bandwidth, low-latency machine-to-machine communications; and it is a key participant in the University's \$20 million Accelerated Strategic Computing Initiative grant from the Department of Energy.

Graduate students have access to hundreds of Unix and Windows NT workstations and to the more specialized equipment that resides in the various research laboratories. This equipment includes a 60 CPU SGI Origin 2000 with 8 Infinite Reality Engines; SGI Power Challenge, Power Onyx, and Origin 200 computers; robot arms, mobile robots, and image digitization and display systems; a variety of visual and non-visual virtual environment interfaces; a professional-quality video editing and teleconferencing facility; advanced graphics display workstations equipped with special-purpose graphics hardware; and a collection of numerically controlled equipment used to produce physical prototypes of computer-generated designs.

The University of Utah is committed to policies of equal opportunity, affirmative action, and nondiscrimination. The University seeks to provide equal access to its programs, services, and activities for people with disabilities. Reasonable prior notice is needed to arrange accommodations.

(This handbook is available online at <http://www.cs.utah.edu/csinfo/handbook/departmental.html>.)

# Computer Science Faculty

<b>Professors</b>	Elaine Cohen, Ph.D. Thomas C. Henderson, Ph.D. John M. Hollerbach, Ph.D. Robert Kessler, Ph.D. Kent F. Smith, Ph.D. William B. Thompson, Ph.D.	Alan L. Davis, Ph.D. Lee A. Hollaar, Ph.D. Gary E. Lindstrom, Ph.D. Richard F. Riesenfeld, Ph.D. Frank Stenger, Ph.D.
<b>Emeritus Professor</b>	Robert R. Johnson, Ph.D.	
<b>Associate Professors</b>	Erik Brunvand, Ph.D. Christopher R. Johnson, Ph.D.	Ganesh Gopalakrishnan, Ph.D. Kris Sikorski, Ph.D.
<b>Assistant Professors</b>	John Carter, Ph.D. Ellen M. Riloff, Ph.D.	Wilson Hsieh, Ph.D. Peter Shirley, Ph.D.
<b>Clinical Professor</b>	David Hanscom, Ph.D.	
<b>Clinical Associate Professor</b>	Joseph L. Zachary, Ph.D.	
<b>Research Professor</b>	Stephen Jacobsen, Ph.D.	
<b>Research Associate Professors</b>	Tony Carter, Ph.D. Chuck Hansen, Ph.D.	Sam Drake, Sc.D.
<b>Research Assistant Professors</b>	Chris Myers, Ph.D.	Mark Swanson, Ph.D.
<b>Adjunct Professors</b>	Andries van Dam, Ph.D. Peter Kind, Lt. Gen., US Army (Ret.)	Martin Griss, Ph.D.
<b>Adjunct Associate Professors</b>	Robert J. Douglass, Ph.D.	Robert McDermott, Ph.D.
<b>Adjunct Assistant Professor</b>	Don Brown, Ph.D.	
<b>Visiting Associate Professor</b>	Abdel Mokkedem, Ph.D.	



# Computer Science Administration

Robert R. Kessler	Chair	581-4653
John Hollerbach	Associate Chair, Research	585-6978
Joseph L. Zachary	Associate Chair, Academics	581-7079
Erik Brunvand	Director, Graduate Studies	581-4345
David Hanscom	Director, Undergraduate Studies	581-7023
Christopher R. Johnson	Director, Computational Engineering and Science	581-7705
Joseph L. Zachary	Director, Educational Programs	581-7079

# Computer Science Staff

Marti Banks	Secretary (part-time)	581-8224
Steve Backus	Computer Professional	581-5844
Mark Bradakis	Computer Professional	585-3358
Erin Davies	Secretary (part-time)	581-8224
Ken Davis	Associate Accountant	585-5025
Monica Heaton	Administrative Assistant	581-8224
Shawn Darby	Administrative Manager II	581-3950
Todd Green	Computer Administrator	581-3569
Corey Hatch	Supervisor, Digital Systems Laboratory	581-7845
Colleen Hoopes	Graduate Academic Program Specialist	581-8224
Chad Lake	Computer Professional	581-8224
Bill Martens	Senior Electronics Technician	581-8065
Jimmy Miklavcic	Computer Professional	581-8770
Raelynn Potts	Executive Secretary	585-5983
Mary Rawlinson	Undergraduate Academic Program Specialist	581-8224
John Smith	Computer Professional	581-8713
Woody Walton	Facility Staff Research Assistant	581-8065



# Contents

<b>Computer Science Faculty</b>	<b>iii</b>
<b>Computer Science Administration</b>	<b>iv</b>
<b>Computer Science Staff</b>	<b>iv</b>
<b>Contents</b>	<b>viii</b>
<b>Figures</b>	<b>ix</b>
<b>1 M.S. and Ph.D. Admissions</b>	<b>1</b>
1.1 Required Application Materials . . . . .	1
1.2 Application Deadline Dates . . . . .	2
1.3 Addresses . . . . .	2
1.4 If You Are Accepted . . . . .	2
1.5 Frequently Asked Questions . . . . .	3
1.6 Transfers Within the CS Graduate Program . . . . .	4
1.6.1 Transfer into the Ph.D. Program . . . . .	4
1.6.2 Transfer into the M.S. Program . . . . .	4
<b>2 Information for M.S. and Ph.D. Students</b>	<b>5</b>
2.1 Curriculum and Examination Requirements . . . . .	5
2.1.1 Fundamental Topics . . . . .	5
2.2 Financial Assistance For M.S. and Ph.D. Students . . . . .	7
2.3 The Graduate Studies Committee . . . . .	9
2.4 Supervisory Committees . . . . .	9
2.5 Course Load . . . . .	9

2.6	Independent Study . . . . .	9
2.7	Transfer Credit . . . . .	10
2.8	Thesis/Dissertation Requirements . . . . .	10
2.9	Thesis Copyright Policy . . . . .	11
2.10	Leave of Absence Policy . . . . .	11
2.11	The Ph.D. Degree . . . . .	11
2.12	The M. Phil. Degree . . . . .	13
2.13	The M.S. Degree . . . . .	13
<b>3</b>	<b>Departmental Facilities</b>	<b>15</b>
<b>4</b>	<b>Computer Science Courses</b>	<b>17</b>
<b>5</b>	<b>CS Faculty And Their Research Interests</b>	<b>25</b>
<b>6</b>	<b>Current Funded Research</b>	<b>37</b>
	<b>Graduate Application Forms</b>	<b>39</b>

# Figures

2.1	Course requirements for M.S. and Ph.D. students in Computer Science . . . . .	6
2.2	Graduate Student Pay Scale . . . . .	8



# 1

## M.S. and Ph.D. Admissions

The Department of Computer Science conducts an extremely active and well funded research program, which allows us to provide high quality graduate education and research experience to a select group of creative and highly motivated graduate students. Approximately twenty-five new graduate students enter the program annually, roughly evenly divided between the M.S. and Ph.D. programs. Most graduate students are supported financially throughout their graduate career via a combination of teaching assistantships (TAs) and research assistantships (RAs). Our admissions standards are high and competition for the limited number of positions in the program is rigorous. Admission is based on an evaluation of both an applicant's *academic profile* and *research potential*. We especially encourage applications from underrepresented minorities and women.

### 1.1 Required Application Materials

The following materials are required of **ALL** applicants to either the M.S. or Ph.D. programs, including those wishing to transfer into the program from another department at the University of Utah:

1. **Application Form:** Complete and return the Department of Computer Science Application for Admission to Graduate School. There is no application fee for the computer science application form. However, once a student is accepted into our graduate program and decides to attend, the student is then required to submit another application form to the University of Utah. The application fee for the university is \$40 for domestic students and \$60 for international students.
2. **Grade Reports and Transcripts:** In accordance with the instructions on the application form, arrange to have two copies of your transcripts sent to the Department of Computer Science directly from the issuing schools.
3. **GRE Aptitude and Advanced Test in Computer Science:** All persons interested in applying to the Department of Computer Science at the University of Utah are required to take the GRE General and *an appropriate Subject exam* and have their scores forwarded to the department. *In almost all cases, the subject exam should be in Computer Science. Applicants wishing to submit subject exam scores from a different area should obtain prior approval from the department. Applicants who are deficient in formal computer science courses may request permission to substitute a related advanced test.* Students should plan far enough ahead to accommodate this requirement. Specifically, applicants must take the GRE exam sufficiently early to allow their scores to arrive by the application deadline.

4. **Letters of Recommendation:** Arrange to have three letters of recommendation sent directly to the Department of Computer Science *Graduate Coordinator*. We strongly prefer that all recommendations be from current or former professors who have a knowledge of the applicant's abilities through classwork, independent study, and/or research. However, a letter from a professional supervisor is acceptable if the supervisor is from a computer-related field and is qualified to judge the academic and technical capabilities, character, and ability of the applicant to perform research. *It is helpful if letters for non-native English speakers address the applicant's English skills.*
5. **Personal Letter:** Send a one- to two-page letter to the Department of Computer Science *Graduate Coordinator*, describing in depth your background, interests, and in particular, your reasons for wanting to pursue graduate studies in computer science at the University of Utah.
6. **TOEFL Scores:** Applicants whose native language is not English must take the TOEFL (Test of English as a Foreign Language) and have the score reported to the Department of Computer Science.

**Note for foreign applicants:** Letters of recommendation and your own personal letter must be written in English.

## 1.2 Application Deadline Dates

*Applicants are normally evaluated for admission effective Fall Semester.* Such applications must be received at the University of Utah by January 15 of the academic year prior to the desired start of studies. Exceptionally well qualified students may be admitted starting Spring Semester with an application deadline of October 1. Applications for entry effective Summer Term are not accepted.

## 1.3 Addresses

Applicants should have *all* application materials (application form, transcripts, GRE scores, letters of recommendation, TOEFL scores (if applicable), and personal letter) sent directly to the Department of Computer Science at the address below. Questions regarding the status of an application or regarding the computer science graduate program in general should be sent to the Graduate Coordinator.

Graduate Coordinator  
University of Utah  
Department of Computer Science  
50 S. Central Campus Drive, RM 3190  
Salt Lake City, UT 84112-9205 USA  
grad-coordinator@cs.utah.edu  
(801) 581-8224

## 1.4 If You Are Accepted

Once you have received your letter of acceptance from the Department of Computer Science and decide to attend, be sure to notify the Department of Computer Science in writing regarding your intention to matriculate. You will then be required to submit another application form to the University of Utah. The application fee for the university is \$40 for domestic students and \$60 for international students. If you are accepted with financial aid, obtain a TA/RA Request form from the Graduate Coordinator and return it promptly.

International Teaching Assistants (TAs who are not native speakers of English) must take either a Test of Spoken English (TSE) or SPEAK test before beginning their first TA assignment. Depending on exam score, remedial instruction at the University's English Language Institute may be required.

## 1.5 Frequently Asked Questions

### What kind of academic background is required?

Neither of the graduate programs (M.S. or Ph.D) is an entry-level degree. Although it is not necessary that you have a B.S. in Computer Science, both degree programs assume a background with knowledge in:

- Programming languages
- Algorithms and data structures
- Operating systems
- Computer system organization
- Machine architecture
- Logic design
- Discrete mathematics

### Is there a GPA cutoff?

Yes, a minimum GPA of 3.0 in undergraduate work is required. However, most students accepted into the graduate program have GPAs well above that level.

### Is the GRE required?

Yes, all applicants are required to take the GRE General and *an appropriate Subject exam*. Because our graduate programs are not entry-level, we strongly prefer that the Subject exam be in Computer Science in order to help us judge an applicant's computer science background. *In truly exceptional circumstances, we may allow an alternate Subject exam to be substituted. Anyone wishing to submit scores from an area other than computer science should consult with the Department's Director of Graduate Admissions prior to submitting his or her application.*

### Is there a TOEFL cutoff?

Yes, 620 is the minimum acceptable score. In addition, any non-native English speaker receiving support as a teaching assistant must attend and pass the appropriate University programs in spoken English.

### What is the application fee? Can it be waived?

There is no application fee for the Department of Computer Science. However, students who are accepted into the computer science graduate program and decide to attend, must then submit an application form to the University of Utah. The university application fee is \$40 for domestic applicants and \$60 for foreign applications. It cannot be waived under any circumstances.

### Is there anything else involved in the admission process besides sending my application to the Department of Computer Science?

Yes, you must supply the department with three letters of recommendation, academic transcripts, a personal statement, official GRE scores, and TOEFL scores (if applicable). Your application cannot be processed by the Department Graduate Admissions Committee until it is complete.

### What are you really looking for in an applicant?

*We are looking for applicants with strong academic backgrounds who have demonstrated a potential to perform creative and innovative research in computer science.*

### Who should I contact if I have any questions?

The Graduate Coordinator.

### **Can I get more information on the Department or copies of the admissions forms electronically?**

Yes. You can get copies of this handbook and all of the admissions forms via anonymous ftp from `ftp.cs.utah.edu`. The pertinent files are in the `grad_info` directory. Furthermore, you can get copies of these files as well as detailed information about individual researchers' interests, ongoing research projects, and recent technical papers via the department's World Wide Web (WWW) server, located at `http://www.cs.utah.edu`.

