Conduit

A RESEARCH AND ALUMNI NEWS MAGAZINE
DEPARTMENT OF COMPUTER SCIENCE
BROWN UNIVERSITY

The CRA-E White Paper:
Some Additional Perspectives

ALSO INSIDE:

Notes from the Chair: the Latest News from 115 Waterman

Greetings to all CS alums, supporters and friends.

The fall semester is well underway and the CIT is a hub of activity. Terrific things continue to happen in the department and I am thrilled to be able to share the highlights with you.

Congratulations are in order for Chad Jenkins, who was promoted to Associate Professor with tenure, effective July 1, 2010. Chad's research addresses problems in robot learning and human-robot interaction. He was selected as a Sloan Research Fellow in 2009. He is the recipient of a Presidential Early Career Award for Scientists and Engineers (PECASE) for his work in physics-based human tacking and of a CAREER Award from the National Science Foundation for his work on robot learning from multivalued human demonstrations. He also received Young Investigator awards from the Office of Naval Research (ONR) and from the Air Force Office of Scientific Research (AFOSR).

In recent months, several of our faculty members have been bestowed with honors:

- Michael Black was awarded the 2010 Koenderink Prize for Fundamental Contributions in Computer Vision. The Prize was given at the European Conference on Computer Vision (ECCV) in Hersonissos, Crete for work that has withstood the test of time.
- Sorin Istrail was given an honorary professorship (Professor Honoris Causa) by his alma mater, Alexandru Ioan Cuza University in Iasi, Romania as part of the celebration of its 150th anniversary.
- Eli Upfal was selected as Chalmers Jubilee Distinguished Visiting Professor for 2010. Chalmers is a Swedish university of technology in which research and teaching are conducted on a broad front within technology, natural science and architecture.
- Maurice Herlihy has been awarded the Fulbright Distinguished Chair in the Natural Sciences and Engineering Lecturing Fellowship for the 2010-2011 academic year. He is visiting the Technion in Haifa, Israel, working on multiprocessor synchronization.

The Artemis Project, the departmental outreach program led by Amy Greenwald, recently received funding from the NSF, Google, Microsoft Research and The National Center for Women & Information Technology (NCWIT) via the NCWIT Academic Alliance Seed Fund. These awards will be used to expand Artemis into new directions and to track outcomes of former Artemis participants. In the summer of 2010, in addition to the four coordinators from Brown, a Boston University student was also hired with the goal of expanding the program to the greater Boston area in future years.

I am also excited to report that enrollment in our introductory classes continues to rise. The number of students taking CS15 has grown by 104% since 2007 while the CS17 enrollment has increased 85% in this timeframe. We are delighted to see so many students taking an interest in our curriculum.

In other news, the Industrial Partners Program has been revamped and now includes a start-up level as well as the option of a collaborative membership with Engineering. Ten new partners joined the program this Fall and we are thrilled to provide more even more job opportunities for our students. We'd also like to congratulate our colleagues in the recently created School of Engineering on their new designation!

I also wanted to let you know that we've set up regional Brown CS pages and an Artemis page on Facebook so that you connect with other CS alums and current students. More information can be found on page 47.

Finally, we urge you to contribute your professional and personal stories for inclusion in upcoming issues of Conduit. Your support of and participation in departmental activities are always appreciated and we are grateful to have such a tight-knit community— thank you!

Roberto Tamassia
Plastech Professor of Computer Science
Chair, Department of Computer Science

Volume 19, Number 2

Fall 2010

Published by the Department of Computer Science at Brown University, *Conduit* is distributed free to all computer science alumni, faculty, staff, students and industrial partners.

Queries about this publication can be directed to: conduit@cs.brown.edu or
115 Waterman Street
Campus Box 1910
Providence, RI 02912

PUBLISHER
Industrial Partners Program,
Department of Computer Science

EDITOR-IN-CHIEF
Amy Tarbox

GRAPHIC DESIGN
Brown Graphic Services

FEATURED COLUMNISTS
Sorin Istrail
Shriram Krishnamurthi
Rodrigo Fonseca
Andy van Dam & Rosemary M. Simpson

CONDUIT ON THE WEB
John Bazik
Kathy Kirman

PRINTING AND DISTRIBUTION
Brett Turner, Information Services
Brown Graphic Services

Cover image by Thomas Doeppner, also the cover image for his new book, *Operating Systems in Depth* (Wiley).

Conduit is printed on Burgo's ChorusArt, an acid free and elemental chlorine free paper containing 50% recycled content and 15% post-consumer waste. You are the difference – reduce, reuse, recycle.

The CRA-E White Paper: Some Additional Perspectives

By Andy van Dam & Rosemary M. Simpson

In the Spring of 2008, the CRA (Computing Research Association) asked Andy to form and chair a committee to consider the question of how to provide guidelines to educators, administrators and government (funding) agencies for the college-level education of future computationally oriented researchers in both Computer Science and in other disciplines making significant use of computation and data.

The CRA-E (CRA-Education) committee was formed with this mission statement:

> "Our charter is to explore the issues of undergraduate education in computing and computational thinking for those who will do research in disciplines from the sciences to the humanities.
>
> As technology and teaching methodologies continue to evolve, how should programs in computer science, computational science, and information science co-evolve? Can we communicate a core set of ideas, principles, and methodologies that is domain-independent?"

When the committee first met at the bi-annual CRA meeting in Snowbird Utah during the summer of 2008, they identified several major themes that they wanted to address relative to this charter. These themes included: identification of a lean core and pathways for addressing specialized interests, including fully integrated joint majors; persistent skills that would be needed throughout a computationally oriented researcher's career; issues of mastery across the curriculum; how to attract students into the field and research practices of computer science; and innovative ways in which computer technology could be used to support pedagogical goals.

Over the next two years, the committee investigated and wrote about these themes in a white paper, accessible on the CRA website. The recommendations are outlined in a sidebar at the end of this article. Unfortunately, because of time and manpower constraints we were not able to address the role of educational technology in the white paper. Since this area has been both a personal and professional interest of Andy's since graduate school, and since it played a significant role in the growth and history of the Brown Computer Science department, we have decided to address some of the pedagogical and educational technology issues, including Brown's historic and current role, and conclude with some observations about the state of educational technology and a few grand challenge problems in this important area.

Pedagogy

Pedagogy, which comprises the goals and methods of teaching and learning, precedes and motivates the use of educational technology. Arguably, the most important educational decisions must come from this concern, and must address the fact that people learn in many different ways. The lines between who is a teacher and who is a learner and between formal and informal learning, which have

always been blurred, are becoming increasingly more so. Rapid change in what is to be taught and learned and how it is to be taught and learned demands flexibility. Thus, the CRA-E white paper recommends a lean core that focuses on the core concepts and cognitive skills that undergird changing subject matter. By building a framework for lifelong learning with these elements, students are better prepared to adapt to whatever circumstances they face in the future.

Successful teaching of persistent core concepts and cognitive skills necessitates understanding of the difficulties that students encounter with the changing levels and kinds of abstractions involved with modeling, representation, and mapping between process, data, and structure at different levels. For example, understanding an algorithm and then representing it in the formal symbolic language of a program is difficult and needs constant practice. Shriram Krisnamurthi's middle school Bootstrap curriculum uses programming games as a motivation to help students see symbolic and structural parallels between program structure and math structure. Other approaches include Phil Klein's intertwining of three areas for each topic in CS-53 (computational linear algebra) - math, algorithms, and applications - in a spiral approach that builds understanding by continually revisiting earlier concepts in new contexts.

Some points to consider include:

- Knowing a core set of concepts and skills facilitates flexibility in educational approaches, as exemplified by our quite different gateway course sequences that offer students great choice - CS 15/16, 17/18, and 19.
- Learning the language of computer science - computational thinking - across different subjects and different methods helps build more powerful mental models.
- Representing a concept in different ways can result in what Roberto Tamassia refers to as "deep understanding"; one of the strengths of the Balsa algorithm animation system was its ability to present different simultaneous visualizations of an algorithm.

Fig 1: Immune Attack game showing game controller at bottom of screen.

- Scalability issues arise in pedagogy as in everything else, e.g., how to give individual attention in very large onsite classes, let alone online classes. Brown's pioneering and highly successful use of UTAs is one approach that provides greatly increased individual attention to students needing help and just as importantly, enriches the student UTAs themselves. The UTA system is an example of the creative blurring of the line between teaching and learning.
- Incorporating the informal learning process within the formal class structure is an evolving process. Students today are online learners who are teaching themselves through access both to information they seek and to information they encounter as a side effect. The effects of the latter can be problematical in that they may incorporate inaccurate assumptions and faulty analytical strategies, which affect their understanding and approach to classroom material

Purpose of Educational Technology

As stated above, the purpose of educational technology is to support the goals and processes of teaching and learning, i.e., the pedagogy. Thus, to assess the value of any given technology one must ask: does it engage the student, does it help the student in understanding the core skills and concepts of modeling, abstraction and relationship creation, and in critical thinking? Being able to hack out a solution to a programming assignment is not sufficient. It is commonplace now to use visualization tools for courses dealing with computational processes and data, such as MatLab and Mathematica. But how can we measure their effectiveness?

Another issue is how to engage students on their turf, leveraging their experiences. Certainly our field is more attuned to providing examples and problems from the modern world our students are helping to shape, e.g., focusing on various problems relating to using and improving the Internet. In her recent talk at Brown danah boyd (CS '00) showed two simple ways of using the Internet as educational technology: 1) Wikipedia discussion pages - making meaning as very detailed discussions evolve with people who have names, not just anonymous comments - and 2) an example of an instructor-student dialog on the course blog discussing why learning a particular topic is important. While these examples address issues of incorporating the Internet into formal education, it is far from sufficient.

There are two major skills that students need to learn to effectively deal with the challenges they will face post-graduation: (1) working with modeling and cognitive skills to gain a deep understanding of the meaning and use of different concepts in different contexts, and (2) working with other people with diverse points of view. How can educational technology help/support this?

A Brief, Brown-centric Excursion into Educational Technology History

Understanding abstractions and relationships among levels of abstraction is hard; modeling with abstractions and relationships is even harder. Humans in general do not naturally work with formal symbols and higher-level abstractions and need to ground their understanding in the concrete to comprehend what the abstraction is actually representing.

Since the purpose of educational technology is to support teaching and learning, the range of tools created for that purpose is very large. Looking back in time one sees tools like the abacus, books, slates, chalkboard, slide rule, index cards, Montessori aids (e.g., her timeline), Cuisinaire rods, and even ditto machines, film strips, and Legos.

We naturally focus on computer-based tools, including authoring support software. Our concern is not computer literacy but proficiency, the understanding of core concepts and cognitive skills and the ability to manifest that understanding in the creation of appropriate artifacts: designs, programs, theorems, etc. Our scope includes, but does not discuss, knowledge worker tools including such familiar general applications as word processing, spreadsheets (non-linear programming), simple databases like Filemaker, personal content assistants for visualizing, analyzing, and sharing, like Tinderbox, as well as more targeted simulation and math tools such as MatLab and Mathematica, and simulation-based games such as SimCity and Spore.

The types of educational software specifically designed for educational purposes vary from lightweight single-concept applets to "serious" simulation-based games to AI-based intelligent tutors to educational programming and authoring environments. The projects described below are just illustrative examples of educational technology use; many more projects existed and many others are currently underway.

Algorithm and Concept Animation and Simulation

Andy tells the story of how his college textbook on Fluid Mechanics only had interleaved text and equations, with not a single visualization, except on the cover, to help relate the math to the dynamics of fluid flow. This frustrating experience of appealing to only mathematical insight and experience stimulated his lifelong interest in interactive illustrations, which later led to use of algorithm and concept animation via Balsa and Exploratories.

While Ken Knowlton's 1966 movie[1] on list processing that used animation to visualize program behavior is arguably the first algorithm animation movie, Ron Baecker's 1981 movie "Sorting out Sorting" that compared nine different sorting[2] algorithms really kicked the field off.

The Electronic Classroom

Brown's Electronic Classroom, also known as the Apollo Lab after the original workstations used, was inaugurated in 1982 as a pioneering facility for using interactive materials during the normal lecture course. The Apollo lab was "a laboratory/lecture hall containing 60 high-performance scientific workstations (Apollos) with bitmap graphic displays, connected together on a high bandwidth resource-sharing local area network. Courses in introductory programming, algorithms and data structures, differential equations, and assembly language have been taught in the lab using the

software environment described in this paper as the principal medium of communication. Rather than explain a concept using a blackboard or a viewgraph projector, instructors in these courses use dynamic graphic presentations."[3] Our present day Sun Lab is the second incarnation of this idea, although regrettably it is used less as an interactive classroom these days and more as lab a for doing programming assignments.

BALSA[3]

In the early 1980s, Professor Robert Sedgewick and his Ph.D. student Marc Brown developed BALSA (Brown ALgorithm Simulator and Animator). They conceived of BALSA as a "laboratory for experimentation with dynamic real-time representations of algorithms", and divided the task of developing algorithm animations into four roles: user - who was viewed as reader of a dynamic book, scriptwriter - the person preparing material for users, algorithm designer - a domain-expert programmer, and animator - programmer who designs and implements the animations using the BALSA facilities. BALSA provided the student interactive capabilities in the choice and sizing of views, of data to put into the algorithms, the speed with which to view the animations, and even the ability to run the programs in reverse.

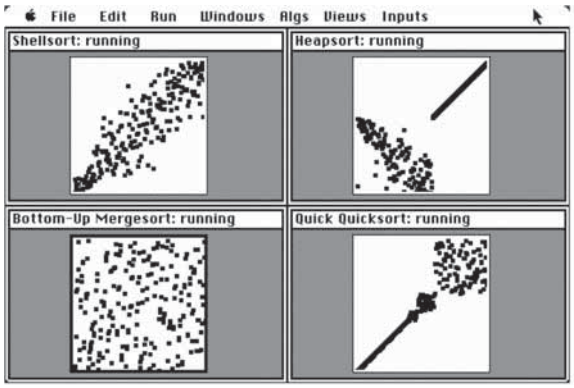


Fig 2: MacBalsa animation showing four different sort algorithms.

Subsequently Marc Brown developed MacBalsa as part of his Ph.D. thesis, Algorithm Animation, which recieved the ACM Distinguished Dissertation award. He continued work through the mid 90's at DEC's Systems Research Center (now HP Labs) focusing on the use of interactive 3D visualization, color, and sound for algorithm animations.

TANGO[4]

In the mid 1980s, as a follow-on to BALSA, Steve Reiss' Ph.D. student John Stasko created Tango, a framework for algorithm animation that provided a clear mapping between the algorithm and the graphical depiction of the state transitions. This framework embodied a formal model, the Path-Transition Paradigm, that helped designers and programmers to articulate the process of algorithm animation.

Stasko later went to Georgia Tech, where he continued research and teaching with algorithm animations, and created a frontend, Samba, an interactive animation interpreter. Samba was a simple animation scripting language that allowed students to create their own 2D animations of such algorithms as sorting and graph computations as two-week homework assignments.

MOCHA[5, 6]

In the early 1990s the Web provided a distributed communications infrastructure that supported hypermedia accessible via Java. This environment allowed secure Web-based deployment, and Roberto Tamassia and his group, in collaboration with Isabel Cruz, extended the possibilities of algorithm animation to the Web through the Mocha model. Mocha, which had a model-view-controller architecture, permitted multimedia algorithm animations to be embedded in a narrative hypermedia structure on the Web.

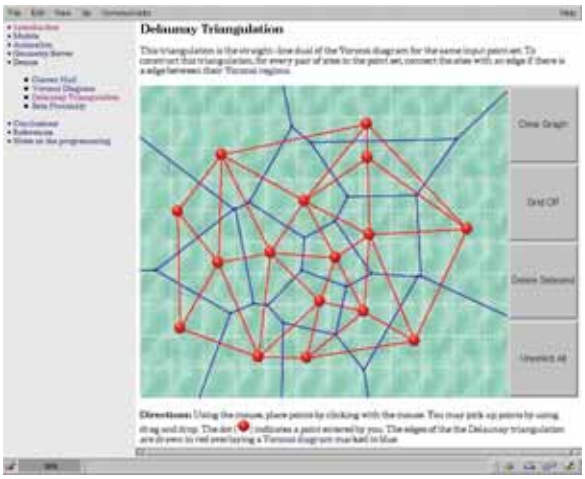


Fig 3: Mocha animation showing the Delaunay triangulation algorithm.