## **1.6 Transfers Within the CS Graduate Program**

The following section details departmental policy and procedures for currently matriculated computer science graduate students who wish to change degree programs (M.S. to Ph.D. or vice versa), and students who are about to complete a degree program and wish to continue in another. All such applications will be reviewed by the Graduate Admissions Committee to ensure standards consistent with those applied to outside applicants. The same admissions deadlines apply as for outside applicants (i.e., a student wishing to transfer from the M.S. program for the Fall Semester must submit their application by January 15).

### **1.6.1 Transfer into the Ph.D. Program**

A student wishing to transfer to the Ph.D. program must submit the following material to the Graduate Admissions Committee:

1. A "goal letter" explaining the student's motives and describing the intended research area.
2. Letters from three current department faculty members supporting the application. In the event that the applicant's supervisory committee has been formed, then at least one of these letters must be from the chair of that committee.
3. A University of Utah transcript.

Applications will not be reviewed unless at the time of application the applicant has completed at least two academic semesters of graduate study. Summer term does not count in this tally.

### **1.6.2 Transfer into the M.S. Program**

Students currently enrolled in the Ph.D. program who wish to transfer to the M.S. should make their request in writing to the Director of Graduate Studies. Supporting letters from three faculty members expressing willingness to serve on the student's M.S. Supervisory Committee must also be provided.

# 2

## Information for M.S. and Ph.D. Students

The following summarizes the formal requirements and procedures of the M.S. and Ph.D. programs. Other pertinent information can be obtained from the bulletin of the Graduate School of the University of Utah.

Many aspects apply to both the M.S. and Ph.D. degree programs in Computer Science. These will be summarized first, followed by a description of requirements specific to each degree program.

Please see the Graduate Coordinator (currently, Colleen Hoopes) for additional material regarding administrative guidelines, procedures, and help. It is the responsibility of the student to initiate processing the various forms; however, the department and the graduate school will try to help you in any way possible.

Students are expected to maintain an overall GPA of at least 3.0 (B), and no class with a grade less than B- can be counted for graduate credit. University of Utah classes must have a course number of 500 or higher to be considered for graduate credit.

### 2.1 Curriculum and Examination Requirements

#### 2.1.1 Fundamental Topics

The Computer Science graduate program is not considered an “entry level” program; therefore, all incoming students will be expected to have demonstrated a basic understanding of fundamental concepts. Students with non-traditional backgrounds must show proficiency in or complete the courses listed as “background” in Figure 2.1.

The curriculum and examination requirements for M.S. and Ph.D. students are designed to certify that all students who receive a graduate degree have a working knowledge of those topics in computer science that are deemed fundamental by the faculty. This comprises a basic education in the core areas of computer science and a deeper education in one or more areas in which they will perform research. We define a *core* area to be one that affects the way that most computer systems are designed and implemented, and the resulting performance of such systems. M.S. and Ph.D. students are required to successfully complete courses in each of these areas. Ph.D. students are further required to pass a comprehensive exam comprising questions in each core area.

**Core Areas:**

Foundations	Background: 300, 354 Required: 505, 512 Optional: 521, 610, 611, 612
Systems and Languages	Background: 355, 356 Required: 506, 511 Optional: 507, 509, 513, 606
Architecture	Background: 361, 362, 363 Required: 561 Optional: 508, 542, 562, 568

**Application Areas:**

Optional:	520, 522, 523, 531, 533, 534, 537, 567, 628, 651, 667
-----------	-------------------------------------------------------------

Figure 2.1: Course requirements for M.S. and Ph.D. students in Computer Science

In addition, because the end-product of most computer science work is some form of computer application, we require students to be skilled in selected application areas.

### Specific course and exam requirements

Figure 2.1 gives the *background*, *required*, and *optional* courses in each core area as well as applications areas.

It is assumed that all graduate students will have a thorough understanding of the material covered in the background courses. All M.S. and Ph.D. students must take the required courses listed above (unless a waiver is obtained based on prior knowledge, e.g., completion of a similar course taken at another University). Ph.D. students must take an additional eight courses listed as *optional* above, including at least one optional course in each of the core areas and two optional courses in the applications area (again, waivers are possible). To satisfy the requirement, a student must receive a grade of B or better in each course taken. Furthermore, all students must achieve a 3.5 grade point average in these courses (five for M.S. students, thirteen for Ph.D. students, counting only courses taken at the University of Utah).

Graduate School policy dictates that a graduate student who receives a full tuition waiver during any quarter in which he or she holds an assistantship, fellowship or traineeship is required to register for at least nine credit hours, including thesis research and seminars.

All teaching assistants, as well as students receiving fellowship or traineeships stipends, are required to register for CS 687 Seminar in Computer Science for one credit hour.

### Organization and implementation of comprehensive and qualifying exams

All M.S. students must pass an oral comprehensive exam centering on a defense of the student's written thesis proposal.

All Ph.D. students must pass a two-phase written comprehensive exam comprising a three-part core exam, and an exam in their selected area of research specialization. Ph.D. students are expected to take both the core and specialized area exams no later than their third year of study.

- The *core area exam* are be given once a year in the Autumn, and, except in the case of retakes, the three parts must

be taken together. Generally, this exam is offered during the week immediately preceding the start of the Autumn Quarter.

The core area exam is prepared and graded by a faculty committee established annually for that purpose. Each year, the department prepares a reading list to assist students in preparing for the core area exams. This includes relevant fundamental readings in pertinent areas of computer science. All questions in this exam are based on the subject matter of the listed readings.

Students failing all or part of a core area exam will be offered one opportunity the following year to retake those portions of the exam which they failed.

- The *specialization area* portion of the comprehensive exam covers material determined by each student's supervisory committee. This exam assesses the student's preparation for conducting Ph.D.-level research in the selected area. Members of the student's supervisory committee contribute questions to this exam. The supervisory committee provides a written evaluation of this portion of the exam, including a recommendation as to whether the student should be allowed to proceed to the qualifying examination.

Students may retake a specialized area exam at most once.

Subsequent to passing both portions of the comprehensive exam, Ph.D. students must pass an oral qualifying exam focusing on a defense of the student's written dissertation proposal. The qualifying exam should be taken no more than four academic year quarters after passing the comprehensive examination. It must be passed no less than two quarters before the finished dissertation is defended.

## 2.2 Financial Assistance For M.S. and Ph.D. Students

### Assistantships and Fellowships in Computer Science

There are three types of financial aid available to graduate students in the Department of Computer Science. Teaching and research positions are awarded on a quarterly basis. Teaching assistantships are provided by the department, while research assistantships are awarded by individual faculty serving as investigators on research grants and contracts. A third form of support is research fellowships (including traineeships), the terms of which vary.

The duties and benefits of departmental teaching and research assistantships are defined as follows:

1. *Teaching Assistantship*: A teaching assistant is a matriculated student employed .25 to .50 FTE (full-time equivalent) to assist a faculty member in teaching. The teaching assistant is required to meet with students regularly in a classroom, laboratory, or other instructional setting; to assist in instructional duties through lesson and materials preparation; to counsel students outside of the regularly scheduled instructional periods, and to evaluate and grade students' work to aid in the determination of final course grades. Tuition waivers supplement stipend support.
2. *Research Assistantship*: The Director of Graduate Studies assigns each research assistant to a particular faculty member based on mutual agreement of the faculty and student. The duties assigned to a research assistant are consonant with the student's research interests and also useful to the professor's research efforts. A research assistantship can be viewed as a kind of internship, whereby the student learns by practicing under faculty supervision. A student wishing to be a research assistant should inquire directly with appropriate faculty sponsors. Tuition waivers supplement stipend support.

Continuation of financial aid is dependent upon continued competent performance of teaching or research duties, as well as satisfactory progress in the student's program of study.

Research fellowships, such as those awarded by national foundations, can be requested directly from the granting agency under the supervision of the Director of Graduate Studies. In addition, the department annually nominates outstanding graduate students for University Research Fellowships, as well as fellowships for certain categories of graduate students from private corporations and federal agencies. In the past students in the department have been awarded fellowships from NSF, DOE, DARPA, ACM, AMOCO, Apple, ARO, ONR, and IBM.

Description	Level of Support
Student in M.S. program	\$1,150/mo
Student in Ph.D. program (pre-proposal)	\$1,150/mo
Student in Ph.D. program (post-proposal)	\$1,250/mo

Figure 2.2: Graduate Student Pay Scale

---

## Summer Support

Faculty members who are conducting research projects often hire students as research assistants on a half or full-time basis during the summer. Students interested in a summer appointment as a research assistant should make arrangements directly with the appropriate faculty member. There are a few teaching assistantships available during Summer Quarter.

## Graduate Assistantship Duration and Pay Scales

Departmental support as a TA is limited to seven quarters on the Ph.D. level and four quarters on the M.S. level, unless special permission is granted by the Graduate Studies Committee. Support is contingent upon satisfactory progress toward degree completion. The number of assistantships in each quarter depends upon available funding. The department will notify students of appointments as soon as possible before the start of a quarter. It is usually not possible, however, to notify students of their specific assignments (that is, whether they will be an RA or TA) until the beginning of each quarter, as these decisions are based on class enrollment figures. Figure 2.2 gives the current pay scale for Computer Science graduate RAs and TAs at the customary 0.5 FTE (20 hours/week) level. Undergraduate TAs are paid on a separate hourly scale.

Full-time students who are employed by the University are exempt from FICA (Social Security) taxes, and should file a form to this effect. This form is available from the department administrator or from the Payroll office in 103 Park Building. A form must be filed each quarter to keep the exemption in effect.

## Tuition Waivers

Tuition waivers are available to Teaching and Research Assistants serving at 0.5 FTE or greater. Tuition benefits are applicable to all courses 500 level and above contributing to the student's degree course requirements. Students must take between 9 and 12 credit hours to receive a full tuition waiver.

By Graduate School policy, tuition waivers are limited to six quarters for M.S. students. Students entering the Ph.D. program with a master's degree in Computer Science are limited to nine quarters of tuition waiver support. Ph.D. students entering without a master's are limited to fifteen quarters of tuition waiver support.

## Procedures for Applying for Financial Assistance

Qualified full-time graduate students are eligible for financial aid in the form of teaching assistantships or research assistantships as described above. For incoming students, a financial assistance application form can be obtained in the Department of Computer Science as well as from the Graduate Fellowship Office, 312 Park Building. This completed application should be submitted directly to the Department of Computer Science by February 1 for aid to commence the following Autumn Quarter.

## 2.3 The Graduate Studies Committee

The functions of the Graduate Studies Committee are as follows:

1. Oversee the graduate curriculum and make timely recommendations to the faculty on its findings.
2. Execute the departmental graduate policy on admissions and financial support.
3. Make teaching assistantship assignments.
4. Monitor the progress of all graduate students in the Department. This monitoring relies on annual progress reports from students and their advisors (supervisory committee chairs). The advisor of a student whose progress is questionable will be invited to discuss the student's situation with the Committee. Appropriate actions according to accepted departmental guidelines result from the student reviewing process.
5. Recruit new graduate students.

During 1996–97, the Committee consists of Professors Erik Brunvand (chair), John Carter, and William Thompson.

## 2.4 Supervisory Committees

Each student is assigned a supervisory committee whose members guide the student's research program. An M.S. student's supervisory committee administers his or her comprehensive examination; for Ph.D. students, the committee conducts the student's specialized comprehensive examination, the oral qualifying examination, and the dissertation defense. Supervisory committees consist of three faculty members in the case of M.S. committees, and five for Ph.D. committees. The latter must include a member from outside the Department of Computer Science. Any computer science regular faculty member may serve as a supervisory committee chair. Other faculty may chair supervisory committees if accorded that privilege by the regular faculty. Individuals who are not faculty members may serve on supervisory committees if nominated by the regular faculty on the committee, and endorsed by the Graduate Studies Committee and department chair.

Final approval of all supervisory committees is granted by the Dean of the Graduate School. Both M.S. and Ph.D. students must form this committee by the end of the second quarter of study, although a committee may be revised later by petition to the Graduate Studies Committee.

## 2.5 Course Load

Students must be registered for at least three hours per quarter, exclusive of summer quarter, in order to remain in a graduate degree program. Students who do not maintain continuous registration and who have not been granted a leave of absence by the Graduate School are subject to being discharged from the degree program. Students must be registered for at least three credit hours during the quarter of the student's thesis defense.

Normally, nine credit hours per quarter is considered a full load for graduate students. Students being supported via research or teaching assistantships who wish to receive tuition waivers must register for between nine and twelve hours per quarter, as described above.

## 2.6 Independent Study

Independent study (CS 690 and CS 790) can be included in courses presented for the M.S. degree, but not the Ph.D. degree. Independent study for M.S. students will be allowed only when the project is self-contained and independent of thesis

research. Course CS 687 Seminar in Computer Science may not be applied to the course requirements of either degree program.

## 2.7 Transfer Credit

A student may not count more than 12 credit hours of non-matriculated graduate work toward any graduate degree unless the student's registration for more than 12 credit hours is specifically approved in advance by the department chair and the dean of the Graduate School. Graduate courses taken as an undergraduate at the University of Utah cannot be counted towards a degree program unless a petition for graduate credit was filed with the University's Registrar at the time the course was taken.