JDSL Visualizer[7]

In collaboration with Michael Goodrich at Johns Hopkins University, Tamassia's group leveraged the lessons learned about effective student use of algorithm animation from their extensive experience in teaching data structures and algorithms. They focused on extending their Java library, JDSL,[8] to provide animations for student-written implementations of fundamental data structures that follow the JDSL API. In addition, they developed testers for automatically comparing the functionality of user-written data structures with that of the reference JDSL implementation. Student evaluations of their experience using these tools in their programming assignments in CS-16 strongly supported the value of these tools for learning data structures and algorithms.

PILOT[9]

While the JDSL Visualizer focused on data structures based on lists and trees, PILOT addressed the visualization of algorithms for fundamental graph problems, such as minimum spanning tree and shortest path. PILOT displays and automatically draws a graph instance, and allows the user to step through the execution of the algorithm. It can be used both for learning an algorithm and to automatically grade the execution of an algorithm by the student. In learning mode, the user repeatedly indicates the next step of the algorithm (e.g., select the next edge to be added to the shortest-path tree) and is given immediate feedback from the system. In exam mode, the user executes all the steps of the algorithm and receives feedback at the end of the execution together with a grade. After an initial prototype of PILOT was created as a collaborative effort by the groups of Michael Goodrich and Roberto Tamassia, extensions and classroom experiments in CS-16 were developed by Brown undergraduates Ryan Baker and Susannah Raub.

Exploratories[10]

Andy's Exploratories project, an interactive Web-based algorithm and concept visualization project, started in 1996 with contributions by Jean LaLeuf fulltime for several years, multiple undergraduates, including Dan Gould and Jeff Beall, and professor John Hughes. It uses Java applets to aid students in exploring the concepts and algorithms for CS-123, introduction to computer graphics. An exploratory is a computer-based combination of an exploratorium and laboratory that provides a microworld for modeling objects, phenomena, and concepts. Particular exploratories are used in class and then students have the opportunity to play with them in lab to gain a deeper understanding of the concepts involved. Lab assignments include both algorithm descriptions and programming projects. One student, Alexandra Schultz, describes her experience in this semester's course thus: "Playing with the applets helped me understand the signal processing concepts much better; I think I could have done the programming assignment since I understood how to use filter kernels, but I definitely would have failed the preparatory algorithm assignment, and I wouldn't have known why my code worked."

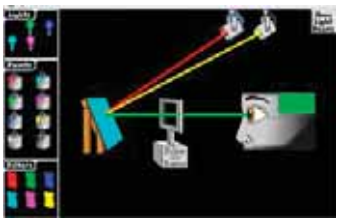


Fig 4: Exploratories screen showing the difference between mixing light and mixing paint*** showing additive and subtractive color mixing with reflective and transmissive material.

PhET (Physics Educational Technology) Online Simulations[11]

The University of Colorado's PhET website, founded by Nobelist Carl Wieman, contains interactive, domain and education research-based simulations that are available on the Web as interactive Flash animations and are downloadable as Java JAR files. The Java applets are embedded in context-providing web pages that contain links to explanatory teaching tips, related simulations, and teacher activity information. The website contains research publications and guides for teachers as to how to incorporate the simulations into their teaching. The project is a well-funded, multi-year effort led by a multi-disciplinary team.

"Is it a hit or is it a miss?"

Looking back at all these efforts, one can say that the glass is half full – there are certainly successful aspects to the algorithm animation projects done at Brown, but they aren't as integral a part of the curriculum as we early enthusiasts and evangelists had hoped. It takes a huge amount of effort to create and maintain any educational software, and

additional effort to integrate it into the pedagogy and assignment structure for a course. Also students have to see significant value in spending the extra time playing with the software when they are typically time-pressured to get their assignments done, and they may be satisfied with just an in-class demonstration. Algorithm animation, like so many things, is much more effective when done actively by students rather than observed passively, and having students interact with them is somewhere in between these two modes in terms of utility.

CAI (Computer-Assisted Instruction) and Cognitive Tutors

CAI originated as a "programmed instruction" strategy for breaking information into single-concept chunks that lent themselves to questions that tested student understanding of each chunk. Mastery of early material was a prerequisite to continuing with the presentation of topics, and the structure permitted simple test-based branching and looping back for remediation to ensure mastery. The simplistic and mechanical aspects of CAI could degrade to what is known as 'drill and kill,' but its fundamental strategy laid the groundwork for the much deeper modeling of ICAI, which included artificial intelligence and cognitive science research understanding.

PLATO (Programmed Logic for Automated Teaching Operations)[12]

In June 2010 the Computer History Museum hosted the PLATO@50 Conference, celebrating the 50th anniversary of "the computer system that many credit with presaging the networked world of social media and interactive education." As part of their collaboration in the educational and CAI courseware development processes, the PLATO community, under PI Don Bitzer's leadership, developed many currently familiar tools such as forums, message boards, online testing, e-mail, chat rooms, instant messaging, remote screen sharing, and multi-player games as well as creating the first plasma display panel. Brian Dear, the PLATO@50 conference organizer who was a programmer and courseware designer during 1979–1984 comments: "here was a computer that was all about connecting people, whereas micros were lonely islands..."

PAT (PUMP Algebra Tutor)[13]

PAT was developed in the early 1980s as an approach to teaching algebra in urban environments - hence the name PUMP (Pittsburgh Urban Math Project). CMU's John Anderson's work on the ACT* theory of learning and problem solving provided the foundation for what became a successful cognitive tutoring system. A cognitive tutor undertakes deep modeling of a student's mental models and provides interactive individualized instruction based on the student's responses. Studies over the years have validated the results with students; thus, PAT is a successful example of one approach to providing a "Teacher for every Learner", the CRA Grand Challenges project led by Andy. It is an ITS (Intelligent Tutoring System), which embodies the principle that in order for students to attain both deep understanding and skill facility they must model the process and concepts they are trying to learn through working with multiple representations of the problem.

Wayang Outpost[14]

Bev Woolf (UMass Amherst)'s Wayang Outpost (2009…) is a multiplatform online ITS for middle school and high school level mathematics that uses multimedia and animated adventures combined with affective intelligent agents that respond to how the students are using the system. The system employs machine learning techniques and individualized strategies that use affective intelligent 'companions'.

Hypertext/hypermedia Systems

Memex[15]

In the August 1945 issue of *The Atlantic Monthly,* MIT's Vannevar Bush wrote a visionary article about a system, which he called Memex, that could provide persistent associative trails through a library of documents stored on microfilm. He thus provided a model for an integrated library of documents that supported non-linear webs through the domain. This system for non-linear collecting, reading, annotating, and trail blazing was the inspiration for later hypertext/hypermedia systems.

HES (Hypertext Editing System)[16]

HES (1967–1969), was the earliest hypertext research project done on commercial equipment, Brown's mainframe computer (the IBM /360 mod 50 with 512K (!) memory and a 2250 vector graphics display). It was implemented by team of undergraduates including Steve Carmody (currently a senior information architect at CIS). In his keynote given at the first ACM Hypertext conference (HT '87), Andy commented on the origin and goals of HES:

> "I ran into Ted Nelson completely by accident at the 1967 Spring Joint Computer Conference, and gossiped with him about what we had both been doing since we left Swarthmore College. He told me about his ideas on hypertext, and one thing led to another and Ted started coming to Providence, using, as he is proud to say in Computer Lib, his own money. We started working on the Hypertext Editing System, which was essentially dual-purpose. One purpose was to produce printed documents nicely and efficiently, since at that time the technology on IBM/360 systems was batch cards for editing (mag card selectrics were not yet common). But the main purpose was to explore this hypertext concept."

HES's non-linear links among text fragments and string search facilities enabled authors to interactively create individual components that they could later select and structure into linear documents for production. In this, it laid the groundwork for future creative process support systems as well as production systems. Ted created one of the first webs, a hypertext on electroplating patents.

While it was a 'first pancake' proof-of-concept system, instructors used it to develop and produce course materials, and later when IBM, the main sponsor, sold it to NASA, it was used to produce documentation that went as microfilm with the Apollo missions.

FRESS[17]

In 1968, the era of punched cards and guarded fortress mainframes, Andy went to the FJCC (Fall Joint Computer Conference) and saw what was later known as "The Mother of All Demos": Doug Engelbart's demonstration of NLS (oNLine System). The Stanford University Engelbart archives site describes it thus:

> "On December 9, 1968, Douglas C. Engelbart and the group of 17 researchers working with him in the Augmentation Research Center at Stanford Research Institute in Menlo Park, CA, presented a 90-minute live public demonstration of the online system, NLS, they had been working on since 1962. The public presentation was a session of the Fall Joint Computer Conference held at the Convention Center in San Francisco, and it was attended by about 1,000 computer professionals. This was the public debut of the computer mouse. But the mouse was only one of many innovations demonstrated that day, including hypertext, object addressing and dynamic file linking, as well as shared-screen collaboration involving two persons at different sites communicating over a network with audio and video interface.[18]"

Andy was both stunned and inspired, and on his return to Brown began work with his team of undergraduates and several Masters students on what became FRESS. FRESS was a multi-user, platform-independent program that, like HES, supported both the text and hypertext authoring process and the production of documents. FRESS had a simple windowing system, configurable views, keyword retrieval, inter-and intrafile linking, basic editing and formatting, including what was probably the first undo command in a text editor, line art, and both network and hierarchical information structures.

In 1976 undergraduate students participated in an NEH-sponsored poetry class taught by English Professor Bob Scholes, founder of MCM, in which all of the work was done online, including instructor annotations and assignment criticisms, source materials (a database of more than 700 poems and professional criticism), student assignments, and student commentary. It was both an online scholarly research and educational community.

Intermedia[19]

Intermedia (1985–1992) was a scholarly and education research tool developed by IRIS (Institute for Research in Information and Scholarship) under CS graduate Norm Meyrowitz's direction. IRIS, initially headed by Bill Shipp and later by Norm, was a spinoff of the hypermedia work in Andy's Graphics Group. Intermedia's information architecture was database-backed and was notable for the separation of hypertext structures and data. This feature

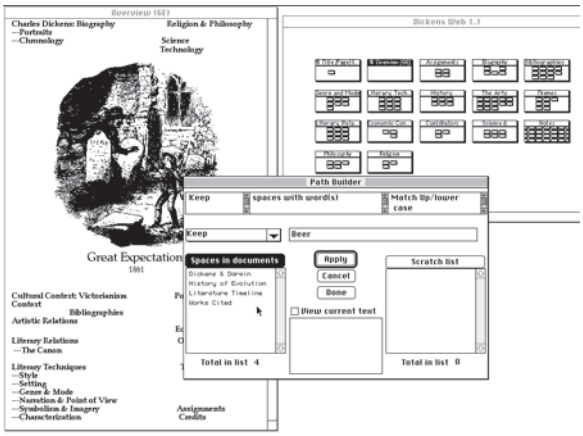


Fig 5: Intermedia screen showing components of the Dickens web.

enabled the support of multiple semantic webs over the same corpus of multimedia materials, multiple users with multiple levels of access rights, and author-reader transparency.

Intermedia was a fully object-oriented system that supported a suite of editors for an arbitrary collection of media types including, among others, text, graphics, timeline, and video, within a hypermedia framework that providing interlinking, annotation, and navigation facilities. An early paper by cultural anthropologist Bill Beeman et al. pointed out that "one of the key aims of Intermedia was to provide the ability to link different materials creating semantically meaningful relationships between different information and ideas. Using this software environment, 'authors' (who may be students or instructors depending on the guidelines determined by the instructor) could link and annotate materials to create 'semantic webs' or trails of meaning."[19]

Unlike today's browser-based implementations of the WWW, all Brown hypermedia systems — HES, FRESS, and Intermedia — shared the characteristic that there was no distinction between authors and readers, and that the focus was as much on process as on product.

Robots and Simulation games

Logo turtles - physical robots and virtual turtles[20]

Seymour Papert and Wally Feurzeig created Logo and its turtle in 1967. The original Logo turtles were physical robots that children moved around a room using Logo commands such as "forward 50." The educational intention was to combine movement in physical space with instructions in computational space, facilitating the mental mapping between symbolic operations and physical geometry operations as a way of having children learn mathematical and logical thinking. Later 'turtles' were screen pointers that drew lines on the screen in response to Logo commands. An interactive Java applet replicates the experiences: http://www.mathsnet.net/logo/turtlelogo/index.html[21]

Alan Kay's Dynabook vision[22]

In 1968 a meeting with Seymour Papert, where he learned about the pedagogical goals of Logo, inspired Alan Kay's DynaBook vision. Kay envisioned a personal portable computer usable by young children that would include the ability for them to author both text and graphics in a multi-model, collaborative network that included simulation-based software. Today's powerful laptops, tablets and pads are an implementation of this pioneering vision, but the full extent of the vision, especially the use of these devices in (simulation-based) learning, has scarcely been realized.

Lego Mindstorms[23]

Lego Mindstorms extends the original Logo pedagogical goals and practices, which instilled modeling and symbol/abstraction mapping skills by combining the manipulation of physical robots with computational math. Lego Mindstorms combines Lego blocks with other modular parts such as sensors and motors to build robots. These robots can then be plugged into a computer and their behavior controlled through programming. The activities vary depending on the age, and can include data gathering and analysis as well as the original geometric movements characteristics of the Logo turtles. Chad Jenkins' "Building Intelligent Robots" course is in this tradition.

Bootstrap curriculum[24]

The Bootstrap curriculum, designed by the PLT group of which Shriram Krishnamurthi is a founder, also instills modeling and symbol/abstraction mapping skills. Its strategy is to introduce middle school students to algebraic concepts and manipulation through the process of creating games, from initial design through final demonstration to parents and friends. The NY Times Magazine Education Technology issue[25] featured an article on the motivational and educational power of game creation for students of this age[26].

Immune Attack game[27]

While Logo, Lego Mindstorms, and the Bootstrap project use creating robots and games as a tools for teaching modeling and symbolic/abstraction mapping skills, the Immune Attack game illustrates the use of "serious games" to extend the role of concept simulation and animation in teaching complex domain processes. Inspired by the "The Fantastic Voyage" film[28], Immune Attack uses a nanobot (cell-sized submarine) to move within the body in response to immunological challenges. Like Palenque[29], Immune Attack has 'on-board advisors' who contribute scientifically valid information to players. Immune Attack development was sponsored by NSF and FAS (Federation of American Scientists), with Andy as PI and then FAS President Henry Kelly as co-PI.

Authoring Environments

Educational technology authoring environments are more context-sensitive than that of educational technology tools: sometimes authors are experts, possibly instructors, creating materials for students to use; sometimes authors are students, either graduate assistants or undergraduates, creating materials; sometimes authors and users comprise a shifting mixture, as in a class where the instructor and the students are creating artifacts together within a framework the instructor has created. Furthermore, general purpose knowledge tools, such as those described earlier, can also be considered as authoring tools.

AgentSheets[30]

AgentSheets, founded at UC Boulder in 1996 by Professor Alexander Repenning, is a visual programming environment targeted at middle and high school students that enables them to create Web-based simulation games. It has a drag-and-drop interface that permits the creation of simple games that can be uploaded to the Web, as well as sufficient power to develop Sims-like AI-based games. The system is part of the Scalable Game Design curriculum initiative whose intention is provide gender and culture-neutral motivation for secondary school level computer science.

Alice[31]

Alice, the legacy of the late Randy Pausch ('82), is an integrated IDE and educational programming language that uses a drag-and-drop environment to create computer animations using 3D models and scenes. Alice encourages storytelling, unlike most other programming languages, which are designed for computation, and is designed to appeal to specific subpopulations not normally exposed to computer programming, such as female students of middle school age.

DrRacket[32]

DrRacket (formerly called DrScheme) is the open-source graphical IDE for Racket, the current name for PLT Scheme. Its original, and continuing, primary intent is pedagogical: to provide an environment, usually Scheme-based, for teaching programming that is easy to use and that grows in power as skill and understanding develops. DrRacket has evolved into a full language research environment with multi-institutional support, along with continuing to serve its pedagogical functions; the Bootstrap curriculum described earlier is founded on DrRacket.

Greenfoot[33]

Greenfoot, developed at the University of Kent at Canterbury (UK) and Deakin University, Melbourne (Australia), with support by Sun Microsystems, is an educationally-oriented Java IDE that supports the development of 2D simulations and interactive games by high school and college-level students. It combines easy access to such facilities as graphics animations, sound, and interaction, while also helping to build an object-oriented mindset through exposure to the abstractions and conceptual structures of object-oriented programming. Guided visualizations and interactions provide constructivist-based learning support.

Kodu[34]

Microsoft Research's Kodu is a game development language that builds on the conceptual legacy of Logo, the Squeak version of Kay's DynaBook vision, AgentSheets, and Alice. It provides an icon-based interface for designing, assembling, and playing game scenarios. The goal is to control character behavior by moving icons around rule-based pages, using game-based notions and strategies. The envi-

ronment supports the process from design through execution by specialized primitives and by expressing programs in physical terms.

Medulla[35]

The U. S. Department of Energy's National Training and Education Resource (NTER) is an open-source framework in development that is designed to coordinate and support the use of virtual worlds and immersive environments for learning, teaching, and research. The design is based on the premise of continual improvement, rather than the more typical educational technology result of design, limited test, triumphant paper reporting on the results, code rot, decay, death, and "on to the next project". The goal is to facilitate interoperable virtual world development using off-the-shelf, open source and customized tools by creating the sort of persistent framework provided by Web browsers. The Federation of American Scientists (FAS) provided the research prototype and proof of concept, called Medulla.

Samba[36]

Samba is a scripting frontend to the Polka algorithm animation system. Polka is John Stasko's Georgia Tech follow-on to his Brown Ph.D. project, Tango. Samba provides upper-level college students with the ability to create algorithm animations for programs written in their choice of language. Students are given algorithm animation homework assignments that typically run 2-3 weeks. They annotate their programs with print statements containing Samba commands, thereby driving a visualization of the program.

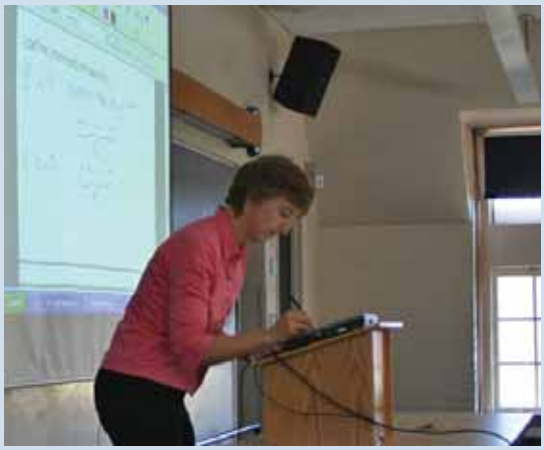
Scratch[37]

Scratch, a project of Mitch Resnick's Lifelong Kindergarten Group of the MIT Media Lab, is a programming language and scripting environment for children that focuses on being intuitive and easy-to-use, and that supports object reuse and sharing across the Internet. To create programs, users drag blocks, which contain code, from a palette on the screen onto the scripting area. Programs can be shared with others by uploading to the Scratch website where other users can download them for use as-is, for editing, and for remixing components with other components.

The Tablet PC may seem like a trivial example of technology use in education. However, as Claire Mathieu and her class have found, simple tools can make a profound difference. Her class likes the fact that the tablet annotations preserve the dynamics of classroom interaction for later reflection and study. Claire, a theoretician who prefers the physical action of working with a pencil to the more remote action of using a computer, has become a convert and likes the way Tablet use models her own preferred mode, as well as liking the ability to preserve 'whiteboard' use for later study.

She comments: "I need the spontaneity of text created on the board during an interactive process, where part of what I teach and how I teach it depend on the students' reactions on the spot. That is why I have always resisted the trend towards lecturing from slides. But now I can have the best of both worlds! My only frustration occurs on the few times when I accidentally disconnect the tablet from the projector; at those moments my students learn a bit more of my French than I would have wanted them to!"

Andy has also found the Tablet PC valuable for its ability to preserve in-class comments for later study so that the PPT slides plus audio captured from the lecture uploaded after class combine both presentation and interaction value.



Where are We Now and Where Do We Need to Go?

Observations

In general, we define a successful tool as one that is indispensable. Certainly Web browsers and productivity tools, such as word processing, and perhaps simple databases, spreadsheets, and, for Andy at least, PowerPoint, fall into this category. There isn't, however, any software specifically designed for educational use about which one can say that it is indispensible. What has tended to happen is two things:

- general purpose productivity and communication tools are used for educational purposes.
- demonstrable successes exist at very different points of the spectrum – e.g., at one end, lightweight (single concept) applets, instructional videos, and other point projects and at the other end, all the genres of heavyweight Intelligent Tutors. But here is and can be no universal tool for authoring and delivering all genres of educational software. (We don't concern ourselves with course management software here).

Stepping back and looking at five decades of educational technology and technology use for educational purposes, we see the following patterns for those areas that have been successful:

- *Small and simple things that require modest investment and that can be used in an ad hoc, remix way.* There is no formal integration involved and the end result depends on the need of the moment. Examples range from the Kahn Academy 15-minute minimal-production-value videos to simple applets (e.g., those used in the graphics course) to remixed material copy-pasted from the Web. It is useful to embed such point-products into an integrated hypermedia web, but that takes some up-front instructional and information-architecture design.
- *Integrated collections that may or may not have been constructed according to an instructional design.* These include reusable components, such as the Java libraries JDSL (Java Data Structures Library) and net.datastructures,[38] which Roberto Tamassia used for having students program to an API and derive animations of their own algorithms.
- *Higher level integrated systems that are designed for instructional purposes.* These include simulation-based serious games such as Immune Attack, simulation-based physical science teaching applets such as the PhET modules, the virtual world construction framework Medula, and AI-research-based cognitive tutors, such as PAT as well as the affect-based cognitive tutor research projects at MIT and UMass Amherst.
- *Learning by doing, i.e., constructivist learning.* This has a tradition going back to Alan Kay's Dynabook vision and the successful NEH Poetry class taught using our FRESS system. Current approaches are quite varied, as one might expect, and include middle school children creating their own games (Bootstrap curriculum), Maker Faire tinkering, and undergraduates creating animations as part of CS course assignments. A lovely example was shown by Shree Nayar during his recent talk at Brown via his bigShot build-it-yourself digital camera (http://www.bigshotcamera.org/sections/home/team.html), whose electrical, electromechanical, and optical workings are used by young students to learn the fundamental concepts in these domains as they assemble to illustrate fundamental concepts in these domains by young students as they assemble the camera.
- *Increasing emphasis on informal learning, unstructured collaboration using Internet tools and materials, and crowd sourcing.*

Open Problems/Grand Challenges

Looking ahead, we can briefly describe some examples of significant open research problems that address both current and anticipated unmet needs. None of these are ready to be well-defined research proposals, but are rather in the nature of quests we are striving to articulate.

Development environment for integrated systems

We need a better development environment for creating multi-disciplinary simulations with multiple levels of detail and multiple points of view. Andy's Clip Model idea[39] articulates one approach: Clip models are simulation- or rule-based models that are designed to be composed into increasingly

higher-level subsystems. The intent is to support a seamless environment in which multiple levels of detail and different points-of-view can be present at the same time, to address the needs of various classes/ages/experience levels of learners and learning situations. A powerful driving example would be a simulation-based "digital human" to be used for teaching/learning biology at a variety of levels. It would contain a huge number of models for component systems at varying levels of fidelity and understandability, and would allow models at different levels to interoperate to simulate higher-level subsystems. For example, a particular subsystem could include an overview-level model of the circulatory system anatomy and physiology while at the same time providing details of the white blood cells' role in immune system defense, e.g., through access to a simulation game such as Immune Attack.

Framework for collaborative authoring

We need a framework for small team (read "individual courses") collaboration using the Internet. As with the early hypermedia systems at Brown, individuals can assume roles as readers, authors, and commentators for all media, and the focus is both on process and on product. The product typically would be a composite of multiple media accessible through the Internet, and reflective of the work and goals of the small team/class.

The framework needs to be able to work with the understanding and manipulation of information structures. It needs to support both a priori designed information structures and emergent structures, with links that are first class citizens with persistent semantics. To provide these capabilities, the framework needs lightweight structure-authoring tools, in effect, a sketchpad for information structures that doesn't require information architect expertise.

Google Wave, unfortunately decommissioned, provided an recent example of one possible direction.

Teacher for every learner support system[40]

How can you provide the help someone needs, on a JIT basis, through some combination of computer tools, guidance to Internet resources, synchronous and asynchronous access to peer-to-peer/community tutoring and to experts? We need more than forums and wikis. When you are trying to articulate a difficulty, whether you are out in cyberspace working alone, or in a class where others think differently than you do, you need to dialog with a human being. There needs to be the ability to discuss the issues, in real time with a real person. Ideally such discussions would take place in a visual environment with shared presence at a smart whiteboard.

The support system could be a suite of tools, or a guided tour to a group of tools. Intelligent tutors can help, but are not sufficient.

Conclusion

It is hard not to be a cynic in the Oscar Wilde sense, i.e., a frustrated idealist, about the state of educational software after so many decades of research, experimentation, and deployment. The goal of providing the kind of huge acceleration and improvement that we've seen in other areas e.g., e-commerce, CAD-CAM, of obtaining access via the Web to information (all too much of it incorrect and of uncertain value), eludes us still in the area of educational software. But, the promise and potential are very much still there and the hope is that research will continue under enlightened sponsorship that recognizes the need for patient investment in the building and sustaining of multidisciplinary teams required to build ambitious tools and environments. If routinely 10–25 M$ is spent to fund a multi-disciplinary team to build an interesting game, why can't the money be found to fund a dozen such projects in innovative educational technology? In 2001 Andy and two colleagues, Henry Kelly, formerly the Director of the Federation of American Scientists, and Randy Hinrichs, formerly at Microsoft Research, founded the Learning Federation and obtained funding from NSF and industry to run workshops defining research roadmaps in learning science and technologies. These roadmaps unfortunately have never been funded, but are even more relevant today than they were in 2003. They are still accessible via the FAS at http://www.fas.org/programs/ltp/policy_and_publications/roadmaps/index.html

We continue to hope...

Computationally-Oriented Foundations

1. Introductory Courses - addressing a broad range of student interests

Address student interests while at the same time ensuring that these courses address a significant subset of the fundamental range of concepts and skills that comprise computational thinking.

Use these courses to instill a set of cognitive skills such as learning how to create, validate, and establish relationships among abstractions from data and information on hand, a key skill in effective modeling, simulation, and validation. This skill in working with abstractions, in turn, undergirds both the scientific method and computational thinking, and should be a part of every computationally-oriented course. The differences among such courses help to reinforce the underlying skills as students meet the same concepts in different contexts.

Other examples of cognitive skills include: working with the tradeoffs involved with different representations; moving, where appropriate, from a declarative understanding of a problem to an imperative understanding of that problem; reducing computationally intractable problems to related tractable problems; and building, simulating, and validating computational models that shed light on important questions.

Specific recommendations

- **Introduce students to computational approaches** through foundation courses across the spectrum of student interests, instilling a set of cognitive skills such as those described in the Introduction to the white paper, "… learning how to create, validate, and establish relationships among abstractions from data and information on hand, a key skill in effective modeling, simulation, and validation…. working with the tradeoffs involved with different representations; moving, where appropriate, from a declarative understanding of a problem to an imperative understanding of that problem; reducing computationally intractable problems to related tractable problems; and building, simulating, and validating computational models[1] that shed light on important questions."
- **Emphasize the creation of appropriate and useable sets of representations and relationships** among different levels; a deep understanding of how to represent information is one of the most difficult cognitive skills students need to learn. The practice of presenting accessible but important research papers as part of introductory courses not only introduces students to actual research work but also starts to build an understanding of how to compare different representations, analyze unstated assumptions, and build a common representation structure across a set of related projects.
- **Establish collaborative efforts** involving a computer science department, which could assume primary responsibility for the courses, and the departments whose prospective students are expected to take one of the courses. Additionally, the computer science department should help other departments in developing follow-on courses that take advantage of computational thinking taught in the first course.
- **Begin to shape a collaborative student culture** that will mature into effective professional teamwork skills, as described in Recommendations Two through Six.
- **Encourage students to begin building a digital portfolio**, including journal entries, possibly online [dealing appropriately with privacy issues], that carries through the core ideas and can be added to and be available for research ideas, building mastery, understanding one's own perspective, use in applying to graduate school. See Recommendation Six for a more complete description of this idea.

Refactoring Computer Science Curricula

2. Core/Foundation for All Computer Science Graduates - lean core with focus on enduring concepts, techniques, and skills

A relatively lean core emphasizes foundational concepts and skills while allowing students more time to explore areas in depth, both by taking courses and by engaging in undergraduate research. Additionally, a lean core makes it easier for students with multidisciplinary interests to pursue a joint major *[See Recommendation 4 - Specialization: Integrated Joint Majors]* while still sharing a common experience with computer science majors.

1 When we use the terms "model" and "modeling" we mean symbolic computational models, not numeric models, which are sets of differential equations.

Specific recommendations

Lean core with focus on the minimum essential cognitive skills, concepts, and techniques.

- Having a relatively lean core emphasizes **foundational cognitive skills and concepts** while allowing students more time to explore areas in depth, both by taking courses and by engaging in undergraduate research.
- The deep issues of **mastery and skills** faced by the core have strong connections to the issues discussed in *Recommendation 5 - Design under Constraints and the Gaining of Mastery.*
- A lean core makes it easier for students with **multidisciplinary interests** to pursue a joint major *[See Recommendation 4 - Specialization: Integrated Joint Majors],* while still sharing a common experience with computer science majors.
- The explicit identification of the lean core components makes it easier for a wide range of institutions to **identify resources**, establish a strong basic computer science foundation, and help their graduates pursue computationally-oriented research careers.

3. ***Specialization: Tracks, Threads, and Vectors - flexible approaches to gaining understanding and skills***

Define sets of meaningful specializations to permit students to pursue their interests in a context that guides their development while providing strong motivation. Ensure that these 'tracks' are specialized enough that a course sequence can lead to a student attaining some reasonable depth in the area but broad enough that someone in a company or graduate school will be able to fit it into their institutional context.

Specific recommendations

Specialized computing, through domain-centered tracks:

- We encourage schools to develop a **broad series of specializations**. The specifications reflected in the undergraduate course offerings of a given department will, of necessity, be based on department faculty interests and capabilities and the availability of courses in other relevant disciplines.
- The concepts that should guide the specializations include considerations of the ways in which the components of a particular sequence **build the skills and mastery** needed post-graduation, and how graduates will be viewed by graduate schools and potential employers.
- There is also a question of **scale –** departments with small faculties and student enrollments are clearly not able to offer as many tracks as larger departments. Concern with "break-even" course sizes are a necessary pragmatic. So, the number of tracks, and their depth, will vary quite a bit; small, resource-limited departments may not be able to do this at all, although as described in the White Paper, above Harvey Mudd, with an enrollment of 700, provides a counter-example of the creative use of resources to create specialized approaches.