Students who have done graduate study at other institutions may apply appropriate course work to their program at the University of Utah. The following guidelines apply:

1. The courses must be *bona fide* graduate level class work (e.g., independent study is excluded), with grade B- or better. The course work must have been taken on a postgraduate basis, i.e., not have been used to fulfill Bachelor degree requirements.
2. Credit may be given to M.S. students for up to 9 quarter hours, and up to 30 quarter hours for Ph.D. students. Semester hours are converted to equivalent quarter hours by a 1.5 multiplicative factor.
3. Approval of each course is granted by the student's supervisory committee. Course appropriateness is determined by consideration of course content and the student's declared research area.
4. Approved courses are certified by inclusion on the student's *Application for Admission to Candidacy for the M.S. Degree or Program of Study for the Ph.D. Degree*. (Henceforth, both these forms will be uniformly referred to as "Program of Study" forms.) These must be submitted by the end of the student's second quarter of study.
5. Approval of a course taken elsewhere does not imply fulfillment of any specific required course. Such approval can be petitioned for using a form available from the Graduate Coordinator. Normally, the Utah professor who most recently taught the required course determines whether approval should be granted. This can be done at any time, e.g., prior to filing the Program of Study form. Petitions may be granted even if the course was applied to complete Bachelor degree requirements (see point 1).
6. For Ph.D. students:
  - (a) Approval of courses taken elsewhere has no effect on comprehensive and qualifying exam requirements. Some students may elect to complete a course at Utah even though they may have taken a comparable course elsewhere, to aid in preparation for these exams.
  - (b) Approval of courses taken elsewhere has no effect on University of Utah residency requirements (see Section 2.11).

## 2.8 Thesis/Dissertation Requirements

Both the M.S. and Ph.D. degrees require a thesis (or dissertation in the case of the Ph.D.; the discussion which follows applies to both theses and dissertations) to be completed and successfully defended before the degree can be awarded. The student must provide one copy of the thesis to the chair of the supervisory committee at least three weeks before the thesis defense, and one copy to each of the other committee members at least two weeks prior to the defense. A complete draft of the thesis must be delivered to the Graduate Coordinator one week prior to the announced time of defense. This copy will be made available for public access. Students are expected to offer each committee member a bound copy of the thesis once it is completed.

The supervisory committee must give preliminary approval of the thesis prior to the defense. The student has one month after the defense to make any revisions prior to submitting the thesis to the Graduate School Thesis Editor. After successfully defending the thesis, the student must obtain approval from the Final Reader (typically the supervisory committee chair), department chair, and Dean of the Graduate School. A draft of the final thesis must then be presented to the Thesis Editor. Successful completion of the thesis defense must be reported to the Graduate School at least four weeks before the last day of examinations in the final quarter. Students should also read the document regarding copyright notices provided by the department and declare their intentions regarding granting the department the right to photocopy the thesis before notifying the Graduate Coordinator of completion of the thesis defense.

There will be at most two months to complete any changes required by the Thesis Editor before final acceptance. After that time, the candidate must redo the oral defense of the thesis. Access to all departmental facilities will be withdrawn after a three month period. The final thesis must be filed one week before the end of the quarter of graduation.

## 2.9 Thesis Copyright Policy

The Department of Computer Science has a fiduciary interest in seeing that the results of its research programs are widely disseminated so that other researchers and the public as a whole can benefit. This is especially true for work supported in part by governmental agencies. Copyrighting of a thesis, without making any provision for its legal reprinting, severely limits the dissemination of research results. It also makes it difficult for the department to handle requests for copies of theses received from other universities or researchers.

The Department of Computer Science encourages distribution of theses and dissertations in any of the following ways:

- Through an appropriate agreement with University Microfilms, Inc., allow them to reproduce the thesis on request (this option is only open to Ph.D. dissertations at the present time).
- Give permission to the department to have the thesis reproduced upon request, with the department allowed to recover its costs of reproduction and distribution.
- Furnish the department with the address of the person, agency, or company that will handle the distribution of the thesis.

## 2.10 Leave of Absence Policy

If a student does not plan to take classes during an academic year quarter (i.e., Autumn, Winter, or Spring), a leave of absence must be requested. Contact the Graduate Coordinator for the proper form.

## 2.11 The Ph.D. Degree

The Ph.D. is a research degree offered through the Graduate School. It is awarded to a candidate who has demonstrated breadth in Computer Science in general, and depth in a research specialty within Computer Science. The latter is exhibited through the writing and defense of a dissertation which reports substantial original contributions in an approved area of research.

A student who has been accepted by the Graduate School is formally admitted to candidacy for the Ph.D. by the University at the recommendation of the student's supervisory committee. Admission to candidacy occurs after the student:

- Forms a supervisory committee,

- Files an approved Program of Study form,
- Passes the comprehensive examinations (core and specialized),
- Passes the oral qualifying examination, and
- Submits an approved dissertation proposal.

An application for candidacy must be submitted to the Graduate School no later than the last day preceding the quarter of graduation. For the degree to be conferred, the approved Program of Study form must be completed and the dissertation completed and publicly defended.

Each of these steps is described below. Most of the steps involve completing and submitting a properly signed form. Forms and assistance are available from the Graduate Coordinator.

**Program of Study.** Course work listed on the approved Program of Study form must comprise at least 74 quarter hours of graduate work and dissertation research, exclusive of independent study. Graduate course work applied toward an M.S. degree may be included. At least 20 hours of dissertation research (course number 797) and 54 hours of graduate course work must be included. As discussed earlier, the student is required to maintain a B average or higher in course work listed on the Program of Study form; grades less than B- are not acceptable. All courses listed on the Program of Study form must be at the 500 level or above. CS 501, 502, 503, 687 and independent study cannot be used. The student must satisfy the course requirements detailed in Section 2.1.1. One year of study must be spent in full-time residency at the University (i.e., the student must enroll for a minimum of 9 hours per quarter for three consecutive quarters, summer optionally excluded). After the residency requirement is fulfilled, registration for 3 quarter hours of CS 797 Thesis Research is considered a full load.

The Program of Study form should be filed with the department in the second quarter of study and with the Graduate School prior to taking the qualifying examination. The Program of Study form must be submitted to the Graduate Records Office no later than the last day of the quarter preceding the quarter of graduation.

**Comprehensive and oral qualifying examinations.** Every Ph.D. student must pass a core comprehensive exam comprising three core areas of computer science, plus a comprehensive exam in a specialized area that is closely aligned, but not restricted to, the student's research area. In addition, every Ph.D. student must pass an oral qualifying examination in his or her area of research specialization. Students who have previously completed an M.S. degree in Computer Science should take the core exams no later than the second offering after enrollment; other students, no later than the third offering.<sup>1</sup> Each student should schedule and take the specialized exam during the academic year in which he or she successfully completes the core exams.

The qualifying examination should be passed by the end of the fifth quarter of study (not counting summers) for students with an M.S. in computer science, and by the end of the seventh quarter of study otherwise. These exams are described in greater detail in Section 2.1.

**Dissertation proposal.** The dissertation proposal is the central subject of the qualifying examination. Hence, as noted above, the student should prepare and receive committee approval for a dissertation proposal by the end of the fifth quarter of study (not counting summers) for students with an M.S. in computer science, and by the end of the seventh quarter of study otherwise. A copy of the dissertation proposal will be retained in the student's departmental file. For guidelines on preparing proposals, consult *Discussion on Ph.D. Thesis Proposals in Computing Science*, by H. C. Lauer. Copies are available from the Graduate Coordinator and from the Thesis Editor.

**Completing program of study.** A Ph.D. student is expected to devote the necessary time to courses and research in order to make satisfactory progress toward the degree. Satisfactory progress includes personal participation in the research and teaching environment of the department on a day-to-day basis.

---

<sup>1</sup>Comprehensive exams given in the month of September are considered to have been offered during the academic year about to commence.

**Dissertation.** The Ph.D. dissertation should be completed and defended by the end of the ninth quarter (not counting summers) after the proposal is defended. The completed dissertation must be published either in its entirety (through a legitimate publisher of the student's choice or through University Microfilms) or as one or more articles accepted for publication in approved scholarly journals. An abstract of each dissertation must be published in University Microfilms' Dissertation Abstracts International. Detailed policies and procedures concerning the dissertation are contained in "A Handbook for Theses and Dissertations" published by the Graduate School.

## 2.12 The M. Phil. Degree

This degree requires the same qualifications for admission and scholarly achievement as the Ph.D. However, it does not require a doctoral dissertation, and requires 81 quarter hours of course work. All regulations covering the Ph.D. degree apply to the M.Phil. degree. This degree, like the Ph.D. degree, is a terminal degree; a student cannot be a candidate for both degrees in the same department.

The Department of Computer Science currently considers applications for admission to the M.Phil. program only from students already matriculated in the Ph.D. program.

## 2.13 The M.S. Degree

The M.S. is a research degree offered through the Graduate School. A student who has been accepted by the Graduate School is formally admitted to candidacy for the M.S. degree at the recommendation of the student's supervisory committee. Admission to candidacy occurs after the student:

- Forms a supervisory committee,
- Files an approved Program of Study form,
- Passes the comprehensive examination, and
- Submits an approved thesis proposal.

An application for candidacy must be submitted to the Graduate School no later than the last day preceding the quarter of graduation. For the degree to be conferred, the approved Program of Study form must be completed and the thesis completed and publicly defended.

Each of these steps is described below. Most of the steps involve completing and submitting a properly signed form. Forms and assistance are available from the Graduate Coordinator.

**Supervisory committee.** An M.S. committee consists of three members. A committee typically consists of departmental faculty, but may include qualified external members. The committee should be formed by the second quarter of enrollment in the M.S. program.

**Program of Study.** Course work listed on the approved Program of Study form must consist of at least 45 quarter hours or graduate course work and thesis research. At least 36 quarter hours must be completed in resident study at the University of Utah. A minimum of 30 quarter hours must be in course work with the balance in thesis hours (course number CS 697). Students must complete a minimum of 9 thesis hours, and must be registered for a minimum of 3 credit hours per quarter during the quarter in which the thesis is defended. Students are required to maintain a B average or higher in course work listed on the Program of Study form; grades less than B- are not acceptable. All courses listed on the Program of Study form must be at the 500 level or above.

The Program of Study form should be filed with the department in the second quarter of study and with the Graduate School prior to taking the comprehensive examination. The Program of Study form must be submitted to the Graduate School by the last day of the quarter preceding the quarter of graduation.

**Comprehensive examination.** The comprehensive examination for M.S. students consists of an oral examination on the thesis proposal and research area in a very broad sense. This examination is conducted by the student's supervisory committee and should be completed by the end of the student's fourth quarter of study (not counting summers) as a graduate student in the department. The examination should serve as the defense of the student's thesis proposal as well as to establish competence in the research area.

**Thesis proposal.** The student should prepare and receive approval for a thesis proposal by the end of the fourth quarter of study (not counting summers). A copy of the thesis must be in the student's file. For guidelines on preparing proposals, consult *Discussion on Ph.D. Thesis Proposals in Computing Science*, by H. C. Lauer. Copies are available from the Graduate Coordinator and from the Thesis Editor.

**Completing program of study.** An M.S. student is expected to devote the necessary time to courses and research in order to make satisfactory progress toward the degree. Satisfactory progress includes personal participation in the research and teaching environment of the department on a day-to-day basis.

**Thesis.** A full time student working on an M.S. program is expected to complete the degree requirements within two calendar years. Beyond this period a student generally does not receive graduate financial support from the department. In special circumstances, the student may continue the M.S. program for a third year but without financial support. A student must petition the Graduate Studies Committee to continue beyond the third year. The Graduate School limits M.S. programs to four years.

# 3

## Departmental Facilities

The Departmental Computing Facility is configured to support both instructional and research computing. The instructional facility, or General Computing Facility (GCF), includes more than 130 Unix workstations from Hewlett-Packard, Silicon Graphics, Digital Equipment, and Sun; 30 Hewlett-Packard Pentium/NT machines; and 30 X terminals. These machines are supported by ten file/application servers that provide a total of 200 GByte of disk storage via both NFS file systems and 40 GB of application replicated across five AFS servers. The instructional machines are interconnected with these multi-homed NFS/AFS application/file servers via ethernet, fast ethernet, and FDDI rings. Students in the Department also have access to the College of Engineering's workstation laboratory, which consists of five file/application servers and over 100 Unix workstations from Sun and Hewlett-Packard.

The Research Computing Facility (RCF) includes more than 330 Unix workstations from DEC, HP, Sun, IBM, and SGI; and Pentium NT machines from various vendors. These machines are supported by 20 file/application servers that provide a total of 500 GByte of NFS/AFS disk storage and are supported by the same AFS applications server as in the instructional environment. These common AFS application servers allow the research and instructional computing environment to be easily traversed by both researcher and student. The research computing facility is interconnected with ethernet, fast ethernet, and FDDI rings.

The individual research laboratories contain a wide array of specialized equipment, including

- an SGI Origin 2000 Reality Monster (60 processors, 8 IR heads);
- an SGI Power Challenge (12 processors);
- an SGI Power Onyx (4 processors, 2 RE2 heads);
- a variety of specialized equipment for image analysis, including a GRF-2 light striper and a DIGI-BOT II laser range scanner;
- two FDDI/Fast Ethernet clusters comprised of IBM RS 6000s, HP 700s, and SGI Indigo2s;
- a collection of graphics workstations from each workstation vendor;
- a multi-source video editing environment;
- robot arms;

- specialized D/A and A/D equipment for realtime signal processing.

The College of Engineering operates a research-scale integrated circuit (IC) fabrication facility that is used extensively by Computer Science. Equipment for testing and debugging both internally and externally fabricated circuits is housed in an integrated circuit testing facility that contains state-of-the-art HP, Tektronix and Micromanipulator automated IC testing equipment.