4. ***Specialization: Integrated Joint Majors - deep collaboration among disciplines***

Coherent, integrated multidisciplinary, inter-departmental joint majors provide a balanced approach that addresses the differences in intellectual culture, concepts, and strategies between different fields by establishing the common ground between them. Use these integrated joint majors to provide a creative synthesis beyond that which can be provided by a computer science department alone, one that blends the cultures and mindsets of multiple departments and synergistically establishes new techniques for problem solving.

Specific recommendations

Deep collaboration among disciplines, exemplified by integrated joint majors:

- Coherent, integrated multidisciplinary, inter-departmental joint majors provide a **balanced approach** that addresses the differences in culture, concepts, and strategies between different fields by establishing the common ground between them.
- To undertake the considerable resource cost of joint major development, we recommend encouraging—indeed, incentivizing—the additional faculty effort required to design and implement new integrated courses and curricula. **Incentives** could include summer salary, release time, the designation of a dedicated 'curriculum czar', and the recruitment of students as research and teaching assistants to help with design and implementation.
- **Initial exploration** of collaborative possibilities can include multi-disciplinary curriculum committees, individual experimental courses, support for multi-disciplinary GISPs (Group Independent Study Projects), plus Internet-based collaboration using existing tools such as wikis.

Establishing Mastery across the Curricula

5. Design Under Constraints and the Gaining of Mastery - deepening the skill set

Provide students the ability to attain mastery by gaining experience in learning new technologies and techniques, building and analyzing artifacts, and learning to understand design as an iterative process that involves evaluating tradeoffs, analyzing system performance, and testing at each step. Create design and development experiences that tap into the actual interests of the students within a structure that both rewards effort and requires debugging/dealing with the uncertainties and approximations of real-world non-determinacy.

Specific recommendations

- Provide **integrative design experiences** earlier in the curriculum, including the first course, and throughout the curriculum, building on the student's increasing skills.
- Incorporate **skill descriptions** in addition to course topics in all courses. Articulate how the elements of mastery, wizardry and purpose form part of the course outcomes.
- Integrate success stories of **project integration across the curriculum** into individual courses, to better leverage instructor time and resources. Make course developments widely available on the web, so that others may use, adapt, and extend them for their own courses and for the community at large.

6. Attracting, Selecting, and Preparing Students for Research Careers - developing computationally-oriented researchers

Skillfully introduce research problems and their intellectual excitement in all courses, thus helping to entice potential research students by disabusing them of the notion that our field has become routinized. Successful courses that attract and excite students present new concepts within the context of the ongoing research of the R&D community.

Combine explicit research skill training with an apprenticeship approach to acculturate future researchers to their communities of practice. Provide systematic guidance in the practices of computationally-oriented research from freshman year through graduation combined with the support provided by close relationships with graduate students, research groups, and professors.

Specific recommendations

Attract students to research careers through the introduction of research approaches and skills across the full undergraduate spectrum:

- Combine explicit **research skill training** with an **apprenticeship approach** to acculturate future researchers to their community of practice. This means systematic guidance in the practices of computationally-oriented research from freshman year through graduation combined with the support provided by close relationships with graduate students, research groups, and professors.
- Attract and prepare the best qualified students by exposing them to and engaging them in **exciting computing research –** the earlier, the better.
- **Engage undergraduates in research** via various means: focused study groups that include graduate students, student-initiated GISPs (Group-Independent Study Programs), seminars that undergraduates are encouraged to attend, undergraduate research assistantships, summer programs and internships, university-sponsored internships, special scholarship programs, such the University of Utah Access Program and Engineering Scholars Program.
- Facilitate the creation and use of a **persistent digital portfolio** from the beginning of their first introductory classes, continuing through all of their ongoing courses, to provide both an idea resource base and a record for future papers and graduate school work
- Emulate the model for project-oriented courses provided by the **domain-specific introduction to graduate research methods** (Brown's CS237 - Interdisciplinary Scientific Visualization), described in the "How should we prepare them for this goal?" section in the White Paper.
- Provide **small classes** including a large percentage of qualified students to enable significant teacher interaction for the students.

1. Ken Knowlton. L6: Bell Telephone Laboratories Low-Level Linked List Language. 16-minute black-and-white film. 1966.
2. Ron Baecker (with assistance of Dave Sherman). Sorting out Sorting. 30 minute color film (distributed by Morgan Kaufmann Pub.), 1981.
3. Marc H. Brown and Robert Sedgewick. 1984. A system for algorithm animation. SIGGRAPH Comput. Graph. 18, 3 (January 1984), 177–186.
4. John T. Stasko. "Tango: a framework and system for algorithm animation," IEEE Computer, 23:9, pp.27–39, September 1990.
5. James E. Baker, Isabel F. Cruz, Giuseppe Liotta, and Roberto Tamassia. 1999. Visualizing geometric algorithms over the Web. Computational Geometry: Theory and Applications 12(1–2): 125–152 (1999).
6. James E. Baker, Isabel F. Cruz, Giuseppe Liotta, and Roberto Tamassia. 1996. The Mocha algorithm animation system. In Proceedings of the Workshop on Advanced visual interfaces (AVI '96), Tiziana Catarci, Maria F. Costabile, Stefano Levialdi, and Giuseppe Santucci (Eds.). ACM, New York, NY, USA, 248–250.
7. Ryan S. Baker, Michael Boilen, Michael T. Goodrich, Roberto Tamassia, and B. Aaron Stibel. 1999. Testers and visualizers for teaching data structures. SIGCSE Bull. 31, 1 (March 1999), 261–265.
8. http://www.cs.brown.edu/cgc/jdsl
9. Stina S. Bridgeman, Michael T. Goodrich, Stephen G. Kobourov, Roberto Tamassia. "PILOT: an interactive tool for learning and grading" in Proceedings of the SIGCSE Technical Symposium on Computer Science Education, pp, 139–143, 2000.
10. Anne M. Spalter and Rosemary M. Simpson. "Integrating interactive computer-based learning experiences into established curricula: a case study" in Proceedings of AMC ITiCSE '00, pp. 116–119, 2000.
11. PhET website - Physics simulations - http://phet.colorado.edu/; http://phet.colorado.edu/en/about
12. Kirk L. Kroeker. Celebrating the Legacy of PLATO" in ACM CACM, 53:8, pp 19–20, August 2010.
13. PAT (PUMP - Pittsburgh Urban Mathematics Project - Algebra Tutor) - http://act.psy.cmu.edu/awpt/awpt-home.html; ctat.pact.cs.cmu.edu/index.php?id=timeline
14. Wayang Outpost - http://wayangoutpost.com/; Beverly P. Woolf, Ivon Arroyo, Kasia Muldner, Winslow Burleson, David Cooper, Robert Dolan, and Robert M. Christopherson. "The Effect of Motivational Learning Companions on Low-Achieving Students and Students with Learning Disabilities" in Proceedings of the International Conference on Intelligent Tutoring Systems, 2010, Pittsburgh.
15. Memex article "As We May Think": http://www.theatlantic.com/magazine/archive/1969/12/as-we-may-think/3881/
16. Steven Carmody, Walter Gross, Theodor H. Nelson, David Rice, and Andries van Dam. "A Hypertext Editing System for the /360" in Faiman and Nievergelt (eds.) Pertinent Concepts in Computer Graphics: Proceedings of the Second University of Illinois Conference on Computer Graphics, pp. 291–330, University of Illinois Press, 1969.
17. David G. Durand and Steven J. DeRose. 1993. FRESS hypertext system (abstract). In Proceedings of the fifth ACM conference on Hypertext (HYPERTEXT '93). ACM, New York, NY, USA, 240-; Erwin K. Welsch. "Hypertext, Hypermedia, And The Humanities." Library Trends 40 (Spring 1992): 614-46.
18. http://sloan.stanford.edu/MouseSite/1968Demo.html
19. Wiliam O. Beeman, Kenneth T. Anderson, Gail Bader, James Larkin, Anne P. McClard, Patrick McQuillan, and Mark Shields. 1987. Hypertext and pluralism: from lineal to non-lineal thinking. In Proceedings of the ACM conference on Hypertext (HYPERTEXT '87). ACM, New York, NY, USA, 67–88.
20. http://el.media.mit.edu/logo-foundation/logo/turtle.html; http://en.wikipedia.org/wiki/Logo_(programming_language)
21. http://www.mathsnet.net/logo/turtlelogo/index.html
22. Alan Kay. "A Personal Computer for Children of All Ages" - http://www.mprove.de/diplom/gui/Kay72a.pdf
23. Lego Mindstorms: http://www.lego.com/education/school/default.asp?locale=2057&pagename=ict_home&l2id=3_2
24. http://www.bootstrapworld.org/
25. http://www.nytimes.com/indexes/2010/09/19/magazine/index.html
26. http://www.nytimes.com/2010/09/19/magazine/19video-t.html?ref=magazine
27. Immune Attack: http://www.fas.org/immuneattack/
28. http://en.wikipedia.org/wiki/Fantastic_Voyage
29. Kathleen S. Wilson. "Palenque: an interactive multimedia digital video interactive prototype for children" in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Washington, D.C., United States, May 15–19, 1988). J. J. O'Hare, Ed. CHI '88. ACM, New York, NY, 275–279, 1988.
30. AgentSheets home page: http://www.agentsheets.com/
31. Alice home page: http://www.alice.org/
32. DrRacket programming environment: http://docs.racket-lang.org/drracket/index.html; Racket language: http://racket-lang.org/learning.html; PLT (Programming Language Team) - http://www.cs.rice.edu/CS/PLT/
33. http://www.greenfoot.org/about/whatis.html; http://en.wikipedia.org/wiki/Greenfoot
34. http://research.microsoft.com/en-us/projects/kodu/; http://fuse.microsoft.com/project/kodu.aspx
35. Michelle Roper Fox, Henry Kelly, and Sachin Patil. "Medulla: A Cyberinfrastructure-Enabled Framework for Research, Teaching, and Learning with Virtual Worlds" in Online Worlds: Convergence of the Real and the Virtual. Human-Computer Interaction Series, 2010, pp. 87–100.
36. Samba algorithm animation system: http://www.cc.gatech.edu/gvu/softviz/algoanim/samba.html; John T. Stasko. "Using student-built algorithm animations as learning aids" in ACM SIGCSE Bull. 29:11, pp. 25-29, March 1997.
37. Scratch home page: http://scratch.mit.edu/; http://en.wikipedia.org/wiki/Scratch_(programming_language)
38. http://net.datastructures.net/
39. Andy van Dam and Rosemary Simpson. "Next Generation Education Software: Issues and Possibilities" in Conduit, 14:2, Fall/Winter 2005, pp.1, 10–19.
40. Andries van Dam. "Grand Challenge 3. Provide a Teacher for Every Learner" in Grand Research Challenges in Information Systems. Anita Jones and William Wulf, editors, pp. 18–22, Computing Research Association, 2003.
41. CRA-E White Paper: http://www.cra.org/uploads/documents/resources/rissues/CRA-E-Researcher-Education.pdf

The Artemis Project

The Artemis Project, a free five-week program offered to rising ninth grade girls in the Providence area to introduce them to the field of computer science, completed its 15th year this summer. Usually run by four Brown undergraduate women, this summer's Artemis welcomed a fifth coordinator from Boston University, Katherine Zhao '12, who plans to expand the program to the Boston area. Along with Katherine, coordinators Michelle Micallef '12, Natalie Serrino '12, Miranda Steele '13 and Tashyana Thomson '12, led this year's group of 21 girls.

The Artemis program included lessons on a variety of computer science topics, with lab exercises to provide the girls with hands-on computer experience. Early on in the planning process, the coordinators decided that showing the girls a breadth of topics was more important than delving in depth in only a few to illustrate the wide applicability of computer science and increase the chance each of the girls would find an area she loved. The girls learned about binary numbers and logic operators, sorting algorithms, breadth-first and depth-first search, and computer hardware basics. The coordinators used interactive teaching methods to capture the students' attention and create a fun learning experience. For example, the coordinators created a board game called "Motherboard" which led the players to navigate through the hardware components of a computer that are used for a task the girls were familiar with – composing and saving a document. The classroom activities were important to show the math and science foundations behind computer programming.

In the computer labs, the girls learned how to use Photoshop to turn a photo into an advertisement and created their own websites using HTML/CSS, and both projects were as popular this year as in the past. The coordinators selected the Java language for the programming exercises. The girls were first introduced to Alice, a software tool developed at CMU to teach novice users object-oriented programming concepts by creating virtual stories. This simplified the teaching of the Java language constructs as the coordinators were able to refer back to the Alice counterparts to help the girls understand. The last lab exercise involved programming Finch robots, on loan to Artemis by CMU, using Java. The girls programmed their Finches to dance, culminating in a "dance-off" and implemented the game "Finch, Finch, Revolution" where the girls interacted with the robots according to arrows on the computer.

The Artemis coordinators invited speakers from Adobe, Google, and Microsoft to speak to the girls about computer-related jobs, the future of technology, and women in computer science. The girls also enjoyed the presentations from several Brown Computer Science faculty about their research, including Internet security, robotics, and computer graphics. Field trips are a tradition of the Artemis Project, and this year the girls went to the MIT museum in Cambridge, where they programmed LEGO Mindstorm robots and explored the museum's robotics exhibit, and discovered the history of computing in their tour of the Boston Museum of Science.

Artemis is looking for new coordinators for next summer's program. If you are interested or have any questions or suggestions you can email the current coordinators at artemis@cs.brown.edu.

IPP Symposium on Cloud Computing

by Rodrigo Fonseca

On May 13th we hosted the 42nd Industrial Partners Program Symposium, following a long and successful tradition of very interesting and engaging events that bring together our partners, students and faculty to discuss relevant, current topics in Computer Science.

Like many emerging trends, Cloud Computing is not without a lot of hype, a multitude of definitions, and controversy. For example, Larry Ellison, from Oracle Corporation, was quoted as saying that *"The interesting thing about Cloud Computing is that we've redefined Cloud Computing to include everything that we already do. I don't understand what we would do differently in the light of Cloud Computing other than change the wordings of some of our ads."*[1]

On the other hand, Bill Gates said that *"We're taking everything we do at the server level and saying we will have a service that mirrors that exactly. (…) it's getting us to think about data centers at a scale that never existed before... [to create] a mega-data center that Microsoft and only a few others will have."*[2]

Indeed, the vision of computing as a utility, much like water or electricity, has been around for a long time, and was one of the driving forces for the Multics operating system, which started in the 60s. It was not until recently, however, that several trends converged to make this a viable alternative: a large number of well-connected clients, very large, well-connected datacenters leveraging economies of scale, and robust distributed software systems that allow elastic scalability of applications. Application providers can reach large populations of clients with low initial investment, cloud providers can profit from the efficiency of multiplexing their vast computational resources among many tenants, and users benefit from a software delivery model that always provides the latest version of the software and stores most of the data in the cloud. As more users provide data to increasingly popular services, these services can then use the data to their advantage, finding trends and tuning their offerings even more.

There are many flavors and definitions of Cloud Computing, depending on the provider and on the goal. The National Institute of Standards and Technology defines it as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."[3] Cloud providers vary in the level of abstraction they present to their tenants. Common variations are Infrastructure as a Service, Platform as a Service, and Software as a Service. An example of the first is Amazon's offering of EC2, which presents to their client bare virtual machines. Google's AppEngine and Microsoft's Azure offerings are examples of the second type, presenting a programming environment with tools and services to allow application development, deployment, and management. The last variation is the provision of full applications hosted in the provider's cloud, as is the case with SalesForce.com. Clouds can also be public or private. The four examples above are of public cloud services, but companies like Yahoo!, Facebook and eBay have internal cloud services used by the different services they provide.[4]

We had an impressive set of speakers for this Spring's symposium, coming from VMWare, Facebook, Yahoo!, Google, Cloudera, and Microsoft. Taken together, these companies own a very significant fraction of the world's computers dedicated to Cloud Computing! In addition to these, we had two Ph.D. students representing relevant research being done at Brown. Other IPP member companies were represented in the 55-people audience, including GTECH and Edelman & Associates.