# 4

## Computer Science Courses

Current class schedules and registration information<sup>1</sup> are available on line.

**100 Engineering Computing (4) Qtr: AWS** Prereq: Math 111; Coreq: Physics 221, CS 110

An introduction to the use of a variety of widely available software tools to solve problems in science and engineering. Symbolic computation using Maple, introduction to procedural programming using C or Fortran. Emphasis on hands-on experimentation. Examples drawn from disciplines of science and engineering.

**101 Programming with FORTRAN (3) Qtr: Su** Prereq: Math 106; Coreq: CS 110

An introduction for non-majors to programming using the high-level language FORTRAN. Emphasis on laboratory practice and structured problem-solving techniques. **(Not offered 1997–98.)**

**102 Programming with C++ (3) Qtr: A** Prereq: Math 105; Coreq: CS 110

An introduction for non-majors to programming using the high-level language C++. Emphasis on laboratory practice and structured problem-solving techniques.

**110 Introduction to Unix (1) Qtr: AWSSu** An introduction to the Unix operating system in a workstation environment. Topics include the X Windows system, Unix shell commands, file system issues, text editing with Emacs, electronic mail, and filters and pipes. Emphasis on hands-on laboratory practice.

**120 Programming with C (4) Qtr: Su** Prereq: Math 105, programming experience; Coreq: CS 110

An introduction for non-majors to programming using the high-level language C. Emphasis on laboratory practice and structured problem-solving techniques. **This class is not recommended for students with no previous experience in high level language programming.**

**201 Introduction to Computer Science I (4) Qtr: AWS** Prereq: programming experience; Coreq: CS 110, Math 111

The first course required for students intending to major in computer science. Students should have programming experience in a language such as Pascal or C. An introduction to the engineering skills required to effectively program computers and to the range of issues confronted by computer scientists. The role of procedural abstraction and data abstraction in decomposing programs into manageable pieces. Extensive programming exercises in C++.

---

<sup>1</sup> <http://www.acs.utah.edu/student/index.htm>

**202 Introduction to Computer Science II (4) Qtr: WSSu** Prereq: CS 201

The second course required for students intending to major in computer science. An introduction to the problem of engineering computational efficiency into programs. Classical algorithms (including sorting and searching) and data structures (including stacks, queues, linked lists, sets, trees, and graphs). Analysis of program space and time requirements. Extensive programming exercises in C++.

**300 Discrete Structures (3) Qtr: A** Prereq: CS 202

An introduction to formal logical arguments, finite sets, relations, functions, graphs, semi-groups, groups, elementary Boolean logic, and their applications to Computer Science. Elementary logic.

**354 Software Fundamentals: Algorithms and Data Structures (4) Qtr: A** Prereq: CS 202 and proficiency in C++ programming

Data-structure implementation concepts; memory management; abstract datatypes; execution-time measurement and analysis; sorting algorithms; relationship between choice of data structures and implementation efficiency of set and graph algorithms; derivation and solution of recurrence relations; algorithm design principles; reducibility among problem classes. Laboratory exercises.

**355 Software Fundamentals: Programming Language Structure (4) Qtr: W** Prereq: CS 354

Syntax and semantics of programming languages. Use of a language with higher-order procedures as a descriptive and interpretive base for studying semantic issues. Laboratory exercises.

**356 Software Fundamentals: Object-Oriented Software Engineering (4) Qtr: S** Prereq: CS 355

Detailed examination of object-oriented (O-O) programming language concepts including abstract data types, visibility control, subclasses, dynamic function binding, static type checking, single and multiple inheritance, templates, and exceptions. Software engineering methodologies, including O-O design, abstract classes, and O-O software reuse and re-engineering. Current topics in O-O systems, such as framework-based libraries and O-O databases. Programming practice with C++.

**361 Hardware Fundamentals: Digital Design (4) Qtr: A** Prereq: CS 202, Physics 222.

Number representations, Boolean algebra, minimization of logic functions, design of combinational circuits and arithmetic circuits, design of synchronous sequential circuits and finite state machines, state minimization and assignment, system timing disciplines and asynchronous sequential circuits. Includes a two-hour laboratory that is scheduled during the first week of class.

**362,363 Hardware Fundamentals: Computer Architecture and Design (4,3) Qtr: W,S** Prereq: CS 361

An in-depth two-quarter study of computer architecture and design. Performance measures, parallel and serial communication, hardwired and microprogrammed control. Computer organizations including stack architectures, RISC machines, register issues, memory systems, and parallel processors. Hardware-software interface organization including operating system issues, interrupts, and compilation issues. CS362 includes a two-hour laboratory that is scheduled during the first week of class.

**367 Computer Design Laboratory (3) Qtr: S** Coreq: CS 363

Laboratory for Computer Engineering students to be taken in conjunction with CS 363. Class is project-oriented with groups of students designing, building and testing a programmable device such as a computer or a calculator.

**375 Defect Prevention Software Engineering (3) Qtr: A** Prereq: CS 202

Study of a software engineering process for producing high quality software with increased productivity. Techniques for preventing the introduction of defects during software development.

**376 Scientific Computation (3) Qtr: A** Prereq: Math 252, CS 202

Survey of scientific computation relevant to Computer Science, Chemical Engineering and Engineering in general. Topics covered include floating point arithmetic, systems of linear equations (direct and iterative techniques), nonlinear equations (univariate and multivariate), interpolation and differentiation (divided differences), integration (mechanical and Gaussian quadratures, optimal quadratures), approximation by spline functions (natural splines and B-splines, optimality of splines).

**431 Introduction to Computer Graphics (4) Qtr: A** Prereq: CS 356

Introductory concepts in 3D including vector graphics, raster graphics and interactive methods. Laboratory stresses graphics environmental issues – operating systems, programming languages and realtime interaction. Use of graphics in Computer-Aided Design (CAD).

**432 Computer-aided Geometric Modeling (4) Qtr: W** Prereq: CS 431

Basic techniques for generating computer models of mechanical parts. Solid primitive Modeling and sculptured surface modeling. Case studies using Modeling operations. Experience with standard geometric Modeling packages.

**433 Computer Graphics Applications Programming (4) Qtr: S** Prereq: CS 432

Course includes developing a computer graphics application package based on standard graphics functions as well as attributes of a graphical user interface. Experience in applying interactive computer graphics techniques to industrial problems.

**451,452,453 Software Engineering Laboratory (3,3,3) Qtr: A,W,S** Prereq: CS 356, CS 363, Writing 301, standing as senior in CS Department

**All three classes must be completed to receive credit.** This series of classes teaches the fundamentals of software engineering – system requirements definition, project planning, design, documentation, implementation, verification, validation, and maintenance. Classes are project-oriented with emphasis being placed on the development of significant software systems by small groups of programmers rather than by individuals.

**501 Algorithm and Data Structure Design (4) Qtr: A** Prereq: Programming experience in C++ and permission of instructor. May be taken only by graduate students from other departments.

The design of algorithms and data structures to efficiently solve common classes of programming problems. Introduction to the complexity analysis of algorithms. Laboratory exercises.

**502 Programming Language Concepts (4) Qtr: W** Prereq: CS 501 and permission of instructor. May be taken only by graduate students from other departments.

Functional, imperative, and object-oriented programming styles. The role of language design issues such as orthogonality, parameter passing mechanisms, and type systems in determining the character of a language. The role of abstraction in designing and understanding programs. Techniques for defining the meanings of programs. Laboratory exercises.

**503 Object-oriented Software Engineering (4) Qtr: S** Prereq: CS 502 and permission of instructor. May be taken only by graduate students from other departments.

Introduction to object-oriented programming and its role in software engineering.

**505 Introduction to Theoretical Computer Science (3) Qtr: W** Prereq: CS 300 or graduate standing.

Survey of discrete mathematical concepts relevant to Computer Science. Sets, relations, functions, propositional calculus, predicate calculus, graphs, partial orders, equivalence relations. Finite state-machines, regular expressions, languages, grammars, Turing machines, partial recursive functions, and other simple computation models.

**506 Operating Systems (3) Qtr: A** Prereq: CS 356, CS 363

Characteristics, objectives and issues concerning computer operating systems. Process manipulation, synchronization, memory management, name management, protection, resource allocation, system Modeling, pragmatic issues.

**507 Compiler Principles and Techniques (3) Qtr: S** Prereq: CS 355, CS 362, CS 505

Top-down and bottom-up parsing, lexical analyzers, symbol tables, internal forms and intermediate languages, code generation, code optimization, semantic specifications, error detection and recovery, comparison of methods. Use of software tools for lexical analysis and parsing.

**508 Data Communications and Networks (3) Qtr: W** Prereq: CS 354, CS 363

Principles of data communications and networks, including data transmission, data encoding, low-level protocols such as HDLC, circuit and packet switching, local area networks, high-level protocols such as TCP/IP and X.25, applications protocols such as ftp and mail, and remote procedure calls.

**509 Parallel Programming (3) Qtr: W** Prereq: CS 356 and Math 507, or consent of instructor

Examination of three models of parallel programming: (i) coroutines (non-pre-emptively scheduled shared memory threads), (ii) asynchronously interleaved tasks (pre-emptively scheduled shared memory threads), and (iii) message-based distributed computing (non-shared memory multiprocessing). Study of appropriate abstractions, language constructs, and program development methodologies for each of the three models. Examination of three principal application areas: (i) discrete state simulation, with emphasis on object-oriented modeling, (ii) client-server paradigms, with emphasis on resource management by operating systems, and (iii) distributed simulation, with emphasis on deadlock avoidance and “time warp” methods. **(Not offered 1997–98.)**

**511 Programming Languages (4) Qtr: S** Prereq: CS 356, CS 505, CS 507 (CS 507 is a corequisite for graduate students)

Study of programming paradigms and the ideas behind modern programming languages. Functional, object-oriented, and logic programming. Lambda calculus, type systems, modularity, high-level control structures.

**512 Algorithms (3) Qtr: S** Prereq: CS 354

Graduate-level survey of algorithm design and analysis.

**513 Database Systems (4) Qtr: A** Prereq: CS 356

Modeling of real world structures and their mapping into relational, network, and hierarchic schemata; the design and implementation of database systems including integrity, security and concurrency control; programming experience on a commercial database system using both data definition and data manipulation languages. Coverage ranges from proven practical techniques to current research activities. **(Not offered 1997–98.)**

**520 Architectures and Algorithms for Scientific Computing (3) Qtr: W** Prereq: CS 354, Math 353

An introduction to algorithms and architectures for the solution of large scale computational problems in science and engineering. Topics include analysis of parallel algorithms including performance issues and scalability, models of parallel computers, basic communication operations, direct and iterative numerical algorithms, overview of individual high performance architectures, and the relationship between parallel algorithms and architectures. **(Not offered 1997–98.)**

**521 Fundamentals of Scientific Computing (3) Qtr: A** Prereq: CS 354, Math 251, Math 252, or permission of instructor

An introduction to existing classical and modern numerical methods and their algorithmic development and efficient implementation. Topics include: numerical linear algebra, interpolation, approximation methods, parallel computation methods for nonlinear equations, ordinary differential equations, and partial differential equations.

**522 Advanced Methods in Scientific Computing (3) Qtr: W** Prereq: CS 521 or Math 560, Math 353

A study of the numerical solution of two and three dimensional partial differential equations which arise in science and engineering problems. Topics include: finite difference methods, finite element methods, boundary element methods, multigrid methods, mesh generation, storage optimization methods, and adaptive methods.

**523 Scientific Visualization (3) Qtr: S** Prereq: CS 354, Math 353.

An introduction to the techniques and tools needed for the visual display of data. Students will explore many aspects of visualization, using a “from concepts to results” format. The course begins with an overview of the important issues involved in visualization, continues through an overview of graphics tools relating to visualization, and ends with instruction in the utilization and customization of a variety of scientific visualization software packages.

**531 Introduction to Robotics (3) Qtr: A** Prereq: CS 202, Math 252

Basic robotics, including, position and velocity sensing, actuators, control theory, robot coordinate systems, robot kinematics, differential motions and the Jacobian, path control, kinetics, force control and compliance, robot vision, multisensor integration and robot programming languages.

**533 Artificial Intelligence (3) Qtr: A** Prereq: CS 355 or consent of instructor

An introduction to the field of artificial intelligence, including selected topics from the following: heuristic programming, problem solving, search, theorem proving, question answering, machine learning, pattern recognition, game playing, robotics, computer vision, philosophical and social issues.

**534 Natural Language Processing (3) Qtr: W** Prereq: CS 356 or consent of instructor

Computational models and methods for understanding written text. Introduction to syntactic analysis, semantic analysis, discourse analysis, knowledge structures, and memory organization. A variety of approaches are covered, including conceptual dependency theory, connectionist methods, and statistical techniques. Applications studied include information retrieval, machine translation, and speech recognition.

**537 Introduction to Computer Vision (3) Qtr: W** Prereq: CS 356, Math 252

Basic pattern recognition techniques and image analysis techniques, including, low-level representation, intrinsic images, "shape from" methods, segmentation, texture analysis, motion analysis, and the representation of 2-D and 3-D shape.

**542 Integrated Circuit Design Techniques (4) Qtr: A** Prereq: CS 363 or EE 221 or consent of instructor

Project-oriented class for the design of an LSI circuit using high level design tools. Use of high-level CAD tools such as PPL. Two-student teams design, lay out, and simulate a complete integrated circuit. Teams must conduct design reviews, give progress reports, and prepare final written reports. All projects must meet criteria for MOSIS tiny chips and are submitted for fabrication at MOSIS if students agree to test the parts when they are returned.

**543 Fundamentals of Integrated Circuit Design (4) Qtr: W** Prereq: CS 542 or consent of instructor

Introduction to basic concepts of the design of CMOS integrated circuits for students with a wide range of backgrounds. Static and dynamic properties of MOS circuits, composite layout of CMOS circuits, and modeling of transistors for use in SPICE simulations. Commonly encountered CMOS circuits. Students complete design, composite layout, and digitization of a simple integrated circuit using computer-aided design tools.

**544 Advanced VLSI Theory and Design (3) Qtr: S** Prereq: CS 542 or consent of instructor

Advanced course on the fabrication and design of GaAs MESFET integrated circuits. Concepts for design using GaAs MESFETS rather than CMOS. Discussion of a number of GaAs MESFET logic families; detailed study of DCFL design. Traditional JFET models for the GaAs MESFET transistor; advanced models such as the Hyperbolic Tangent model. Noise margins in DCFL and SCFL circuits. Introduction to design concepts for GaAs Heterojunction Bipolar Transistor (HBT) circuits.

**545 Testing of Integrated Circuits (1) Qtr: S** Prereq: Receipt of a circuit from MOSIS or design during CS 542.

Laboratory course featuring testing of chips designed during CS 542 (autumn quarter) and fabricated at MOSIS (winter quarter). Operational details of the Tektronix LV500 tester leading to actual tests on student chips. Written report required.

**547 Asynchronous VLSI System Design (3) Qtr: A** Prereq: CS/EE 361 or consent of instructor. Cross-listed as EE 547.

Introduction to systematic methods for the design of asynchronous VLSI systems from high-level specifications to efficient, reliable circuit implementations. Topics include specification, controller synthesis, optimization using timing information, technology mapping, data path design, and verification.

**548 Computer Aided Design of Digital Circuits (3) Qtr: A** Prereq: CS/EE 361 and CS 354 or consent of instructor. Cross-listed as EE 548.

Introduction to theory and algorithms used for computer-aided synthesis of digital integrated circuits. Topics include algorithms and representations for Boolean optimization, hardware modeling, combinational logic optimization, sequential logic optimization, and technology mapping.

**550 Digital and Analog Interfaces (3) Qtr: A** Prereq: EE 230, CS/EE 361, CS/EE 362

Interfacing digital and analog circuits. Computer interfacing to PCs. Fundamentals of digital-to-analog (D-to-A) and analog-to-digital (A-to-D) circuits. Basic interfacing with relays, stepper motors, and digital switches.

**551,552 Microprocessor Design Laboratory (3,3) Qtr: W,S** Prereq: CS/EE 363, CS/EE 550

Microprocessors and their interfaces, control logic design, memory devices, interrupt systems, DMA protocols, I/O systems. Cache memory systems, memory management units, digital signal processors, coprocessors, parallel processing systems, additional industry standards. This is a project class that involves the design of a microcomputer system that includes RAM, EPROM, and I/O devices. Formal written reports, as well as one or more oral presentations related to the project, are required.

**561 Advanced Computer Organization (3) Qtr: A** Prereq: CS 363 (CS 506 and 507 recommended)

Topics introduced in the 361-363 sequence are discussed in greater depth, as well as current research issues in computer architecture and machine organization. Topics include: Instruction set design and analysis, high-level language computer architecture, support for operating systems, interleaved memory systems and caches, pipelined systems, branch prediction strategies.

**562 Introduction to Parallel Computer Organization (3) Qtr: S** Prereq: CS 561

Hardware and architectural issues of parallel systems design and analysis. Array processors, associative processors, multiprocessors, systolic arrays, data flow computers, interconnection networks, scheduling techniques.

**567 Digital Signal Processing (4) Qtr: A** Prereq: EE 323

The z-transform; design of frequency selective, linear time-invariant filters using pole-zero placement; structures for implementing finite impulse response (FIR) and infinite impulse response (IIR) filters; discrete Fourier transform (DFT); fast algorithms for DFT and convolution; design of FIR and IIR filters.

**568 VLSI Architectures (3) Qtr: W** Prereq: CS 356, CS 367, CS 542

Project-based study of a variety of topics related to VLSI systems. Use of field programmable gate arrays to design, implement, and test a VLSI project.

**570-579 Topics in Computer Science (Arr)** Prereq: Consent of instructor.

Current topics in computer science. Topics will be announced. The following offerings are scheduled for 1997–98. Contact the instructor for details.

**573 Machine Learning (3) Qtr: S** Prof. Riloff.

**575 Computers and Law (Arr.) Qtr: S** Prof. Hollaar.

**575 Natural Language Processing (Arr.) Qtr: W** Prof. Riloff.

**577 High Performance Computer Architecture (Arr.) Qtr: W** Prof. Davis.

**579 Advanced Modeling and Manufacturing (4) Qtr: A** Prof. Drake.

**579 Legal Protection of Computer Software (3) Qtr: W** Prof. Hollaar.

**590 Independent Study (Arr.)** Prereq: Consent of instructor

Special reading and/or projects. May be repeated for credit.

**591-599 Seminar (Arr.)** Prereq: Consent of instructor

Current topics in computer science. Topics will be announced. May be repeated for credit.

**606 Advanced Operating Systems (3) Qtr: W** Prereq: CS 506 (CS 508 strongly recommended)

Practical distributed operating systems concepts from basics through the state of the art. Topics include interprocessor communication, remote procedure call (RPC) systems, multicast primitives, distributed shared memory, distributed file systems, portable computing, software fault tolerance, distributed databases, and process checkpointing. Work includes individual oral presentations, a group project, and a written research report.

**610 Formal Languages (4) Qtr: W** Prereq: CS 505, CS 507

Properties of various classes of formal languages, the grammars which generate them, and the abstract machines which can accept them. Relevance to the syntactic structure and parsing of programming languages discussed. **(Not offered 1997–98.)**

**611 Formal Methods for System Design (3) Qtr: A** Prereq: CS 505 (CS 511 recommended)

Study of methods for formally specifying and verifying computing systems. Specific techniques include explicit state enumeration, implicit state enumeration, automated decision procedures for first-order logic, and modern theorem-proving techniques. Examples selected from the areas of superscalar CPU design, parallel processor memory models, various distributed (e.g. coherence) protocols, and security, all highly relevant to ongoing research in the Department of Computer Science.

**612 Computational Complexity (3) Qtr: W** Prereq: CS 505

Analysis of time and memory requirements of algorithms for sorting, set manipulation, graph analysis, matrix operations, arithmetic, Fourier transforms, and pattern matching. NP complete problems. Complexity hierarchies.

**628 Robot Dynamics and Control (4) Qtr: W** Prereq: CS 531; ME 521, 531, or equivalent. Cross-listed as ME 628.

Kinematics, dynamics, and control of robots and prosthetic systems. Solving dynamics, inverse kinematics for linkage systems in real-time, control stratagems for robotic systems. Laboratory included.

**649 Real-Time Hardware Verification (3) Qtr: S** Prereq: CS/EE 363 and CS/EE 548 or consent of instructor. Cross-listed as EE 649.

Theory and algorithms used for computer-aided verification of real-time hardware systems; strong emphasis on gaining experience with existing verification tools. Topics include both finite and  $\omega$ -automata based methods, binary decision diagrams, and real-time verification.

**651 Computer Graphics (3) Qtr: A** Prereq: CS 356, Math 252, linear algebra

Basic display techniques, display devices, vector generation, display processors. Homogeneous coordinates, transformations and clipping in 2D. Graphics systems, interactive graphics. Introduction to raster graphics. Some elements of photography as related to computer graphics.

**652 Computer Graphics (3) Qtr: W** Prereq: CS 651

Representations of 3D objects, polygons, 3D visualization techniques, hidden line and hidden surface removal, polygon clipping, continuous tone pictures, color displays, lighting models, the aliasing problem. Some fundamentals of photographing computer generated gray-scale images.

**653 Computer Graphics (3) Qtr: S** Prereq: CS 652

A project course offering more detailed treatment of topics selected from those covered in CS 652. Study of most recent developments in computer graphics. Further photographic studies.

**667,668,669 Computer-Aided Geometric Design (3,3,3) Qtr: A,W,S** Prereq: CS 453 or graduate standing; Coreq: CS 651 (for 667 only)

Introduction to current concepts and issues in CAGD systems with emphasis on freeform surface design concepts. Development of the mathematics of freeform curve and surface representations including Coons patches, Bezier method, B-splines, triangular interpolants, and their geometric consequences. Classical surface geometry. Issues such as local and global design tradeoffs and explicit and parametric tradeoffs are addressed. Subdivision and refinement as techniques in modeling. Understanding of current production capabilities as compared to advanced research is developed by laboratory experiments with current CAD systems.

**670-679 Advanced Topics in Computer Science (Arr.)** Prereq: Consent of instructor.

Current topics in computer science. Topics will be announced. The following offerings are scheduled for 1997–98. Contact the instructor for details.

**672 Virtual Environments and Teleoperation (3) Qtr: S** Prof. Hollerbach.**673 Vision/Robotics Laboratory (Arr.) Qtr: S** Prof. Thompson**673 Robot Calibration (Arr.) Qtr: W** Prof. Hollerbach.**674 Computational and Numerical Methods for Inverse Problems (Arr.) Qtr: A** Prof. Johnson.**680-689 Seminar (Arr.)** Prereq: Consent of instructor

Current topics in computer science. Topics will be announced. May be repeated for credit.

**690 Independent Study (Arr.)** Prereq: Consent of instructor

Special reading and/or projects. May be repeated for credit.

**697 Thesis Research: Master's (Arr.)** Prereq: Consent of instructor**698 Research Consultation: Master's (3)** Prereq: Consent of instructor

**780-789 Seminar (Arr.)** Prereq: Consent of instructor

Current topics in computer science. Topics will be announced. May be repeated for credit.

**790 Independent Study (Arr.)** Prereq: Consent of instructor

Special reading and/or projects. May be repeated for credit.

**797 Thesis Research: Ph.D. (Arr.)** Prereq: Consent of instructor

**798 Research Consultation: Ph.D. (Arr.)** Prereq: Consent of instructor

**799 Continuing Registration: Ph.D. (0)** Prereq: Consent of instructor

# 5

## CS Faculty And Their Research Interests

### Erik Brunvand

Associate Professor of Computer Science  
Ph.D., Carnegie Mellon University, 1991

Professor Brunvand<sup>1</sup> joined the Department of Computer Science in 1990. He has interests in computer architecture and VLSI systems in general, and self-timed and asynchronous systems in particular. One aspect of his research involves compiling concurrent communicating programs into asynchronous VLSI circuits. The current system allows programs written in a subset of Occam, a concurrent message-passing programming language based on CSP, to be automatically compiled into a set of self-timed circuit modules suitable for manufacture as an integrated circuit. He is also interested in investigating the effects of asynchrony on computer systems architecture at a higher level. To explore these ideas he is building a series of prototype asynchronous computer systems out of FPGA and custom VLSI chips.

- G. Gopalakrishnan, E. Brunvand, N. Michell, and S. Nowick, “A Correctness Criterion for Asynchronous Circuit Validation and Optimization”, in *IEEE Transactions on Computer Aided Design* November 1994.
- A. Khoche and E. Brunvand, “Testing Micropipelines”, in *Symposium on Advanced Research in Asynchronous Circuits and Systems (Async94)*, November 1994.
- G. Gopalakrishnan, P. Kudva, and E. Brunvand, “Peephole Optimization of Asynchronous Macromodule Networks”, in *International Conference on Computer Design (ICCD)*, October 1994.
- P. Kudva, G. Gopalakrishnan, and E. Brunvand, “Performance Analysis and Optimization of Asynchronous Circuits”, in *International Conference on Computer Design (ICCD)*, October 1994.
- E. Brunvand, “Designing Self-Timed Systems using Concurrent Programs”, *Journal of VLSI Signal Processing*, 1994. Special issue on asynchronous systems.

---

<sup>1</sup><http://www.cs.utah.edu/~elb/>

### John Carter

Assistant Professor of Computer Science  
Ph.D., Rice University, 1992

Professor Carter<sup>2</sup> joined the Department of Computer Science in January 1993. His research interests include multiprocessor computer architecture, operating systems, distributed computing, and computer networks. Of particular interest are scalable shared memory architecture designs, both hardware and software. Dr. Carter is co-leading three DARPA-sponsored research projects: the Avalanche scalable multiprocessor architecture design effort, the Adaptable Memory Systems project, and the Flux operating system project. The goal of the Avalanche project is to develop an integrated cache, memory, and communication architecture that significantly reduces the latency of both distributed shared memory and message passing multiprocessor communication. The goal of the Adaptable Memory Systems Project is to attack the primary problem limiting performance in future computer systems—the inability of conventional memory systems to supply data fast enough to avoid processing stalls—by developing a main memory controller and associated software that allows applications to dynamically change the way that the processor’s memory hierarchy is managed. The goal of the Flux project is to develop an operating system that provides a much higher degree of flexibility than traditional operating systems, and provides a testbed for exploring other systems issues such as global caching and distributed shared memory.