Rather than trying to get the participants to agree on a definition of Cloud Computing, they were asked to convey their own vision on the subject, and talk about some specific aspect that they thought was relevant and interesting, and that would lead to further discussion. The result was a series of talks that mixed vision and also specific aspects, mechanisms or infrastructure services, such as storage, analytics, management, and security.

After brief introductions by Professors Ugur Cetintemel and Rodrigo Fonseca, Orran Krieger, from VMWare, talked about how VMWare sees cloud computing. Virtualization, VMWare's chief expertise, is central to many cloud provision solutions. It is not the whole story, though, as the management of large collections of virtual computing, storage, and networking components on top of physical resources are very challenging. VMWare is leading the effort to produce an open cloud management API, vCloud, which will both make the task easier and more standardized, according to Orran. Cloud computing today is a segmented ecosystem, where it is very hard for clients of one cloud provider to migrate to another provider. Ideally, one would be able to move from private clouds to public clouds depending on scale and business needs, and a uniform cloud interface can help bring us in that direction.

Harry Li '02, from Facebook, presented two of Facebook's solutions to serve dynamic content to its half a billion users, highlighting how one has to balance efficiency, scalability, elegance, and pragmatism when deploying applications at this scale. The first system, Haystack, is a novel datastore optimized to serve Facebook's photos. Facebook recently became the largest photo-sharing site, with over 2.5 billion pictures uploaded every month. Haystack is a clean-slate design that is highly optimized, foregoing traditional POSIX file system semantics to incur at most one disk access per photo served, which allows the site to handle its load that reaches 500,000 image requests per second at times! In the second half of the talk he described how Facebook aggressively uses memcached to reduce the load on MySQL databases for its persistent storage, and how they cope with the consistency problems that arise with concurrent accesses from multiple datacenters. He mentioned how the second solution is not as elegant, but works well enough, and how both approaches contribute to allow Facebook to continue to scale.

Some of the Symposium participants. From left to right, Srikanth Kandula (MSR), Rodrigo Fonseca (Brown), Joseph Hellerstein (Google), Adam Silberstein (Yahoo! Labs), Andy Pavlo (Brown), Charalampos Papamanthou (Brown), and Harry Li (Facebook). Missing from the picture are Orran Krieger (VMWare), and Jeff Hammerbacher (Cloudera).

Adam Silberstein, from Yahoo! Research, also talked about storage systems at the cloud scale, but from the point of view of benchmarking. Given a multitude of offerings for scalable storage backends, occupying different points in the design space, but targeting horizontal scalability to large datacenters and collections of datacenters, how can we evaluate their benefits and pitfalls? Each system provides their own evaluations, but on different hardware setups and under unique workloads, making comparisons impossible. Adam presented the Yahoo! Cloud Serving Benchmark, an open-source tool released by Yahoo!, that facilitates the comparisons among cloud serving systems, and discussed experimental results comparing four systems: Facebook's Cassandra, Apache's HBase, Yahoo!'s PNUTS, and a static, sharded deployment of MySQL.

Joe Hellerstein, from Google, dived into the infrastructure to provide computation at large scale, specifically about how to use quantitative models to help guide the many decisions

involved in building these systems. These design choices include scheduling policies for compute clusters, caching and replication policies for storage, and approaches to integrating bandwidth management with application requirements for quality of service (QoS). They must be evaluated along several dimensions, such as throughput, latency, jitter, as well as the consumption of power, compute, storage, and network bandwidth. Joe shared some details of how they instrument production systems to obtain metrics that are then used to tune the systems, and to produce workload models useful for further exploration. His team at Google is making some of this data available for the research community, which is an exciting first step.

Right after lunch, Jeff Hammerbacher, from Cloudera, talked about the new needs for business analytics in the face of the unprecedented amounts of data generated by companies today, and how relational database management systems are just one of the pieces of the whole research cycle. As an example, he mentioned how Microsoft's SQL Server has evolved, in the 2008 R2 version, to include tools other than the RDBMS for event handling, integration, reporting, analysis, search, and OLAP. What does this have to do with the Cloud, you may ask… Jeff argued that Hadoop, the open source implementation of Google's MapReduce paradigm, with its set of related projects, is the base for a new powerful, open source, enterprise-grade analytics platform. Hadoop is designed to run on the same infrastructures that power cloud computing offerings, warehouse-scale datacenters with thousands of commodity machines working together.

Andy Pavlo, a Ph.D. student from our department, continued on the topic of where MapReduce fits relative to traditional database management systems. Some proponents of MapReduce claimed that the extreme scalability of MapReduce would relegate database technology to niche applications. Andy showed results from benchmarks comparing the two technologies, highlighting that they each have their merits, and that MR produces less than satisfactory results for some workloads for which databases are suitable. Rather than having to select a winner, what we see is cross-pollination between the two worlds, with more complex query languages added to MapReduce, and some of the processing and fault-tolerance characteristics of MapReduce being integrated with several database products.

Srikanth Kandula, from Microsoft Research, presented some recent research results on how to optimize the runtime of map-reduce jobs in a large cluster by identifying and selectively duplicating computation stages that are taking too long to complete. These straggler jobs can severely decrease the performance of whole clusters, even though they may have isolated causes. Their results showed an improvement of over three times in performance on a cluster of 12,000 nodes.

The last talk of the day touched on security, a very important and crosscutting issue in this space. Perhaps one of the greatest barriers to more widespread adoption of Cloud Computing is how to guarantee security – integrity and confidentiality – of the data and applications that companies and individuals delegate to third party cloud providers. How can you trust the provider? Charalampos Papamanthou, also a Ph.D. student from our department, described Dynamic Provable Data Possession, a method that can be used to check the integrity of dynamic data stored in the cloud. The main advantage of this scheme is that one can verify that the cloud provider is correctly storing a potentially very large collection of files without having to transfer all bits of the file.

In all, the IPP symposium achieved both breadth and depth, and generated many interesting questions and discussions during and after the talks. To see the program, abstracts of the talks, and some of the videos of the talks, you can check out the permanent home for this edition of the symposium, listed below, together with additional pointers for each of the talks. We look forward to future editions of the event, so stay tuned!

To learn more:

- 42nd IPP Symposium permanent home: http://www.cs.brown.edu/ipp/symposia/ipp42/
- VMWare's vCloud APIs: http://www.vmware.com/go/vcloudapi
- Facebook's Haystack is described in detail in the paper "Finding a Needle in Haystack: Facebook's Photo Storage", by Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, and Peter Vajgel, Facebook Inc, presented at OSDI, 2010.
- Yahoo!'s Cloud Serving Benchmark is described in detail in the paper "Benchmarking Cloud Serving Systems with YCSB", presented in the First ACM Symposium on Cloud Computing, Indianapolis, 2010. http://research.yahoo.com/Web_Information_Management/YCSB
- Joe Hellerstein's talk is complemented by the paper "Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters", Asit Mishra, Joseph L Hellerstein, Walfredo Cirne, Sigmetrics Performance Evaluation Review (2009). http://www.google.com/research/pubs/archive/35611.pdf
- A presentation by Jeff Hammerbacher on the industry track of SIGMOD 2010, slightly expands his talk at the symposium: http://www.slideshare.net/jhammerb/20100608sigmod
- Andy Pavlo gave a similar talk at MIT in January, "MapReduce and Parallel DBMSs: Together At Last". http://www.cs.brown.edu/~pavlo/presentations/mapreduce-nedb2010.pdf
- Srikant's talk draws from material in this OSDI 2010 paper: "Reining in the Outliers in Map-Reduce Clusters", by Ganesh Ananthanarayanan, Srikanth Kandula, Albert Greenberg, Ion Stoica, Yi Lu, and Bikas Saha

1. Finally, to learn more about Dynamic Provable Data Possession you can read the paper "Dynamic Provable Data Possession", by Chris Erway, Alptekin Küpçü, Charalampos Papamanthou and Roberto Tamassia, in Proceedings of the ACM Int. Conference on Computer and Communications Security (CCS), pages 213-222, Chicago IL, USA, 2009. http://dx.doi.org/10.1145/1653662.1653688

Endnotes

1. 'Oracle's Ellison nails cloud computing', CNET News, Sep 26th, 2008, http://news.cnet.com/8301-13953_3-10052188-80.html
2. Bill Gates Keynote: Microsoft Tech-Ed 2008 – Developers, June 3rd, 2008. http://www.microsoft.com/presspass/exec/billg/speeches/2008/06-03teched.mspx
3. 'NIST Definition of Cloud Computing v15', http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc
4. See for example 'Who Has the Most Web Servers', at http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/

John von Neumann Days and The Genome and the Computational Sciences: The Next Paradigms

May 3–7, 2010, Brown University

Symposium organized by the Center for Computational Molecular Biology, Department of Computer Science, and the Office of Brown University President Ruth J. Simmons

A Distinguished Lectures only symposium. All lectures followed by Sweatbox Session. All Distinguished lectures and their Sweatbox Session video are available at http://www.brown.edu/Research/CCMB/

The Brown University Center for Computational Molecular Biology Symposium was held May 3–7, 2010. The symposium was composed of two parts. The first part (May 3–4) was entitled John von Neumann Days and hosted a series of John von Neumann Distinguished Lecturers.

Brown University President Ruth J. Simmons introducing Marina von Neumann Whitman

Marina von Neumann and family

Brain Signaling Panel left to right: Sean Eddy, Stuart Geman, Elie Bienenstock, Kenneth Arrow, Sorin Istrail, and Leon Cooper

Daughter of John von Neumann

Marina von Neumann Whitman

Professor of Business Administration
And Public Policy,
University of Michigan

"John von Neumann: A Daughter's View"

May 3, 2010

von Neumann Distinguished Lecture on Mathematics

John Conway

John von Neumann Professor of Mathematics,
Princeton University

"von Neumann, Set Theory and Life"

May 3, 2010

von Neumann Distinguished Lecture on Economics

Richard Karp

University Professor
University of California, Berkeley

"Understanding Science Through the Lens of Computations"

May 3, 2010

von Neumann Distinguished Lecture on Physics

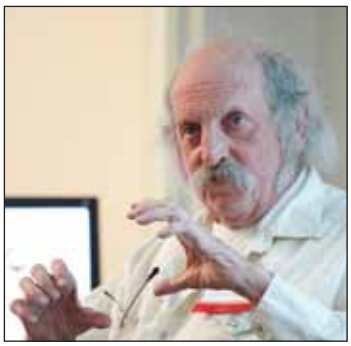
Leon Cooper

Nobel Laureate, Thomas J. Watson Professor of Science,
Director, Institute for Brain & Neural Systems,
Brown University

"von Neumann and the Problem of Measurement in the Quantum Theory"

May 4, 2010

von Neumann Distinguished Lecture on Biology



Eric H. Davidson

Chandler Professor of Cell Biology
California Institute of Technology

"Genomic Control System for Development: The Sea Urchin Embryo Gene Regulatory Network

May 5, 2010

von Neumann Distinguished Lecture on Economics

Kenneth Arrow

Nobel Laureate
Joan Kenney Professor of Economics
Professor of Operations Research Emeritus
Stanford University

"My Beliefs and Ours: Formation, Costs, Interactions, and Effects"

May 7, 2010

Distinguished Lecture on John von Neumann's Legacy

George Dyson

Historian of Science

Presented for Freeman Dyson

Professor Emeritus of Physics,
Institute of Advanced Study,
Princeton University

"A Walk Through Johnny von Neumann's Garden"

May 4, 2010

On the computer screen, Professor Freeman Dyson joined the panel session "Brain Signaling: Digital, Analog or Both?" on the last day of the symposium via videoconference. Professor Dyson was not able to deliver his distinguished lecture due to hospitalization the weekend before the sympsosium. His son, George Dyson, delivered the lecture for him. Professor Dyson's appearance via videoconference only a few days after his hospital release to contribute to the celebration of John von Neumann, a scientist hero to so many of us, was a triumph in conquering adversity.

The Genome and the Computational Sciences

May 5–7, 2010

A toast in celebration of the new CCMB based Ph.D. program in computational biology at Brown

Left to right: Stuart Geman, Roberto Serrano, Sorin Istrail, Kenneth Arrow, Chip Lawrence, Leon Cooper

Andrew G. Clark, Jacob Gould Schurman Professor of Population Genetics, Cornell University
"Consequences of Explosive Human Population Growth on Complex Genetic Disorders"

Ellen V. Rothenberg, Albert Billings Ruddock Professor of Biology, California Institute of Technology
"A Gene Regulatory System Guiding Cell Identity Choice: Stem Cells to T Cells"

Jonathan W. Yewdell, Chief, Biology Section, Laboratory of Viral Disease, NIAID, NIH
"It's Not Written in Stone: Stress Induced Alteration in the Genetic Code"

Richard Lewontin, Alexander Agassiz Research Professor, Museum of Comparative Zoology, Harvard University
"The Problem of Natural Selection for Genomic Variation"

Martha Bulyk, Associate Professor of Medicine & Pathology, Harvard Medical School
"Transcription Factor-DNA Interactions: Cis Regulatory Codes in Genomes"

David Shaw, Chief Scientist, D.E. Shaw Research, Senior Research Fellow, Center for Computational Biology and Bioinformatics, Columbia University
"Millisecond-Long Molecular Dynamics Simulations of Proteins on a Special Purpose Machine"

Martin Meier-Schellersheim, Team Leader, Computational Biology Group, NIAID, NIH
"The Challenges of Spatially Resolved Computational Modeling of Cellular Signaling Processes"

Sam Broder, Medical Officer, Celera and former Director of the National Cancer Institute, NIH
"The Development of Antiretrovial Therapy and Its Impact on the Global HIV-1/AIDS Pandemic: Personal Reflections on a Journey to Treat an 'Untreatable' Virus After 25 Years"

Sean Eddy, Group Leader, Janelia Farm, Howard Hughes Medical Institute
"Reading Genomes"

Symposium Organizers:

Leon Cooper, Thomas J. Watson Sr. Professor of Science, Director, Institute for Brain and Neural Systems

Stuart Geman, James Manning Professor of Applied Mathematics, Division of Applied Mathematics

Sorin Istrail, Julie Nguyen Brown Professor of Computational and Mathematical Sciences, Professor of Computer Science, Director, CCMB, Symposium Chair

Roberto Serrano, Harrison S. Kravis University Professor of Economics

The symposium administrative coordinator was **Lisa Manekofsky**.
Software engineering support was provided by **Derek Aguiar**.

Commencement

Daniel Grollman

Diana Huang and Lyla Fujiwara

Micha Elsner, Suman Karumuri & wife, David McClosky, Deepak Santhanam

Al Urim, James Stout, Spencer Brody

Hojoon Park, Marek Vondrak

Christina Salvatore, Matt Jacobs, Elizabeth Cheever, Andy van Dam, Andres Douglas Castroviejo, Jonathan Natkins

Department News and Awards

Michael Black Awarded Prize for Fundamental Contributions in Computer Vision

Michael Black was recently awarded the 2010 Koenderink Prize for Fundamental Contributions in Computer Vision. The Prize was given at the European Conference on Computer Vision (ECCV) in Hersonissos, Crete for work that has withstood the test of time. Papers from the ECCV 2000 meeting were eligible and Michael and his collaborators Hedvig Kjellstrom (nee Sidenbladh) and David Fleet received the prize for their paper "Stochastic Tracking of 3D Human Figures Using 2D Image Motion."

Amy Greenwald Awarded $15,000 for the Artemis Project from Microsoft Research and the National Center for Women & Information Technology

The Artemis Project, the departmental outreach program led by Amy Greenwald, recently received $15,000 from Microsoft Research and The National Center for Women & Information Technology (NCWIT) via the NCWIT Academic Alliance Seed Fund.

Artemis is a five-week summer day camp for high school girls in the Providence area that provides an introduction to computer science and technology. The learning process includes a range of both educational and confidence-building activities. Participants attend lectures from women scientists and other potential role models from both academia and industry. Artemis is provided at no cost to the participants, who come from predominately low-income, minority households.