- J.B. Carter, J.K. Bennett, and W. Zwaenepoel. Techniques for reducing consistency-related communication in distributed shared memory systems. *ACM Transactions on Computer Systems*, August 1995.
- J.B. Carter. Design of the Munin Distributed Shared Memory System. *Journal of Parallel and Distributed Computing*, Vol. 29, Number 2, pp. 219-227, September 1995.
- A. Saulsbury, T. Wilkinson, J. Carter, and A. Landin, An Argument for Simple COMA. In the *Proceedings of the First Annual Symposium on High Performance Computer Architecture*, January 1995; and the *Future Generation Computer Systems (FGCS)* journal, Number 11, November 1995.
- J.B. Carter, J.K. Bennett, and W. Zwaenepoel. Implementation and performance of Munin. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pages 152–164, October 1991.
- J.K. Bennett, J.B. Carter, and W. Zwaenepoel. Adaptive software cache management for distributed shared memory architectures. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pages 125–134, May 1990.

### Elaine Cohen

Professor of Computer Science  
Adjunct Associate Professor of Mathematics  
Ph.D., Syracuse University, 1974

Professor Cohen<sup>3</sup> has held a faculty position in the Department of Computer Science since 1974. Currently she is co-head of the department’s Computer-Aided Geometric Design Group. Present research is centered around representational and algorithmic problems associated with geometric modeling, graphics, scientific visualization physically based modeling, process planning, CAD/CAM and CAE. Dr. Cohen received a B.A. in Mathematics from Vassar College in 1968 and M.S. and Ph.D. degrees in mathematics from Syracuse University in 1970 and 1974, respectively.

- T. V. Thompson II, D. E. Johnson, and E. Cohen,, "Direct Haptic Rendering Of Sculptured Models," in Proc. Symposium on Interactive 3D Graphics, (Providence, RI), pp. 167-176, ACM, April 1997.
- G. Elber and E. Cohen, "Adaptive Iso-Curves Based Rendering for Free Form Surfaces," *Transactions on Graphics*, 1996, v. 15, n. 3, pp.249 - 263.
- E. Driskill and E. Cohen, "Interactive Design, Analysis, and Illustration of Assemblies," in Proc. Symposium on Interactive 3D Graphics, pp. 27-32, ACM, April 1995.

<sup>2</sup><http://www.cs.utah.edu/~retrac/>

<sup>3</sup><http://www.cs.utah.edu/~cohen/>

- G. Elber and E. Cohen, “Arbitrarily Precise Computation of Gauss Maps and Visibility Sets,” in Proc. Solid Modeling 95, (Salt Lake City, UT), Solid Modeling, May 1995.
- R. Riesenfeld, E. Cohen, S. Drake, and L. Gursoz, “Modeling Issues in Solid Freeform Fabrication”, in Proc. NSF Solid Freeform Fabrication Workshop, 1995.

### **Al Davis**

Professor of Computer Science  
Ph.D., University of Utah, 1972

Professor Davis<sup>4</sup> joined the Department in 1993. His research interests include convergence parallel processing system architectures, VLSI, VLSI CAD, high performance communication, and asynchronous circuits. Prior to his joining the faculty in the fall of 1993, he spent the previous 12 years as a research scientist working on the design and implementation of parallel processing systems at Schlumberger Palo Alto Research and subsequently at Hewlett-Packard Laboratories. Recent accomplishments include 1) the development of an automatic asynchronous circuit synthesis system called STETSON; 2) the design and implementation of an asynchronous scalable parallel communication fabric VLSI component called FEDEX which is capable of supporting 500 MB/sec sustained bandwidth on each of its 7 ports; and 3) the development of an extensible and scalable parallel processing system called MAYFLY which contains 19 processing elements and has to date exhibited scalable performance for a wide range of business and scientific applications. Current research interests include the development of low-latency communication protocols and network interface hardware (supported by Hewlett Packard), the design of a scalable parallel processor which supports flexible distributed shared memory and low-latency message passing programming models (supported by ARPA), and a novel adaptive memory system (supported by ARPA).

- A. Davis, M. Swanson, M. Parker. Efficient Communication Mechanisms for Cluster Based Parallel Computing. Springer-Verlag Lecture Notes in Computer Science #1199. Feb. 1997, pp. 1–15.
- A. Davis. *Asynchronous Digital Circuit Design*. Chapter 3: ‘Synthesizing Asynchronous Circuits: Practice and Experience’. Springer-Verlag Workshops in Computing series, April 1995, pp. 104 – 151.
- A. Davis. R2 - A Damped Adaptive Router Design. Parallel Computer Routing and Communication. Springer-Verlag Lecture Notes in Computer Science #853. May 1994, pp. 295-309.
- A. Davis, B. Coates, K. Stevens. The Post Office Experience: designing a large asynchronous chip. Integration Vol. 15, No. 3, November 1993, pp. 341-366.
- A. Davis. Mayfly: A General-Purpose, Scalable, Parallel Processing Architecture. Lisp and Symbolic Computation, Volume 5, Numbers 1/2, May 1992, pp. 7–47.

### **Ganesh C. Gopalakrishnan**

Associate Professor of Computer Science  
Ph.D., State University of New York at Stony Brook, 1986

Professor Gopalakrishnan’s<sup>5</sup> research is primarily in two areas: asynchronous circuit design and formal verification. His asynchronous design group is developing synthesis algorithms to generate asynchronous circuits from descriptions in high-level hardware description languages (currently an enhanced subset of Verilog). A tool embodying these algorithms features user-guided partitioning and complex-gate generation. His formal verification group is involved in the verification of the protocols (pertaining to distributed shared memory management as well as message passing) used in an experimental multiprocessor “Avalanche” under construction at Utah. By using verification as a design aid, we hope to not only detect protocol errors at the earliest stages of design but also provide verified models for synthesis into VLSI circuits. He is also involved in the design of a high-speed image compression chip that uses novel clock distribution methods. He is a member of IFIP working-group 10.5.

---

<sup>4</sup><http://www.cs.utah.edu/~ald/>

<sup>5</sup><http://www.cs.utah.edu/~ganesh/>

- Prabhakar Kudva, Ganesh Gopalakrishnan, and Venkatesh Akella, “An Asynchronous High Level Synthesis System Targeted at Interacting Burst-Mode Controllers”, Proceedings of the 1995 Computer Hardware Description Languages, Chiba, Japan, August, 1995.
- Jae-Tack Yoo, Ganesh Gopalakrishnan, Kent Smith, and John Mathews, “Counterflow-Clocked Pipelining Illustrated on the Design of High Speed HDTV Subband Vector Quantizer Chips”, Advanced Research on VLSI, Chapel Hill, March 1995.
- Venkatesh Akella and Ganesh Gopalakrishnan, “Specification and Validation Control Intensive ICs in hopCP”, *IEEE Transactions on Software Engineering*, Vol.20, No.6, June 1994, pages 405-423.
- Prabhat Jain and Ganesh Gopalakrishnan, “Efficient Symbolic Simulation Based Verification Using the Parametric Form of Boolean Expressions”, *IEEE Transactions on Computer Aided Design*, Vol.13, No.8, August 1994, pages 1005-1015.
- Ganesh Gopalakrishnan, Erik Brunvand, Nick Michell, and Steven M. Nowick, “A Correctness Criterion for Asynchronous Circuit Validation and Optimization”. *IEEE Transactions on Computer Aided Design*, Vol.13, No.11, November 1994, pp. 1309-1318.

### David H. Hanscom

Clinical Professor of Computer Science  
Ph.D., Case Western Reserve University, 1970

Professor Hanscom's<sup>6</sup> background is in the field of communications processor design at Sperry Univac, where he worked from 1970 to 1982. Since then he has been responsible for administering the undergraduate Computer Science program at the University of Utah. In that capacity he teaches core computer science classes, serves as faculty advisor for individual students and for the Student Chapter of the Association for Computing Machinery, participates in student recruiting activities, and serves as director of the Summer Computing Institute. His interests are in the areas of undergraduate education, computer architecture, and data communications.

- Hanscom, D.H., *Instructor's Manual to Accompany Structures and Abstractions, An Introduction to Computer Science with Pascal*, by William I. Salmon, Richard D. Irwin, 1991, 393 pages.

### Thomas C. Henderson

Professor of Computer Science  
Ph.D., University of Texas, 1979

Professor Henderson's<sup>7</sup> professional interests include artificial intelligence, computer vision and robotics. Major areas of current research are robot behavior specification, simulation, multisensor integration and sensing strategies, and parallel vision algorithms. Prior to his arrival at Utah, he was a visiting professor at the Institut National de Recherche en Informatique et en Automatique (INRIA), France, and a Research Associate at the Institut fuer Nachrichtentechnik, Deutsche Forschungs und Versuchsanstalt fuer Luft und Raumfahrt (DFVLR), West Germany.

- “Flat Surface Reconstruction Using Sonar,” Thomas C. Henderson, Mohamed Dekhil, Beat Brüderlin, Larry Schenk and Larkin Veigel, *International Journal of Robotics Research*, to appear.
- “Evolutionary Teleomorphology,” *Journal of Robotics and Autonomous Systems*, Vol. 19, No. 1, November 1996, pp. 23–32. (Thomas C. Henderson and Alexei A. Efros).
- Henderson T. C. with S. Susswein, J. Zachary, C. Hansen, P. Hinker, and G. Marsden, “Parallel Path Consistency,” *International Journal of Parallel Programming*, Vol. 20, No. 6, 1992.

<sup>6</sup><http://www.cs.utah.edu/~hanscom/>

<sup>7</sup><http://www.cs.utah.edu/~tch/>

- Henderson, T.C. with Chuck Hansen, *CAGD-Based Computer Vision*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 11, No. 11, pp. 1181-1193, 1989.
- Henderson, T.C. with Amar Mitiche, Eliot Weitz, Chuck Hansen, *Multisensor Knowledge Systems: Interpreting 3-D Structure*, International Journal of Robotics Research, Special Issue on Multisensor Integration, Vol. 7, No. 6, pp. 114-137, 1988.

### Lee A. Hollaar

Professor of Computer Science

Ph.D., University of Illinois at Urbana-Champaign, 1975

Professor Hollaar's<sup>8</sup> primary interest is in legal issues regarding computers, particularly the intellectual property protection of software and information. As a Fellow with the Committee on the Judiciary, he has advised the United States Senate on computer-related issues such as encryption, copyright and patent, and regulation of the internet. He was one of the drafters of the Utah Digital Signature Act, the first law in the world to legally recognize digital signatures, and is active in the implementation of the required infrastructure. He directed the Utah Retrieval System Architecture (URSA) project, which developed hardware and software systems to support large information retrieval systems, including a special-purpose VLSI processor for the rapid searching of text and one of the first workstation-based client-server distributed systems for information retrieval. He was also the University's director of campus networking, and continues to work in communications networks and distributed systems.

- Hollaar, L.A., *Now That the CDA's History, Let's Plan Anew*, The National Law Journal, July 14, 1997.
- Hollaar, L.A., *Justice Douglas Was Right: The Need for Congressional Action on Software Patents*, AIPLA Quarterly Journal, Winter 1996.
- Hollaar, L.A., *Method for Error Recovery in a Digital Data Communications System*, United States Patent 5,396,613, March 7, 1995.
- Hollaar, L.A., *Special-Purpose Hardware for Information Retrieval*, in *Information Retrieval, Data Structures and Algorithms*, Prentice-Hall, 1992.
- Hollaar, L.A., *Direct Implementation of Asynchronous Control Units*, IEEE Transactions on Computers, December 1982.

### John M. Hollerbach

Professor of Computer Science

Ph.D., Massachusetts Institute of Technology, 1978

Professor Hollerbach's<sup>9</sup> interests include robotics, teleoperation, virtual reality, and human motor control. Autonomous and teleoperated grasping is pursued with a pair of left/right Utah/MIT Dextrous Hands. Teleoperation research is focusing on the use of the Sarcos Dextrous Arm Master and Slave, an advanced hydraulic telemanipulator. In virtual reality, the focus is on improving the transparency and sense of immersion through better mechanical interfaces and their control, better visual and auditory displays, and sensorimotor integration. More specifically, haptic interfaces are being employed for virtual manipulation of Alpha1 CAD models and for scientific visualization. The Treadport locomotion interface is being employed for walk-in synthetic environments. Real-time architectures employing the Myrinet for low-latency networking of devices, computations, and graphics are being developed. Methods for perturbation analysis and nonlinear system identification of joint mechanical properties are being developed, for use in clinical and human operator dynamics settings.

- Hollerbach, J.M., Cohen, E., Thompson, W., Freier, R., Johnson, D., Nahvi, A., Nelson, D., Thompson, T., and Jacobsen, S., "Haptic interfacing for virtual prototyping of mechanical CAD designs," *ASME Design for Manufacturing Symposium*, Sacramento, Sept. 14-17, 1997.

---

<sup>8</sup><http://www.cs.utah.edu/~hollaar/>

<sup>9</sup><http://www.cs.utah.edu/~jmh/>

- Hollerbach, J.M., and Wampler, C.W., “The calibration index and taxonomy of kinematic calibration methods,” *Intl. J. Robotics Research*, 14, 1996, pp. 573-591.
- Johnston, D., Zhang, P., Hollerbach, J.M., and Jacobsen, S.C., “A full tactile sensing suite for dextrous robot hands and use in contact force control,” *Proc. IEEE Intl. Conf. Robotics and Automation*, Minneapolis, April 22-28, 1996, pp. 3222-3228.
- Thompson, T.V. II, Nelson, D.D., Cohen, E., and Hollerbach, J., “Dynamic models with a haptic virtual environment,” *6th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Dallas, Nov. 15-21, 1997.
- Xu, Y., and Hollerbach, J.M., “Nonlinear time-varying identification of human arm joint mechanical properties using a portable pneumatic thruster,” *Proc. American Control Conf.*, Albuquerque, June 4-6, 1997, pp. 3281-3285.

### Wilson C. Hsieh

Assistant Professor of Computer Science  
Ph.D., Massachusetts Institute of Technology, 1995

Professor Hsieh<sup>10</sup> joined the Department of Computer Science in September 1997. His research interests are in compilers, programming languages, and systems. His primary interest lies in how compiler and programming language technology can be used to build fast, flexible systems—both uniprocessor and multiprocessor systems. For the past few years he has been investigating the use of high-level languages and dynamic code generation in building extensible operating systems. Prior to joining the faculty, Professor Hsieh was a postdoctoral research associate at the University of Washington, where he worked on the SPIN extensible operating system.

- Wilson C. Hsieh, M. Frans Kaashoek, and William E. Weihl, “Dynamic Computation Migration in Distributed Shared Memory Systems.” In *Proceedings of Supercomputing '96*, November 1996.
- Dawson R. Engler, Wilson C. Hsieh, and M. Frans Kaashoek, “‘C: A Language for Efficient, Machine-independent Dynamic Code Generation.” In *Proceedings of the 23rd Symposium on Principles of Programming Languages*, January 1996.
- Deborah A. Wallach, Wilson C. Hsieh, Kirk L. Johnson, M. Frans Kaashoek, and William E. Weihl, “Optimistic Active Messages: A Mechanism for Scheduling Communication with Computation.” In *Proceedings of the Fifth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, July 1995.
- Wiebren de Jonge, M. Frans Kaashoek, and Wilson C. Hsieh, “The Logical Disk: A New Approach to Improving File Systems.” In *Proceedings of the 14th Symposium on Operating Systems Principles*, December 1993.
- Wilson C. Hsieh, Paul Wang, and William E. Weihl, “Computation Migration: Enhancing Locality in Distributed-Memory Parallel Systems.” In *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, May 1993.

### Christopher R. Johnson

Associate Professor of Computer Science  
Research Assistant Professor of Physics  
Adjunct Assistant Professor of Mathematics and Bioengineering  
Ph.D., University of Utah, 1989

Professor Johnson's<sup>11</sup> research interests are in the area of scientific computing. Particular interests include inverse and imaging problems, adaptive methods for partial differential equations, automatic mesh generation, numerical analysis, large scale computational problems in medicine, and scientific visualization. In 1992, Professor Johnson was awarded a Young Investigator's Award from the NIH, in 1994 he was awarded the National Young Investigator (NYI) Award from the NSF, and in 1995 he was awarded the Presidential Faculty Fellow (PFF) Award from the NSF. He directs the Center for Scientific Computing and Imaging and is Co-Director of the Computational Engineering and Science Program.

<sup>10</sup><http://www.cs.utah.edu/~wilson/>

<sup>11</sup><http://www.cs.utah.edu/~crj/>

- S.G. Parker, D. Beazley, and C.R. Johnson. Computational steering software systems and strategies. IEEE Computational Science and Engineering, 1997 (to appear).
- C.R. Johnson. Computational and numerical methods for bioelectric field problems. Critical Reviews in BioMedical Engineering, pp. 1-104, 1997 (to appear)
- R.N. Klepfer, C.R. Johnson and R.S. MacLeod. The effects of inhomogeneities and anisotropies on electrocardiographic fields: A three-dimensional finite element study. IEEE Transactions on Biomedical Engineering, 1997 (to appear).
- S.G. Parker, D.M. Weinstein, and C.R. Johnson. The SCIRun computational steering software system. Modern Software Tools in Scientific Computing, edited by E. Arge, A. M. Bruaset and H. P. Langtangen, Birkhauser Press, pp. 1-44, 1997
- Y. Livnat, H.W. Shen and C.R. Johnson. A near optimal isosurface extraction algorithm for structured and unstructured grids. IEEE Transactions on Visual Computer Graphics, vol. 2, no.1, pp. 73-84, 1996.

### Robert Kessler

Associate Professor of Computer Science  
Ph.D., University of Utah, 1981

Professor Kessler<sup>12</sup> is director of the Center for Software Science. He is interested in research and development of programming languages – mainly compilers and run-time systems for Lisp, C, and C++, operating systems – work on the Mach and BSD systems, and parallelism – parallel programming. A recent project was just completed in conjunction with the Hewlett Packard Research Labs to design and implement the software system for the Mayfly, a new, distributed memory, parallel processor. The software included two new parallel programming languages: Concurrent Scheme and Distributed C++. Current ARPA-sponsored work is in support of multi-lingual programming and persistence in Lisp. Professor Kessler also has interests in object-oriented programming, and expert system technology applied to software problems. He is Co-Editor-in-Chief of the *International Journal on Lisp and Symbolic Computation* and is general chair for the 1994 Lisp and Functional Programming Conference.

- “Persistent Immutable Shared Abstractions,” in “Parallel Symbolic Computing: Languages, Systems, and Applications” (US/Japan Workshop Proceedings), Springer-Verlag Lecture Notes in Computer Science 748, November 1993 (with B. Yih and M. Swanson).
- “Compiling Distributed C++,” the Fifth IEEE Symposium on Parallel and Distributed Processing, Dallas, Texas, December 1993, (with H. Carr and M. Swanson), pp 496-503.
- “Allocation of Parallel Programs with Time Variant Resource Requirements,” 1993 International Conference on Parallel Processing, August 1993, (with J. Evans).
- “The Design of the Schizophrenic Workstation System”, Proceedings of the 1993 Usenix Mach Workshop, Santa Fe, New Mexico, April 1993, pp 291-306, (with M. Swanson, L. Stoller, and T. Critchlow).
- “Implementing Concurrent Scheme for the Mayfly Distributed Parallel Processing System,” International Journal on Lisp and Symbolic Computation, Vol 5:1/2, January 1992, (with H. Carr, L. Stoller, M. Swanson).

### Jay Lepreau

Assistant Director, Computer Systems Laboratory  
B.S., University of Utah, 1983

Jay Lepreau<sup>13</sup> has been leading research in the Department of Computer Science since 1992 as the Assistant Director of the Computer Systems Laboratory. His research interests focus on operating systems, including many other research areas

<sup>12</sup><http://www.cs.utah.edu/~kessler/>

<sup>13</sup><http://www.cs.utah.edu/~lepreau/>

related to building secure, flexible, and high performance systems. These include information security, programming and domain-specific languages, compilers, networks, distributed systems, and software assurance and engineering. He is Principal Investigator of two DARPA-sponsored research grants focusing on developing a secure, flexible, and high-performance operating system, including user-level but strong management of arbitrary resources, such as memory and the cpu. In this effort, much software has been developed by his group, including the Flick IDL compiler, the OSKit, the Fluke operating system, and the OMOS linker and object server.

- Bryan Ford, Godmar Back, Greg Benson, Jay Lepreau, Albert Lin, and Olin Shivers. The Flux OSKit: A Substrate for OS and Language Research. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, October 1997.
- Eric Eide, Kevin Frei, Bryan Ford, Jay Lepreau, Gary Lindstrom. Flick: A Flexible, Optimizing IDL Compiler. In *Proceedings of the ACM SIGPLAN 1997 Conference on Programming Language Design and Implementation*, June 1997.
- Bryan Ford, Mike Hibler, Jay Lepreau, Patrick Tullmann, Godmar Back, Stephen Clawson. Microkernels Meet Recursive Virtual Machines. In *Proceedings of the Second Usenix/ACM/IEEE Symposium on Operating Systems Design and Implementation*, October 1996.
- Patrick Tullmann, Jeff Turner, John McCorquodale, Jay Lepreau, Ajay Chitturi, Godmar Back. Formal Methods: A Practical Tool for OS Implementors. In *Proceedings of the Sixth IEEE Workshop on Hot Topics in Operating Systems*, May 1997.
- Jay Lepreau, Bryan Ford, Mike Hibler. The Persistent Relevance of the Local Operating System to Global Applications. Presented at and appears in *Proceedings of the Seventh ACM SIGOPS European Workshop*, September 1996.

### Gary E. Lindstrom

Professor of Computer Science  
Ph.D., Carnegie-Mellon University, 1971

Professor Lindstrom's<sup>14</sup> research interests include programming languages, databases, and parallel and distributed computing. He is on the editorial board of *International Journal of Parallel Programming*, and was Editor-in-Chief from its founding until 1993. With Doug DeGroot, he co-edited the book *Logic Programming: Functions, Relations and Equations* published by Prentice-Hall. Professor Lindstrom has been a member of the National Science Foundation Computer and Computation Research Advisory Committee, and served as a Distinguished Visitor of the IEEE Computer Society. In 1981 he received the College of Engineering Outstanding Teaching Award.

- Jon Oler, Gary Lindstrom, and Terence Critchlow. Migrating relational data to an OODB: strategies and lessons from a molecular biology experience. In *Proc. of the Conference on Object-Oriented Programming, Systems, Languages, and Applications*, Atlanta, GA, October 1997. ACM SIGPLAN.
- Eric Eide, Kevin Frei, Bryan Ford, Jay Lepreau, and Gary Lindstrom. Flick: a flexible, optimizing IDL compiler. In *Proc. of the Symposium on Programming Language Design and Implementation*, pages 44–56, Las Vegas, NV, June 1997. ACM SIGPLAN.
- Guruduth Banavar and Gary Lindstrom. An application framework for modular composition tools. In *Proc. of the European Conference on Object-Oriented Programming*, pages 91–113, Linz, Austria, July 1996. Springer LNCS 1098.
- Rob Sargent, Dave Fuhrman, Terence Critchlow, Tony Di Sera, Robert Mecklenburg, Gary Lindstrom and Peter Cartwright. The design and implementation of a database for human genome research. In *Proc. of the Eighth International Conference on Scientific and Statistical Database Management*, pages 220–225, Stockholm, June 1996. IEEE Computer Society Press.

---

<sup>14</sup><http://www.cs.utah.edu/~gary/>

- Guruduth Banavar, Douglass Orr and Gary Lindstrom. Layered, Server-based Support for Object-Oriented Application Development. In *Proc. of the International Workshop on Object-Oriented Operating Systems*, Lund, Sweden, August 1995.
- Robert Mecklenburg, Charles Clark, Gary Lindstrom, and Benny Yih. A dossier driven persistent objects facility. In *Proc. of the C++ Conference*, pages 265–281, Cambridge, MA, April 1994. USENIX Association.

### **Richard F. Riesenfeld**

Professor of Computer Science  
Ph.D., Syracuse University, 1973

Professor Riesenfeld<sup>15</sup> has been involved in research in the areas of computer graphics, animation, computer aided geometric design and CAD/CAM since joining the Computer Science faculty in 1972. Recently he has been investigating a broad spectrum of research problems in computer graphics, geometric modeling, and manufacturing within an integrated experimental testbed system motivated by the unifying principles of spline theory.

- R.F. Riesenfeld, R. Fish, and S. Drake, “A Case Study in Multi-disciplinary Distributed Collaborative Design,” in *Proc. ASME Conference on Network-Centric CAD*, in press, Sept. 1997.
- M. Sturgill, E. Cohen, and R. Riesenfeld, “Feature Based 3-D Sketching for Early Design,” *Proceedings of the 1995 ASME Computers in Engineering Conference*
- R. Riesenfeld, “Some Video Related High Bandwidth Communications Needs,” in *Proc. NSF Workshop on vBNS and the Research Agenda for Networking and Applications*, June 1995.  
*Fundamental Developments of Computer Aided Geometric Design*, L. Piegl (ed.), Academic Press, 1993.
- Y.C. Hsieh, R. F. Riesenfeld, and S. H. Drake, “Reconstruction of Sculptured Surfaces using Coordinate Measuring Machine”, *ASME Design Automation Conference*, 1993.

### **Ellen M. Riloff**

Assistant Professor of Computer Science  
Ph.D., University of Massachusetts at Amherst, 1994

Professor Riloff’s<sup>16</sup> research interests are in the areas of natural language processing (NLP), information retrieval, and artificial intelligence. Her current research projects include conceptual sentence analysis, information extraction, corpus-based knowledge acquisition, and text categorization and segmentation. The NLP group at Utah is currently building a conceptual natural language processing system called Sundance, which consists of corpus-based components that can be easily adapted for different domains. Professor Riloff also works on corpus-based systems for generating extraction patterns automatically (the AutoSlog and AutoSlog-TS systems), for building semantic lexicons, and for NLP-based text categorization and segmentation.

- Riloff, E. and Shepherd, J., A Corpus-Based Approach for Building Semantic Lexicons, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*. 1997.
- Riloff, E., Automatically Generating Extraction Patterns from Untagged Text, *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*. 1996.
- Riloff, E., An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains, *Artificial Intelligence*. Volume 85. 1996.
- E. Riloff. Little Words Can Make a Big Difference for Text Classification. To appear in *Proceedings of the Eighteenth International Conference on Research and Development in Information Retrieval (SIGIR-95)*. 1995.
- E. Riloff and W. Lehnert. Information Extraction as a Basis for High-Precision Text Classification. *ACM Transactions on Information Systems*, Vol. 12, No. 3, 1994.