The award will be used to expand Artemis into new directions. This year, in addition to the four coordinators from Brown, (Michelle Micallef, Natalie Serrino, Miranda Steele and Tashyana Thompson) a Boston University student (Katherine Zhao) was hired with the goal of expanding the program to the greater Boston area in future years.

The NCWIT Academic Alliance Seed Fund provides U.S. academic institutions with start-up funds to develop and implement initiatives for recruiting and retaining women in computer science and information technology fields of study. The Seed Fund was initiated in 2007 with start-up funding from Microsoft Research and to-date has awarded $265,450 in funding. University at Albany SUNY and Virginia Polytechnic Institute and State University also received awards from the NCWIT Academic Alliance in this round of funding.

NCWIT is a national coalition of over 170 prominent corporations, academic institutions, government agencies, and non-profits working to strengthen the computing workforce and cultivate technology innovation by increasing the participation of women. NCWIT's work connects efforts to increase women's participation in technology along the entire pipeline, from K-12 and higher education through industry, academic, and entrepreneurial careers.

The NCWIT Academic Alliance brings together more than 90 representatives from computer science and IT departments at colleges and universities across the country – spanning research universities, community colleges, women's colleges, and minority-serving institutions – to work towards gender equity, diversity, and institutional change in computing higher education.

Maurice Herlihy Named Fulbright Distinguished Chair in the Natural Sciences and Engineering

Maurice Herlihy has been awarded the Fulbright Distinguished Chair in the Natural Sciences and Engineering Lecturing Fellowship for the 2010-2011 academic year. He is visiting the Technion in Haifa, Israel, working on multiprocessor synchronization.

Maurice said, "The advent of multicore architectures has made this a particularly exciting time to be working on concurrency, and I look forward to establishing new collaborations."

Recipients of Fulbright awards are selected on the basis of academic or professional achievement, as well as demonstrated leadership potential in their fields. The Fulbright Program, America's flagship international educational exchange program, is sponsored by the United States Department of State, Bureau of Educational and Cultural Affairs and operates in over 155 countries worldwide. It was established in 1946 under legislation introduced by the late Senator J. William Fulbright of Arkansas.

Sorin Istrail Awarded Title of Professor Honoris Causa from Alexandru Ioan Cuza University

As part of the celebration of its 150th anniversary, Alexandru Ioan Cuza University in Iasi, Romania, presented an honorary professorship (Professor Honoris Causa) to alumnus Sorin Istrail. "I am very humbled and honored to receive this recognition," he said.

Sorin's alma mater, where he earned a B.S. in computer science in 1975, was also celebrating the centennial of its Alexandru Myller Mathematical Seminar. Sorin helped commemorate the anniversary by participating in the seminar as an invited lecturer, delivering a technical talk titled "Concepts of Mathematical Rigor for Algorithms in De-randomization, Statistical Physics, and Molecular Biology."

The lecture's first topic explored how to construct a random walk deterministically. What seems paradoxical can be well defined, Sorin said, if one asks for a construction that cannot be distinguished from a random walk by logarithmic space computation. Sorin presented his construction of universal traversal sequences (introduced by Steve Cook) for the de-randomization of algorithms employing one-dimensional random walks, published in the Proceedings of the 1988 ACM Symposium on Theory of Computing (STOC).

The second topic was the existence of analytical closed-form formulas for the partition function in the statistical mechanics of the Ising model, the most studied model in statistical physics. Sorin outlined his negative solution of the notorious 3D-Ising Model Problem showing that "the statistical mechanics world of the Ising model is flat!" His proof showed axiomatically the Ising model "equation": "Non-Planarity + Translational Invariance = NP-completeness (of the partition function)." This result, published in STOC 2000, ruled out, in the computational intractability sense, the existence of exactly solvable models for every 3D model (even strongly, every non-planar model).

He concluded his lecture with results from his 2004 paper in collaboration with Eric Davidson (Caltech) on the logic functions of the genomic cis-regulatory code (published in the Proceedings of the National Academy of Sciences), and 10 years of research together (five while at Brown) on the search for the fundamental algorithm to predict the functional meaning of regulatory DNA.

Sorin is the Julie Nguyen Brown Professor of Computational and Mathematical Sciences and Professor of Computer Science, as well as director of the Center for Computational Molecular Biology. His research focuses on computational molecular biology, human genetics, medical bioinformatics, statistical physics and complex systems, combinatorial algorithms, and computational complexity.

He is co-editor-in-chief of the Journal of Computational Biology, co-founder of the RECOMB conference series, co-editor of the MIT Press Computational Molecular Biology book series, and co-editor of the Springer Lecture Notes in Bioinformatics book series.

Chad Jenkins Promoted to Associate Professor

The Department is excited to announce the promotion of Odest C. (Chad) Jenkins to Associate Professor with tenure, effective July 1, 2010. "Chad's promotion recognizes his outstanding research work, innovative teaching, and exemplary service," said Department Chair Roberto Tamassia.

Chad joined our department in 2004 after completing his Ph.D. at University of Southern California, his M.S. in computer science

from Georgia Institute of Technology and his B.S. in computer science and mathematics from Alma College.

Selected as a Sloan Research Fellow in 2009, Chad is also a recipient of a Presidential Early Career Award for Scientists and Engineers (PECASE) for his work in physics-based human tracking and of a CAREER Award from the National Science Foundation (NSF) for his work on robot learning from multivalued human demonstrations. He received Young Investigator awards from the Office of Naval Research (ONR) for his research in learning dynamical primitives from human motion and from the Air Force Office of Scientific Research (AFOSR) for his work in manifold learning and multi-robot coordination.

Chad's research addresses problems in robot learning and human-robot interaction, primarily focused on robot learning from demonstration, as well as topics in computer vision, machine learning, and computer animation. Videos of Chad's current research work can be found on the YouTube channel of the Brown Robotics Group.

In 2009, Chad co-authored Creating Games: Mechanics, Content, and Technology with Brown CS alum Morgan McGuire. He has also authored chapters for several other books, including *Data-Driven 3D Facial Animation*, *Human-Robot Interaction* and *From Motor to Interaction Learning in Robots*.

"I consider myself very fortunate to be a part of the Computer Science Department at Brown," said Chad. "My promotion is both a reflection of the supportive culture and community spirit of the department as well as the hard work of the students and post-docs who have been in my group. I look forward to continued success at Brown, especially with our new robotics lab space in the CIT building."

Ben Raphael and Eli Upfal Receive NSF Grant to Develop Techniques for Analysis of DNA Sequence Variants

The National Science Foundation (NSF) awarded a research grant, in the expected amount of $500,000, to Ben Raphael and Eli Upfal to develop robust algorithmic and statistical techniques for the analysis of DNA sequence variants in the context of known and novel gene-gene interactions. These techniques will allow biomedical researchers to identify DNA variants associated with risk for various diseases, including cancer.

Algorithms developed in this project will be implemented and released as open-source software for use by the biological and medical community. The project will also support the training of graduate students and undergraduate researchers.

Eli Upfal Selected as Chalmers Jubilee Distinguished Visiting Professor

Eli Upfal was recently invited to serve as the Chalmers Jubilee Distinguished Visiting Professor for 2010. Chalmers is a Swedish university of technology in which research and teaching are conducted on a broad front within technology, natural science and architecture.

The Jubilee Distinguished Visiting Professor Chair was created by the Swedish government when Chalmers University celebrated its 150th anniversary in 1979. To goal of the visiting chair is to bring new skills to the University while strengthening international relations.

According to Eli, "I look forward to working on interesting research problems with Devdatt Dubhashi and his students. There are also exciting opportunities to create new collaborations. Chalmers has been one of the world's leaders in statistical research and I look forward to collaborating on new ways to integrate statistical concepts and methods in algorithmic research. This is especially useful in applications with large datasets, such as computational biology, web browsing and social networking."

Pascal Van Hentenryck Receives Philip J. Bray Award for Teaching Excellence in the Physical Sciences

Pascal Van Hentenryck received the Philip J. Bray Award for Teaching Excellence in the Physical Sciences. In addition to the recognition of his achievements, Pascal will receive a professional development fund of $2,000 for each of two years.

The award was presented at the 2010 University Awards Ceremony, co-sponsored by the Dean of the Faculty, the Dean of the Graduate School, the Harriet W. Sheridan Center for Teaching and Learning, and Computing & Information Services, Brown University. In announcing the award, Rajiv Vohra, Dean of the Faculty, noted Pascal's "profound and sustained impact on undergraduate education in computer science at Brown."

Pascal said "This award is a credit to the department and how it nurtures young faculty members in undergraduate education. I was very fortunate to benefit from role models like Andy, Philip, Leslie Kaelbling, and Dave Sklar. And, of course, we all greatly benefit from teaching highly motivated and gifted undergraduates who are constantly challenging us. It is amazing how much influence they have on the classes."

The Faculty Teaching Excellence Awards recognize Brown faculty members for sustained and continued excellence in undergraduate teaching. Awards are made in each of the four major areas of the curriculum: humanities, life, physical and social sciences. The awards are named for past faculty members recognized for their teaching achievements: John Rowe Workman (Humanities), Elizabeth LeDuc (Life Sciences), Philip J. Bray (Physical Sciences), and William G. McLoughlin (Social Sciences).

The following members of the department were also recognized for their teaching contributions at the Awards Ceremony: Associate Professor (Research) and Vice Chair Tom Doeppner (Sheridan Faculty Liaison) and Ph.D. students Nathan Backman (Sheridan Teaching Certificate I), Yuri Malitsky (Sheridan Teaching Certificate I) Deqing Sun (Sheridan Teaching Certificate I) and Justin Yip (Sheridan Graduate Student Liaison).

BU, Brown and UC Irvine receive $3 million NSF grant

Computer scientists from Boston University, Brown University and the University of California, Irvine, will collaborate on a grant from the National Science Foundation (NSF) in the anticipated amount of $3 million to investigate "trustworthy interaction in the cloud." The cloud refers to Internet-based outsourced computation (popularly know as cloud computing), whereby shared resources, software, and information are provided to computers and other devices on demand.

As one of the most promising emerging concepts in information technology, outsourced computation is transforming how IT is consumed and managed, yielding improved cost efficiencies and delivering flexible, on-demand scalability. However, despite the relatively fast growth and increased adoption of clouds, aspects related to their security, privacy, and economic value proposition remain largely unanswered and are regarded by some technology experts as impediments to broader acceptance of this approach to computing.

"Developing the right mechanisms for the specification and verification of trust-enhancing service-level agreements in the cloud will avert conflicts among cloud market stakeholders," says Azer Bestavros, lead principal investigator and professor of computer science at Boston University. "Doing so will also improve the utility and hardness of our cyber-infrastructure, with significant benefit to our economy and society."

"As more and more data is being stored in the cloud, keeping that data private is becoming critical, especially for applications in finance and medicine," says Michael Goodrich, principal investigator and Chancellor's Professor at the University of California, Irvine.

The project supported by the NSF grant will address these concerns by examining the feasibility of extending cloud service-level agreements to cover aspects such as integrity of outsourced services, information leakage control, and fair market pricing. The project will also explore mechanisms that verify trust-enhancing service-level agreements are being followed and develop "trustworthiness" guarantees and tradeoffs to cloud customers and system integrators that are both practical and useable.

"We envision a new generation of trusted cloud computing services where users will be able to verify the integrity of their data stored in the

Recent PhDs

Warren Schudy

Crystal Kahn

Aparna Das

cloud and the correctness of computations performed in the cloud," says principal investigator Roberto Tamassia. Tamassia is chair and Plastech Professor of Computer Science at Brown University.

The project's co-principal investigators include Leo Reyzin, associate professor, Jonathan Appavoo, assistant professor, and Nikos Triandopoulos, research assistant professor, at BU and Anna Lysyanskaya, associate professor, and Rodrigo Fonseca, assistant professor, at Brown.

In exploring these cloud computing-related issues, the team will collaborate with researchers at leading IT industrial labs at IBM, Microsoft, NetApp, RSA (the security division of EMC) and VMware. The project also will involve BU's Center for Reliable Information Systems and Cyber Security (RISCS) and the new Massachusetts Green High-Performance Computing Center (MGHPCC) to examine broader implications and impacts of cloud technology on society.

The project's ultimate goal is to define a viable marketplace for cloud computing resources in which users are assured that the services they acquire meet their performance, security, and privacy expectations.

Ph.D. Student Ryan Tarpine Wins Grand Prize in Dyalog Programming Contest

Ryan Tarpine received the grand prize in the Dyalog 2010 Worldwide Computer Programming Contest. The purpose of this contest was to encourage students and others to investigate APL (A Programming Language). According to the contest's judges, Ryan's entry won because, "Although Ryan was new to APL, he writes that he does most of his programming in functional languages, and there is clear evidence in his code that this helped him put his best foot forward when taking advantage of APL. Ryan's code was modularized in a way which allowed him to take advantage of operators including the power operator, to derive concise and elegant expressions. His code was a joy to read and looked as if it would be easy to maintain. For his efforts, Ryan received a trip to the APL 2010 conference in Berlin, in addition to the $2,500 cash prize".

PhD Profiles

Kiran Pamnany

Kiran completed his undergraduate degree at Bangalore University, India, in 1994. After ten years in the software industry, he returned to school to pursue a growing interest in research, and began his PhD at Brown in 2005. Kiran's research is focused on the problem of writing correct concurrent programs, with an emphasis on correctness before performance. With his adviser, Professor John Jannotti, Kiran is developing a system that uses program analysis directed runtime scheduling to enable the safe concurrent execution of programs developed to a serial programming model.

This system supports his thesis, which he expects to defend in the coming semester. Kiran's other research interests include embedded and realtime systems and information security.

Micha Elsner

An online chat room can be a confusing place, with many different conversations going on at once. This can make it hard to match questions with their answers, or figure out who is talking to whom. Micha Elsner's research aims to separate out individual threads of conversation from the chat. By modeling the way writers provide context to make their statements intelligible, his system can decide whether a proposed conversational thread hangs together or not. The same information can also come in useful for summarization, since the computer can use it to check if a candidate summary has enough contextual information to make sense.

Elsner hopes to continue to study conversations and other long texts, helping natural language processing to scale up from sentence-by-sentence analyses and gain a better idea of how sentences relate to one another. He hopes his research will enable not only better summarization, but also information extraction and automated editing. He plans to graduate in early 2011.

Yuri Malitsky

Yuri received his BS in computer science from Cornell in 2007, and is now a fourth year PhD student working with Meinolf Sellmann on automatic instance-specific algorithm configuration. This research challenges the paradigm of manually tweaking parameters to improve a solver's performance. Instead the project builds off the observation that different strategies tend to do better on different problem instances. Yuri proposes a new methodology that applies machine learning techniques to the development of combinatorial optimization solvers. This approach first automatically clusters training problem instances and then optimizes a different solver for each group.

Therefore, when a new problem instance needs to be evaluated, it is first classified and then evaluated with the appropriate algorithm.

This research has been empirically shown to provide significant performance improvements in a variety of problem domains and even for state-of-the-art solvers like Cplex.

Planning to defend his research in the Spring of 2011, Yuri is now searching for post-doc positions where he hopes to continue applying machine learning techniques to improve the performance of commonly used algorithms.

P4P: An Experiment in Syntax

By **Shriram Krishnamurthi**

There are things programming languages people don't talk about in polite company. Not very many things, but things nonetheless. Very few things, in fact. Come to think of it, just one: syntax.

So it is with some trepidation that I confess I've been thinking about syntax a lot lately. In particular, I've been experimenting with an alternate syntax for Racket, which I call P4P. P4P reduces the parenthetical burden, makes the language appear more traditional, and respects indentation, though not in the way you think. It does all this while retaining the essence of Racket syntax, and even offering a high degree of tool reuse.

1 A Thought Experiment

Consider the following Racket function definition:

```
(define (square x)
  (* x x))
```

Imagine you were asked to consider reducing the number of parentheses in Scheme. Well, we certainly can't remove the parentheses around the multiplication, because it has variable arity; nor those around the function header, since we wouldn't be able to tell whether * is an argument or not. But it would appear `define` contains all the information we need; that is, we could write

```
define (square x)
  (* x x)
```

Except, of course, that this has now made our language ambiguous. If instead we were to write

```
define (square x)
  (* x x)
(+ x x)
```

what do we mean? Is the addition expression a second body in the definition of `square`, or is it a separate top-level expression (with `x` unbound)?

Aha, you might think: "Didn't he say *indentation* earlier?" And indeed, that would help us distinguish

```
define (square x)
  (* x x)
  (+ x x)
```

from

```
define (square x)
  (* x x)
(+ x x)
```

and decide the earlier option (where `(+ x x)` had uncertain indentation) was ambiguous and hence erroneous.

Except that's not what we're going to do. Read on.

2 Examples

Before we dive into details, let's see some running P4P programs. I intentionally show only code, not output. If there is any doubt as to what these programs mean (and you aren't just trying to be ornery), P4P has failed.

First, a few variable and function definitions:

```
defvar: m = 10
defvar: this-better-be-6 = add(1, 2, 3)
defvar: this-better-be-0 = add()
deffun: five() = 5
deffun: trpl(x) = add(x, x, x)
deffun: g(a, b, c) = add(a, b, c)
```

Commas – yes, commas! We'll have more to say about them below.

Anonymous functions:

```
deffun: d/dx(f) =
  defvar: delta = 0.001
  fun: (x) in:
    div(sub(f(add(x, delta)),
            f(x)),
        delta)
```

Conditionals:

```
deffun: fib(n) =
  if: numeq(n, 0)
    1
  elseif: numeq(n, 1)
    1
  else:
    add(fib(sub1(n)), fib(sub(n, 2)))
```

Structures:

```
defstruct: memo has: (key, ans)
```

Sequencing (look for `do:`):

```
deffun: memoize (f) =
  defvar: memo-table = box(empty)
  fun: args in:
    defvar: lookup =
      filter(fun: (v) in:
                  equal?(args, memo-key(v)),
             unbox(memo-table))
    if: empty?(lookup)
          do: (
            set-box!(memo-table,
                     cons (make-memo(args,apply(f, args)),
                          unbox(memo-table))),
            apply(f, args))
else:
          memo-ans(first(lookup))
```

Expressions in function position:

```
defvar: this-better-be-9 = {fun: (n) in: mult(n, n)}(3)
```

Local bindings:

```
  let:
      x = 3,
      y = 2
     in:
      +(x, y)

  let*: x = 3, y = x in: add(x, y)
```

```
letrec: even = fun: (n) in: if: zero?(n) true else: odd?(sub1(n)),
        odd = fun: (n) in:
              if: zero?(n)
                false
              else:
                odd?(sub1(n))
  in: list(odd?(10), even?(10))
```

3 The Central Idea

P4P hinges on one central idea and its consequences.

First, the idea: let's get rid of implicit-`begin`-ness. Where we need a variable number of terms, write a `do:`. This small change suddenly eliminates the ambiguity that pervades Racket parsing and forces parentheses to clarify intent. The odds are that the extra typing engendered by `do:` will be offset by the reduction in typing parentheses.

Once we have made the syntax unambiguous *without the help of parentheses*, we can get rid of the parentheses themselves. That is, keywords like `deffun:` are sufficient to tell us what shape of terms to expect in legal programs. (Of course, every language construct must follow this property – even `do:`.)

This results in a pleasant invariant about parenthetical structure. In Racket, Scheme, and Lisp, functions are notorious for trailing off into a saga of parentheses (which in Racket are broken up by the odd square-bracket, which sometimes makes maintenance even more painful). In P4P, the only closing parentheses are from *expressions*, because the language's constructs (other than `do:` and its kin, which anyway use braces) do not contribute any. Thus, the parenthetical depth is precisely the same as the function nesting depth. For beginners, in particular, since this rarely exceeds 2–3, neither does the number of adjacent parentheses. For instance, in Racket, a typical memoized Fibonacci function ends in

```
  (+ (memofib (sub1 n)) (memofib (- n 2)))])))))
```

whereas the equivalent in P4P (using `if:`s and `elseif:`s in place of Racket's `cond`) ends (using the same operators, though P4P also defines `add` and `sub`) in

```
  +(memofib(sub1(n)), memofib(-(n, 2))))
```

(The one parenthesis not accounted for on this line itself is the invocation of `memoize`.)

Those old enough to remember Pascal will know this isn't the whole story. Pascal enabled programming language course instructors to ask students such world-class exam questions as the value of

```
IF false IF false THEN 2
```

(taking some liberties with Pascal syntax): the question being, of course, which `IF` the `THEN` associates with. If you're thinking now, finally, P4P will rely on indentation, you're wrong again (in the land of the one-armed `IF`, the people go blind from squinting). Rather, the P4P equivalent of this expression is simply illegal. If you want a one-armed conditional, use `when:` or `unless:`. Students, rejoice!

4 Design

Now I present some design decisions and design choices. Decisions are those I believe in and would change only under duress; choices are points of flexibility where I can be talked into alternatives.

4.1 Decisions

4.1.1 Embracing Prefix

We remain unabashedly prefix. By doing so, we circumvent all decisions about precedence, binding, associativity, and so on. Some initial grumbling may ensue when confronted with code like `+(1, 2)`, but this seems much less strange after you have seen `append(list1, list2)`. Bootstrap anyway wants students to understand that exalted + is just another operation – just like lowly `append`.

4.1.2 Adopting Racket's Token Syntax

By not permitting infix, we are free to be generous about token names: `append-string`, `overlay/xy`, and `d/dx` are available. However, there is no reason to preclude `e^<i*pi>-1`, either. In short, we use Racket's token syntax, which will simplify interoperation with traditional, parenthesized Racket.

4.1.3 Keeping Parsing Predictable

Despite the lack of parentheses, the parser is top-down and syntax-directed. It has only one token of lookahead, in this one case: when given an identifier in expression position, it has to look ahead for a left-parenthesis to determine whether or not this is an application. This is common in other languages too. If the input stream (file, REPL interaction, etc.) ends after the identifier, P4P treats it as a variable reference. (This ambiguity will affect tools like the kill-s-expression key-binding: if it faces an identifier, it will have to check whether the identifier is followed by an argument list.)

One potential source of ambiguity is the function position of an application being a non-identifier expression. In such cases, the expression must be wrapped in braces. Because the use of expressions in function positions is not common, this is a small price pay. Note that functions passed as arguments are bound to identifiers, so they will not suffer from this burden; the problem similarly disappears if the expression were first bound to a name (which might clarify intent).

4.1.4 Leaving the Semantics Untouched

This is *purely* about syntax. The semantics of P4P is precisely that of Racket. For instance, the P4P equivalent of `begin` currently allows only a sequence of expressions; if Racket began to permit definitions before expressions, so would P4P. Even naming stays untouched: if tomorrow structure constructors were to no longer be preceded by `make-`, that would be just as true of P4P.

4.1.5 Attaining Arity Clarity

Function invocations are delimited. Therefore we neither need to a-priori fix arity nor need types to tell us what the arity will be. Despite this, we can have functions that unambiguously consume multiple arity, just as in Racket: `+(1, 2)`, `+(1, 2, 3)`, `+(1)`, and `+()` are all legal P4P expressions with the expected meanings.

4.1.6 Adopting Indentation Without Semantics

I increasingly view emphasizing good indentation as critical. In some languages, however, indentation controls semantics. I view this as a mistake.

In P4P, instead, the semantics controls indentation: that is, each construct has indentation rules, and the parser enforces them. However, changing the indentation of a term either leaves the program's meaning unchanged or results in a syntax error; it cannot change the meaning of the program. I believe this delivers the advantages of enforced indentation while curbing its worst excesses.

There is a pleasant side-effect to this decision: the parser can be run in a mode where indentation-checking is simply turned off. (Obviously, this is meaningless to do in a language where indentation controls semantics.) This can be beneficial when dealing with program-generated code. Thus, it preserves the Lisp tradition's friendliness to generated code while imposing higher standards on human programmers.

4.1.7 Reusing the Tool Chain

P4P is implemented entirely using existing high-level Racket tools: it is defined entirely in terms of (a particular pattern of) `syntax-case` and some lower-level syntax-processing primitives. It does not define a lexer or LR-parser. I initially viewed this as a choice, but I have come to view this as a decision: this is the best way to ensure fidelity to Racket syntax.

4.1.8 Avoiding Optional Syntax

P4P does not have any optional syntax. I believe this makes it easier to teach people to program: they want clear instructions, not "You can do this, or you can do that...you can do whatever you want!" (If they were ready to do whatever they wanted, they wouldn't be asking you.) These trade-offs are best left to semantic and design levels, not syntax. The only options in P4P are thus semantic choices: e.g., you can use or leave out `elseif:` terms in a conditional, but that is a function of your program's logic, not your syntactic whimsy.

4.1.9 Avoiding New Spacing Conventions

While P4P's spacing conventions can (and should) be understood in their own right, experienced Racket programmers can safely fall back on their knowledge of Racket syntax. This, for instance, tells us that both `deffun: f(x) = x` and `deffun: f (x) = x` are valid (and so, even, is `deffun: f(x)= x)`, but `deffun:f(x) = x` and `deffun: f(x)=x` will not have the presumed intended effect. I do not view this as problematic: beginners (both educators and students)

always ask about spacing conventions. Since using spaces around tokens is safe, there is an easy rule to follow, which also enhances readability. It would help for P4P's parser to be sensitive to the presence of special tokens and build in context-sensitive checks for them (e.g., if the first token after the function header is an identifier that begins with =, this should be caught by a special error case that admonishes the user to insert a space).

4.2 Choices

4.2.1 Distinguishing Keywords

P4P uses colons at the end of keywords. I believe the principle of distinguishing keywords is beneficial: it tells the user, "You are about to use a construct whose basic syntax, rules of indentation, and rules of evaluation may all be different from what you expect." The particular choice of colon is whimsical and free to change, though it was inspired by Python's use of colons (which is somewhat different). P4P does not prevent ordinary program variables from ending in :, though it would be silently frowning as it processed programs that took advantage of this liberty.

4.2.2 Using Syntactic Embellishments

There are many syntactic embellishments in P4P.

- = in `defvar:` and `deffun:` aren't necessary, but adding them seemed to immensely improve readability. In particular, they emphasize the substitution nature of these definitions.
- There is no = `in fun:`; I chose `in:` instead. This is because the argument list does not equal the body, but rather is bound in it. The choice of `in:` is thus not entirely whimsical, but is very open to improvement. Likewise, there is no = in `defstruct:`, but instead `has:`, to emphasize that a structure *has* the following fields.
- `do:` uses braces (rather than parens) to delimit its sub-terms. (Semi-colons between terms in the `do:` will never be enforceable, so `do:` uses commas instead.)
- Using the `def-` prefix for the definition constructs leaves open `fun:` for anonymous functions.
- The syntax of `fun:` feels a bit naked: one needs to really understand expression-ness to understand (beyond indentation) where a function ends. A pair of delimiters wrapping the entire body would reduce this anxiety.
- `if:` does not need any intermediate keywords at all. In their absence, however, the programmer would be reduced to counting the number of preceding expressions and checking parity to know what they were looking at. Intermediate keywords improve both readability and error-reporting (which are probably linked).

4.2.3 Handling Variterm Constructs

Some constructs, such as Racket's `cond, begin,` and `when,` contain a variable number of body terms. This makes it challenging to keep their parsing simple and predictable. I see two broad ways to handle these: what I call `if:`-style and `do:`-style. `do:`-style is the lazy option: it uses a delimiter pair (specifically, brackets) and brutally dumps the terms between the delimiters. `if:`-style instead uses carefully-designed intermediate keywords as guideposts to the parser. The brutality of the `do:`-style could be reduced by the use of intermediate keywords, but at that point the delimiters wouldn't be necessary any longer. (They wouldn't be *necessary*, but they may still be helpful, as the number or size of sub-terms grows large.) Constructs like `when:`, which frequently have multiple, imperative body terms, would be better served by the brutalist style, because otherwise programmers would have to write an additional `do:` inside the single body term most of the time.

4.2.4 Avoiding Closing Delimiters

Nothing in the language design precludes closing delimiters. However, because parsing is always predictable, there is no *need* for them, either (except for variterm constructs). Offering them could improve error reporting.

4.2.5 Not Specifying the Indentation of Parenthetical Pairs

P4P currently does not enforce any indentation convention on parenthetical constructs. Indeed, I wonder to what extent the Scheme antipathy towards putting closing delimiters on separate lines is because of just how many darn ones there are. If the only closing delimiters are for constructs that need them (such as `do:`), it may even – gasp – be *good style* to put them on distinct lines, lining up with the opening keyword.

5 Indentation...Rules!

There are only three indentation rules in P4P: SLSC, and SLGC, and SLGEC. These stand for *same-line-same-column*, *same-line-greater-column*, and *same-line-greater-equal-column*, respectively. As you read more about these, you may find them insufficiently restrictive. Keep in mind that indentation rules are contravariant to language size: sub-languages (such as teaching languages) can enforce many more restrictions on lines and columns.

SLGC is the fundamental rule of indentation. As the name suggests, each sub-term must be either on the same line or (if not on the same line) indented to the right from the head term. The same-line part enables one-liners, though a teaching language might want to prevent excessively long lines – for instance, by disallowing the same-line part entirely for some constructs. In fact, the syntactic effect of SLGC is a little subtle: it means the first few arguments can be on the same line as the operator, while all subsequent ones must be indented, like so:

```
+(1, 2,
  dbl(4),
  dbl(dbl(8)))
```

SLSC is used more rarely, when we want rigid alignment. Currently, only `if:` uses SLSC for its internal keywords (`elseif:` and `else:`).

Finally, SLGEC was added for internal keywords that are not the same width as the main keyword. One might want to write

```
let: x = 3
 in: +(x, x)
```

to line up the colons, or instead

```
let: x = 3
in: +(x, x)
```

to keep the code from drifting rightward. (Of course, the programmer can put the `in:` on the previous line, too.) P4P sees no need to choose between these two indentation styles. Hence SLGEC permits an indentation of zero or more.

The decision to use SLGC and not SLSC for, say, argument lists may be surprising. It suggests the following is considered acceptable:

```
deffun: f (x) =
  +(dbl(dbl(x)),
    dbl(x))
```

This looks odd, but consider instead this case:

```
defvar: mfib =
  memoize(
    fun: (n) in:
      ...
```

In other words, when a function has "fat" parameters, we don't want to force rightward drift (or effectively impose shorter function names). Thus, we only expect arguments be farther to the right than the beginning of the function name, not necessarily "within" the argument's parentheses.

In practice, it has proven more pleasant to impose a slightly stricter rule for SLGC: to demand an indentation of at least two spaces, not just one. Two spaces increases readability (Python programmers often use four); it also means egregious

```
deffun: f (x) =
    + (dbl(dbl(x)),
       dbl(x))
```

is illegal, and must instead be at least

```
deffun: f (x) =
  +(dbl(dbl(x)),
    dbl(x))
```

One consequence of the relative laxness of SLGC – which a teaching language might want to tighten – is that P4P doesn't enforce that the immediate sub-expressions in an `if:` are at the same level. Thus, both

```
if: test1
    e1
elseif: test2
    e2
else:
    e3
```

and

```
if: test1
   e1
elseif: test2
          e2
else:
     e3
```

are legal, as different collections of people may prefer different coding styles.

6 On Groves and Brooks (or, Trees and Streams)

The Lisp bicameral syntax tradition is based on processing *trees*. The parentheses chunk tokens into well-formed trees, and the parser chunks these into valid trees. It's parentheses – and thus trees – all the way down.

Except, it isn't. A file is not a tree. Thus, sitting outside every Lisp parser of popular imagination is another parser that operates, instead, on *streams*.

Happily, Racket provides a middle-ground: files without explicit wrappers can be written in `#lang`, but `#%module-begin` turns this back into a tree.