<sup>15</sup><http://www.cs.utah.edu/~rfr/>

<sup>16</sup><http://www.cs.utah.edu/~riloff/>

### Peter Shirley

Assistant Professor of Computer Science  
Ph.D., University of Illinois, 1990

Professor Shirley<sup>17</sup> joined the Department in 1996. He is interested in creating highly realistic images of virtual environments, and visualization of complex data. The former involves explicit and procedural generation of geometric models with realistic optical characteristics, light transport simulation to determine the outgoing light distribution from surfaces, and tone reproduction to create images displayable on low dynamic range media such as paper and CRTs. The latter involves issues of visual representation of complex data, as well as strategies for navigation and interaction that help the user extract local and global information about the data.

- J.A. Ferwerda, S.N. Pattanaik, P. Shirley, and D.P. Greenberg. An Adaptation Model for Realistic Image Synthesis. In *ACM SIGGRAPH Annual Conference*, 1996.
- P. Shirley, C. Wang, and K. Zimmerman. Monte Carlo Methods for Direct Lighting Calculations. *Transactions on Graphics*, January 1996.
- G. Spencer, P. Shirley, K. Zimmerman, and D.P. Greenberg. An Adaptation Model for Realistic Image Synthesis. In *ACM SIGGRAPH Annual Conference*, 1995.
- P. Shirley, B. Wade, P. Hubbard, D. Zareski, B. Walter, D.P. Greenberg. Global Illumination via Density-Estimation. In *Sixth Eurographics Rendering Workshop*, 1995.
- P. Maciel, and P. Shirley. Interactive Navigation of Large Environments using Textured Clusters. In *ACM Symposium on Interactive Graphics*, 1995.

### Kris Sikorski

Associate Professor of Computer Science  
Ph.D., University of Utah, 1982

Professor Sikorski's<sup>18</sup> current research interests are in the areas of distributed parallel scientific computation and computational complexity with emphasis on information based complexity. Of specific interest are applied problems in geophysics (3-D modeling of earthquakes), combustion (fluid mechanics), and electromagnetic wave propagation (Maxwell equations). Various parallel explicit and implicit algorithms are being studied and implemented on massively parallel machines. Information based complexity is a study of optimal algorithms for problems which are approximately solved, because of partial and contaminated information. Optimal algorithms for solving nonlinear problems with use of various error criteria are of special interest to Professor Sikorski.

- Sikorski, K., Tsay, C., and Wozniakowski, H., *An Ellipsoid Algorithm for the Computation of Fixed Points*, Journal of Complexity, March, 1993.
- Sikorski, K., Schuster, J., and Tsay, C., *3-Dimensional Modeling of Acoustic and Elastic Wave Propagation on Parallel Computers*, Transputer Research and Applications 1, IOS Press, 1990.
- Sikorski, K., and Wozniakowski, H., *Complexity of Fixed Points I*, Journal of Complexity, December, 1987.
- Boulton, T., and Sikorski, K., *Complexity of Computing Topological Degree of Lipschitz Functions in N-Dimensions*, Journal of Complexity, Vol. 2, pp. 44-59, 1986.
- Sikorski, K., and Wozniakowski, H., *For Which Error Criteria Can We Solve Nonlinear Equations*, Journal of Complexity, 1986.

---

<sup>17</sup><http://www.cs.utah.edu/~shirley/>

<sup>18</sup><http://www.cs.utah.edu/~sikorski/>

### Kent F. Smith

Professor of Computer Science  
 Professor of Electrical Engineering  
 Ph.D., University of Utah, 1982

Professor Smith's<sup>19</sup> principal interests lie in the design of integrated circuits using computer aids for structured logic. His present research is focused on techniques for high speed GaAs integrated circuits. This research involves the detailed design and fabrication of the actual integrated circuits as well as Computer Aided Design (CAD) tools for the design of these circuits. The GaAs circuits he is working with operate at speed in excess of 10 GHz and thus present problems involving very difficult analysis. For example, high speed design requires that the components be physically placed near each other to give optimum performance. In addition, the actual interconnecting wires between logic gates must be characterized and used in the analysis. The extraction and use of these high speed parameters represents problems that require new methods of design. Prior to joining the University of Utah faculty, Professor Smith was responsible for integrated circuit design and testing at the Microcircuit Laboratory at the University of Utah Research Institute. Prior to that time he was the technical director for Electrical Engineering at the General Instrument Advanced Microelectronics Lab. He holds a number of patents in circuits primarily concerning MOS and bipolar integrated circuits.

- Carter, Tony and Smith, K. F., et. al. *Cell Matrix Methodologies for Integrated Circuit Design* Integration, The VLSI Journal, July, 1989.
- Smith, K.F. and Gu, Jun, *A structured approach for VLSI Circuit Design*, IEEE Computer, Vol 22, No. 11, November, 1989, pages 9-22.

### Frank Stenger

Professor of Computer Science  
 Ph.D., University of Alberta, 1965

Professor Stenger's<sup>20</sup> research interests include the development of new methods of computation and the computer solution of computationally difficult problems from science and engineering, such as inverse problems, crack problems, flow problems and heat problems. He developed the Sinc methods, which provide optimal algorithms in all areas of engineering computation. He is currently writing, jointly with Michael O'Reilly and Tao Zhang, a *Sinc Tool Box* computer-based tutorial package to make these methods accessible to users. One of his students (Kenneth Parker) has recently completed his Ph.D. dissertation *PTOLEMY: A Sinc-Collocation Mapping Sub-System*, which is a computer sub-package of *Maple* that automates the solution of partial differential equations. Several of his students have recently written or are presently writing computer packages for solving a broad range of difficult engineering problem—such as Navier-Stokes equations (Barkey and Vakili, Narasimhan), Maxwell equations (Naghsh-Nilchi) based on his recently discovered Sinc method of approximating indefinite convolutions, which leads to a unified approach for solving elliptic, parabolic, and hyperbolic differential equations. Another student (Ross Schmittlein) is writing a package based on Sinc, constructing conformal maps. Stenger and his former student O'Reilly have been developing methods which make it possible to invert the Helmholtz equation without computing the forward solution. During the next few academic years he expects to extend these inversion methods so that they can be applied to rendering, to visualization, to the determination of paths for robots, and to the inversion of heat and electromagnetic problems for medical and geophysical applications. Seven of his papers have been accepted for publications during this past academic year.

- R. Kress, I.H. Sloan, and F. Stenger, *A Sinc Quadrature Method for the Double-Layer Integral Equation in Planar Domains With Corners*, with R. Kress and I.H. Sloan, to appear in volume honoring P.M. Anselone's 65<sup>th</sup> birthday.
- F. Stenger, B. Keys, M. O'Reilly, and K. Parker, *ODE-IVP—PACK, via Sinc Indefinite Integration and Newton Iteration*, to appear in "Numerical Algorithms".
- F. Stenger, *A Unified Approach to the Approximate Solution of PDE*, to appear in "Communications in Applied Analysis", 1998.

<sup>19</sup><http://www.cs.utah.edu/~k-smith/>

<sup>20</sup><http://www.cs.utah.edu/~stenger/>

- F. Stenger, R. Kress, and I.H. Sloan, *Constructing Conformal Maps via Sinc Methods*, prepared for the Cyprus Proceedings on Computational Complex Variables, Oct., 1997.
- A. Naghsh-Nilchi and F. Stenger, *Solution of the Electric Field Integral Equations via Sinc Convolution*, submitted.

### William B. Thompson

Professor of Computer Science  
Ph.D., University of Southern California, 1975

Professor Thompson's<sup>21</sup> primary research interest is in the area of computer vision. As part of this work, he has been active in the exploration of techniques for analyzing visual motion. Currently, he is exploring problems in sensing for manufacturing and in vision-based navigation, with a particular interest in how higher-level problem solving and lower-level perception can interact. Professor Thompson joined the University of Utah in 1991 after 16 years in the Computer Science Department at the University of Minnesota.

- W.B. Thompson, H.J de St. Germain, T.C. Henderson, and J. C. Owen. Constructing High-Precision Geometric Models from Sensed Position Data. In *Proceedings of the ARPA Image Understanding Workshop*, February 1996.
- K.T. Sutherland and W.B. Thompson. Localizing in unstructured environments: Dealing with the errors. *IEEE Transactions on Robotics and Automation*, December 1994.
- W.B. Thompson and H.L. Pick, Jr. Vision-based navigation. In *Proceedings of the ARPA Image Understanding Workshop*, April 1993.
- W.B. Thompson and J.S. Painter. Qualitative constraints for structure-from-motion. *Computer Vision, Graphics, and Image Processing B: Image Understanding*, July 1992.
- W.B. Thompson and T.C. Pong. Detecting moving objects. *International Journal of Computer Vision*, January 1990.

### Joseph L. Zachary

Clinical Associate Professor of Computer Science  
Ph.D., Massachusetts Institute of Technology, 1987

Professor Zachary<sup>22</sup> joined the Department in 1987. He is interested in finding ways to use computers in teaching science and engineering in general, and computer science in particular. He is currently developing Web-based tools and curricula for teaching introductory programming and computer science courses. Professor Zachary received the University of Utah Distinguished Teaching Award in 1997, was named a Department of Energy Undergraduate Computational Science Education Award winner in 1996, was a University of Utah Presidential Teaching Scholar in 1995, and received the College of Engineering Outstanding Teaching Award in 1990.

- J.L. Zachary. *Introduction to Scientific Programming: Computational Problem Solving Using Mathematica and C*. TELOS/Springer-Verlag, 1998.
- J.L. Zachary. *Introduction to Scientific Programming: Computational Problem Solving Using Maple and C*. TELOS/Springer-Verlag, 1996.
- J.L. Zachary. The gestalt of scientific programming: Problem, model, method, implementation, assessment. In *Papers of the Twenty-Eighth SIGCSE Technical Symposium*. ACM Press, Feb 1997.
- A.H. Lee and J.L. Zachary. Reflections on metaprogramming. *Transactions on Software Engineering*, November 1995.
- J.L. Zachary. Tutorial-based teaching of introductory programming classes. In *Papers of the Twenty-Fifth SIGCSE Technical Symposium on Computer Science Education*. ACM Press, March 1994.

<sup>21</sup><http://www.cs.utah.edu/~thompson/>

<sup>22</sup><http://www.cs.utah.edu/~zachary/>

# 6

## Current Funded Research

As an indication of the breadth and specialties of current research activity in the department, the following list summarizes a partial list of funded projects underway at the time of this handbook's publication.

- *Application Driven Advancement of Asynchronous Design Methods*, NSF, Erik Brunvand and Ganesh Gopalakrishnan.
- *Center of Excellence for Asynchronous Circuit and System Design*, State of Utah, Erik Brunvand and Chris Myers.
- *Communication and Memory Architectures for Scalable Parallel Computing*, ARPA, Al Davis.
- *Applying Hardware Verification Techniques to Industrial-Scale Problems*, DARPA, Ganesh Gopalakrishnan.
- *A Multi-Paradigm Verification System Tailored for the Design Refinement Cycle*, NSF, Ganesh Gopalakrishnan.
- *Graduate Research Traineeships*, NSF, Thomas Henderson and Gary Lindstrom.
- *Acquisition of Computational Steering Instrumentation*, NSF, Thomas Henderson.
- *vBNS Connection to the Internet*, NSF, Thomas Henderson.
- *Medical Robotics*, UBC, John Hollerbach.
- *Rapid Virtual Prototyping of Mechanical Assemblies*, NSF, John Hollerbach.
- *Equipment for Simultaneous Local and Global Interactive Scientific Vision*, Army Research Office, Chris Johnson.
- *Presidential Faculty Fellowship*, NSF, Chris Johnson.
- *Inverse Electroencephalography*, University of Utah Research Foundation, Chris Johnson.
- *Center of Excellence "Center for Scientific Computing and Imaging"*, State of Utah, Chris Johnson.
- *Portable 3D Imaging Ultrasound System on 1.5D and 1.0D Integrated Arrays*, DARPA, Chris Johnson.
- *Fast and Flexible Mach-Based Systems*, ARPA, Jay Lepreau.
- *Mach 4 Kernel and IDL Infrastructure for Security*, DARPA, Jay Lepreau.

- *Object Management - AASERT Traineeships*, ARPA, Gary Lindstrom.
- *Multiphase Integrated Engineering Design (MIND)*, ARPA, Richard Riesenfeld and Elaine Cohen.
- *Computer Graphics and Scientific Visualization*, NSF Science and Technology Center, Richard Riesenfeld and Elaine Cohen.
- *CAREER: Building Conceptual Natural Language Processing Systems for Practical Applications*, NSF, Ellen Riloff.
- *Information Extraction as a Basis for Multi-Faceted Text Categorization Systems*, NSF, Ellen Riloff
- *Realistic Computer Rendering of Complete Scenes Using Quantitative Perceptual Error Metrics*, NSF, Peter Shirley
- *Rapid Prototyping of Analog/Digital Circuits*, FBI, Kent Smith.
- *Quicker PPL Cell Design Changes*, Maryland Procurement, Kent Smith.
- *SINC Algorithm Development*, NSF, Frank Stenger.
- *CISE Research Infrastructure: Modeling Complex Physical and Computational Environments*, NSF, William Thompson.
- *Extraction of Micro-Terrain Features*, DARPA, William Thompson.

# Graduate Application Forms

Postscript copies of the application forms<sup>1</sup> required of applicants to the Department are available.

---

<sup>1</sup>[http://www.cs.utah.edu/csinfo/new\\_grad\\_info.html](http://www.cs.utah.edu/csinfo/new_grad_info.html)