This mapping enables the P4P parser to leverage the Racket macro system to bootstrap. P4P removes tokens sequentially, using a slack term in every pattern to match the rest of the stream; each construct's parser returns a tree and what remains of the stream after it is done processing.

Oh, and commas. Of course, the Racket tokenizer converts commas to `unquote`. In Racket, the `unquote` is followed by a single tree; in P4P, it is followed by an arbitrary *undelimited* expression. So P4P lets Racket turn commas into `unquotes`, and then simply returns the subsequent tree (in Racket's terms) to the front of the token stream, for continued P4P parsing.

7 Error Reporting

I have invested (almost) no time into error messages, yet.

By being a *macro* over existing Racket, P4P inherits much of Racket's context-sensitive error-reporting. Naturally, having additional clauses in P4P can improve error checking. For instance, in the current implementation,

`deffun: f "var" = 3` and `deffun: f(3) = 3` happen to be caught by P4P itself (which highlights the appropriate term), while other errors pass through to Racket, using its error messages and highlighting. (The expression `3(4)` *ought* to demonstrate this, but currently fails on an internal error instead.)

Because P4P's parsing is done through streams rather than trees, it is unclear how much of Ryan Culpepper's infrastructure for strengthening tree-based patterns to insert error checks will apply here. It is more likely that something analogous needs to be created for stream processing. In the best case, of course, Ryan's work will carry over unchanged. Either way, this will be a fruitful area for further examination.

Finally, one known problematic case is this: when a comma-separated list fails to have a term between the (intended) penultimate comma and the closing parentheses (e.g., `f(x, y,)`). This is an unfortunate consequence of P4P's attempt to reuse the Racket toolchain, and will need special support. This is a place where EOPL's sllgen parser has no problems, because it natively implements both scanner and parser.

8 Syntax Extensions

It would be easy to add new constructs such as `provide:`, `test:`, `defconst:` (to distinguish from `defvar:`), and so on.

The current design of P4P also does not preclude the addition of syntactic enhancements such as type declarations, default argument values, and so on. It is presumably also possible to add support for Racket keywords and reader extensions.

One particularly curious form of syntactic extension would be to use fully-parenthesized terms in some contexts. For instance, we might add a `racket:` construct that is followed by a fully-parenthesized s-expression. Because of the nature of P4P's syntax, this can be done without any ambiguity. One might even, say, decide to use P4P syntax to define macros for *parenthetical* Racket; the P4P versions of `syntax-rules` or `syntax-case` can exploit P4P's parenthetical sparsity *except* for the patterns themselves, which would be fully-parenthesized as they would in traditional Racket (and in the source they process).

Beyond this, it is in principle possible for developers to create macros for new P4P syntactic constructs. After all, P4P is already defined using just macros. However:

- The macro definer has to understand the stream-processing pattern, which is different from traditional tree-shaped macro processing.
- Even more importantly, the macro writer undertakes to create a construct that does not introduce syntactic ambiguity – a property that is guaranteed in Racket, but earned in P4P. (To be clear, a new Racket macro can be ambiguous: imagine an `infix` macro, which requires precedence rules for disambiguation. However, this ambiguity is limited to the inside of the new construct, and cannot affect terms past the closing parenthesis. In P4P, the effect may leak past the end of the construct.)

For these reasons, we will probably need to create a macro-definition facility: a `syntax-rules` for streams. However, that is not enough:

- The macro writer needs to check indentation. This may require a pattern language that is indentation-sensitive.
- The output of the macros will, by default, interact with the indentation checking of the underlying P4P language. One option is to have the macros respect this, though it will likely make them too difficult to write (because any loss of source location would leave the underlying P4P parser unable to perform checks, and hence forced to reject the program). A second option is to generate code in a P4P variant that doesn't check indentation. A third, perhaps best, solution would be to generate Racket code directly, just as the current P4P does: that is, the macro system would be an attached-at-the-hip, cooperating twin of P4P, rather than a layer atop it.

9 Conclusion

Racket has a excellent language design, a great implementation, a fine programming environment, and terrific tools. Mainstream adoption will, however, always be curtailed by the syntax. Racket could benefit from some exercise to reduce the layers of parenthetical adipose that engird its code. P4P is a proposal for how to do this without losing the essential nature of the Lisp syntactic heritage (and, indeed, bringing to the surface the streaming nature that has always been hidden within).

Alumni Update

Daniel Aliaga '91

was recently promoted to Associate Professor of Computer Science at Purdue. His research is primarily in the area of 3D computer graphics but overlaps with visualization and with computer vision. His research area is of central importance to many critically important industries, including computer-aided design and manufacturing, telepresence, scientific simulations, and education. He focuses on i) developing fundamentally new 3D model acquisition methods, and ii) combining his 3D acquisition methods with additional novel algorithms to produce pioneering new modeling and visualization frameworks.

Colin Blundell '03

is the recipient of the 2010 IBM Josef Raviv Memorial Postdoctoral Fellowship in Computer Science., an annual award made to a recent PhD recipient who shows exceptional promise for a research career in computer science or computer engineering. IBM received over 120 applications from around the world for the fellowship. After multiple rounds of thorough review, the selection committee identified Colin as the winner of the fellowship. He now works at the IBM T.J. Watson Research Center, primarily with Jose Moreira's Future POWER Systems Concept Team, and is interested in exploring high-performance computer architectures for emerging workloads.

danah boyd '00

On Wednesday, October 6, Brown Computer Science welcomed back one of its most illustrious alumnae: social media researcher danah boyd. Boyd graduated from Brown in 2000 after completing an honors thesis on how sex hormones affect the processing of depth cues and engagement with virtual reality. She subsequently attained a Master's Degree at the Massachusetts Institute of Technology Media Lab's Sociable Media Group and a PhD from the School of Information at University of California, Berkeley. She now works at Microsoft Research New England, where she studies social media, teen culture, and a host of other topics. Boyd returned to campus to give a seminar open to the public entitled "Youth-Generated Culture: Education in an Era of Social Media."


danah boyd

The seminar, presented by the Division of Campus Life & Student Services and the Harriet W. Sheridan Center for Teaching and Learning, covered a variety of topics related to boyd's work at Microsoft Research. Boyd's enthusiasm, intelligence, and ability to captivate an audience were all evident as she discussed the dynamics of online communities and the complications that have arisen now that social networks primarily cater to helping individuals connect with people they already know in physical space rather than strangers with shared interests. Boyd also shed light on the challenges faced by today's teenagers as they struggle to present themselves to their peers while simultaneously being judged by their parents, educators, college admissions officers, and future employers. The audience remained spellbound throughout the lecture, and after a tumultuous round of applause, many listeners approached the stage to ask boyd questions about social media, her research, and what her thoughts were on a spate of recent, nationally discussed tragedies related to online bullying.

This informal question and answer period resembled a session boyd had participated in earlier that day. Women in Computer Science (WiCS), an undergraduate student group devoted to supporting women pursuing degrees in computer science, happily welcomed boyd to a meet and greet a few hours before her seminar. The CIT library was packed with members of WiCS delighted to hear about boyd's experiences at Brown and beyond. Boyd regaled members of WiCS with stories about her time as an undergraduate student, which included eight independent study projects, one radical benefit concert featuring Tracy Chapman, and innumerable amusing stories featuring her advisor, Professor Andy van Dam. When asked about her current employer, Microsoft Research, boyd lauded the academic freedom and spirited culture she had found there. She described some of the Internet curiosities she and her colleagues were currently researching, including fashion blogging, a women-driven phenomena which—though wildly popular according to site traffic and advertising revenue—has been all but ignored by digital sociologists.

The approximately twenty members of WiCS who were able to drop in for at least part of the discussion were amazed by boyd's passion, sense of humor, and expertise. Both undergraduates and graduate students attended the event, and all agreed that boyd was by far one of the most interesting and approachable guest speakers WiCS has ever hosted. As one student later explained to WiCS coordinators Ashley Tuccero ('11) and Claire Kwong ('13), both of whom are avid readers of boyd's blog, boyd is "such an awesome, sweet, and intelligent person [that it's] no wonder that she's your hero".

For more information about danah boyd, see http://www.danah.org/.

For more information about Women in Computer Science, see http://cs.brown.edu/orgs/wics/.

Jeff Bergart and Marion Dancy

Michael Coglianese & Family

Saurya Velagapudi, Owen Strain, Allan Shortlidge

Elli Mylonas, Shawn Zeller, Leslie Zeller, Seth Landy

Aysun Bascetincelik, Onur Keskin

Jared Rosoff, Sharif Corinaldi

Adam Buchsbaum, Roberto Tamassia, Isabel Cruz, Mary Fernandez

Alex Shvartsman, Franco Preparata

danah boyd, Matt Hutson, Sharif Corinaldi

John Peha, Rosie Perera

Daniel Stowell, Eric Rachlin

Lawerence Chan, Suman Karumuri, Gil Benghiat

Tim O'Donnel

Spike Hughes, Lou Mazzucchelli, Adam Buchsbaum

Andrew Shearer, Matt Gilooly, Nate Webb

Keith Dreibelbis, Todd DeLuca, Haidee DeLuca

Amy Kendall & Family

Michael Black

Much of Michael's group has moved into the newly expanded and renovated Video and Motion Capture Lab on the first floor of the CIT. The new lab is significantly bigger and has more flexible space for experiments on human motion and body shape. The new lab includes a full-body 3D laser range scanner, an 8-camera HD video capture system, a Vicon motion capture system, a changing room, work space and storage space. The lab expanded into space previously occupied by CIS. Ongoing expansion on the first floor has put Chad Jenkins' new robotics lab right next door. This will enable closer collaboration between the vision and robotics efforts at Brown.

Tom Doeppner

Tom spent a good part of the summer completing a textbook on operating systems. It's published by Wiley (ISBN-13 978-0-471-68723-8) and is out now.

Maurice Herlihy

Maurice is on sabbatical this year. During September, he visited the Institute for Theoretical Computer Science at Tshinghua University in Beijing, China, and for the rest of the year he is visiting the Technion in Haifa, Israel. His is supported by a Fulbright Distinguished Chair in the Natural Sciences and Engineering Lecturing Fellowship.

John "Spike" Hughes

Spike went to SIGGRAPH in LA this summer, where he saw the usual Brown folks, but encountered a few in new/unusual places: First, former 17/18 TA Daphna Buchsbaum in the Emerging Technologies area, who was helping to show off a bicycle-wheel-mounted display ("Monkeylectric") along with Dan Goldwater and Kipp Bradford (former CS student, now Adjunct Lecturer in Engineering here at Brown), and second, Henry Kaufman (indirectly – his first child was due at the same time, so he didn't actually come to LA), who had his first-ever work in the Art Show.

David Laidlaw

David is experiencing some reentry symptoms after having been on sabbatical, mostly in Barrington, during the last year. He did make eight or 10 trips of varying lengths to take advantage of a less constrained schedule. While not on the road, he has been working to define a multiyear research project to study how cognition can be leveraged to improve visualization tools. The coming year promises to keep him busy. He'll be teaching CS16 – er, I mean CSCI 0160 – this Spring for the first time and will be the general chair for the IEEE VisWeek conference to be held in October 2011 here in Providence.

Claire Mathieu

This summer, Claire was proud to attend the PhD defense of her first graduating PhD student at Brown, Warren Schudy, who is now a post-doc at IBM Research. She was a panelist at a CAPP meeting for mid-career women in Computer Science. She spent a lot of time traveling during the summer, visiting the Theory group at Microsoft Research in Redmond, giving a talk and being given a tour at the INRIA-Microsoft research center (headed by Jean-Jacques Levy) in the suburbs of Paris, France, and stopping by Microsoft Research New England. In Redmond, she had the company of dozens of summer interns from Brown!

While vacationing in France, she spent a few days in Ardeche, where she learned that she is not good at planning a feasible backpacking itinerary, and that taxis called in the middle of nowhere out of necessity can be outrageously expensive. In addition, she still has not learned to slow down *every time* she passes automatic radars, and so, this summer like every summer, she got a speeding ticket and one point off her French driver's license. Those points supposedly regenerate themselves once three years have gone by without a ticket, but that seems hopeless. Finally, taken by a sudden cleaning frenzy, one day she used spray for screens and cleaned her laptop, screen, keyboard and all. It has not yet fully recovered and now goes on strike whenever she tries to tell it to go to sleep.

Barb Meier

The introductory animation course continues to be very popular, but the make up of students taking the class has changed over the years. The most notable change is the number of women. After six years of having exactly four women in the class of twenty each year, last year there were six women. This year, Barb has 10 women and 10 men, probably the only computer science course at Brown to achieve gender parity. Of the ten women, four are planning on double concentrating in visual art and either CS or another science. Only one is concentrating solely in computer science. Animation is a wonderful intersection between science and art. Research has shown that girls are often attracted to "computing with a purpose" (see *Unlocking the Clubhouse: Women in Computing* by Jane Margolis and Allan Fisher, MIT Press, 2002) and it appears that Barb's animation course is providing an outlet for some of these students to combine the creative with the technical. She is honored to be involved in this trend.

Don Stanford

Don is pleased to be back in the classroom again for his 9th semester teaching CSCI0020 (CS2)! CS2 is being taught for the first time in the CIT and Don says it is nice to be teaching in the home department. He is still consulting as the acting CTO at GTECH and teaching in the Engineering PRIME program together with a host of other community related activities, so retirement has certainly not been boring or filled with idle time. His head TA this fall is Megan Hugdahl, a CS concentrator who converted to CS after taking CS2 as a freshman 3 years ago. It's always gratifying when some of our students become concentrators as a result of dipping their toes in the CS waters.

Faculty Notes

Erik Sudderth

Assistant Professor Erik Sudderth and his wife Erika welcomed their son, Kyler Logan Sudderth on March 7.

Nikos Triandopoulos

Ph.D. Alums Olga Papaemmanouil & Nikos Triandopoulos welcomed Baby Christina, born on September 13th 2010. Olga in an assistant professor in the CS Department at Brandeis and Nikos is a principal research scientist at RSA Labs in Cambridge and also holds adjunct appointments at BU and Brown.

Industrial Partners Program

The IPP provides a formal mechanism for interactions between companies and students in the CS Department. Member companies benefit from superior visibility in the Department, exclusive access to event/interview space in the CIT Building and assistance with recruiting events; students benefit from specific information about opportunities for summer internships and permanent employment.

The department wishes to thank our Industrial Partners for their support:

Premier Partner

Adobe

Affiliates

Apple
eHarmony*
Facebook
Google
GTECH
Microsoft
NABsys, Inc.*
Oracle
TripAdvisor*
VMware
Zynga*

Start Ups

Cloudera*
Delphix*
DropBox*
LIPIX, Inc.
Mozilla*
NING*
Vertica Systems Inc.*

Individuals

Paul Edelman, Edelman & Associates

(new partners denoted with an *)

For more information about the Industrial Partners Program, contact:

Amy Tarbox
Program Manager
Telephone: (401) 863-7610
abt@cs.brown.edu

Ugur Cetintemel
Professor and IPP Director
Telephone: (401) 863-7600
ugur@cs.brown.edu

To learn more about the IPP visit: http://www.cs.brown.edu/industry

Connect with the CS Department:

Join the 'Brown University Computer Science' group on Facebook.

facebook

Regional Groups:
Brown CS in Boston
Brown CS in NYC
Brown CS in Seattle
Brown CS in LA/Southern California
Brown CS in SF Bay Area

LinkedIn

Computer Science at Brown University

Department of Computer Science
Brown University
Box 1910
Providence, RI 02912
USA

Ping!

First Name

Last Name

Class Year

Address

City

State

Zip

E-Mail

Where are you and what are you doing?

Let us know what's happening in your life! New job? Received an award?
Recently engaged or married? Use this form to submit your news or e-mail conduit@cs.brown.edu.

My news:

Mail to: Conduit, Department of Computer Science, Brown University, Box 1910, Providence, RI 02912 or conduit@cs.brown.edu