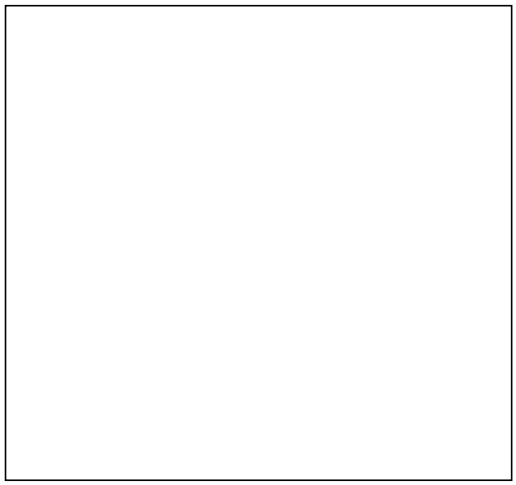
conduit!

Volume 6, Number 1 **Department of Computer Science** **Spring, 1997**
Brown University

FROM RANCH TO ROBOTS & THE COMPUTERS & THOUGHT AWARD



Leslie Pack Kaelbling

The Department is very proud to announce that Leslie Pack Kaelbling has been awarded the coveted IJCAI Computers and Thought Award for 1997. This award is given every other year to a young AI researcher who has contributed significantly to the field. Leslie is cited for her contributions to the semantic theory of information in embedded systems, to the development of programming tools for mobile robots, for algorithmic advances in the solution of partially observable Markov decision processes and for her application of reinforcement learning methods to intelligent embedded control. She is the author of one book, "Learning in Embedded Systems," *and the editor of another. She is the recipient of an NSF National Young Investigator Award and an NSF Presidential Faculty Fellowship. Professor Kaelbling gave the following talk as an inaugural speaker at a College Convocation on April 10. These University-wide forums are designed for faculty to talk about their life and work goals.*

I grew up on a ranch. Anywhere but in California, it would have been called a farm, since the only livestock we had was my horse and assorted dogs and cats. On the ranch, we grew tomatoes, apricots, garlic, and lettuce if we were feeling lucky.

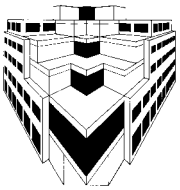
Farming involves (at least, it used to) a lot of solitary work, some physical, some not. Two kinds of people are happy being farmers: those whose minds don't need further stimulation and those whose minds generate their own stimulation. My father was the second kind of farmer; he was a full-time farmer and a part-time amateur philosopher (I'm always a bit suspicious of the professional ones) who liked to think about existentialism and stuff. Farming offers great opportunities for those who want to lead a reflective life of the mind.

As a kid, I did my share of farm work. And I, too, learned to lead a reflective life of the mind. My favorite farm job was irrigating: you adjust water flowing out of big metal pipes down the rows of tomatoes, watching to be sure it flows at the right rate and doesn't overflow (a side job is to whack gophers on the head with a shovel as they get flooded out of their holes; not pretty, but I did it). Irrigating afforded physical pleasures like running barefoot along a cool irrigation pipe on hot summer days and squelching mud between your toes. And it also afforded a lot of time to think. I learned to appreciate solitude and reflection.

I went through the local high school in something approximating the usual way. When I entered, I wanted to major in agriculture, but my mother made me do college prep instead; I thought she was being unreasonable. I got a surprisingly good education, especially given that maybe 10 percent of the students went on to college. I might have been valedictorian, except for my freshman-year B in drafting; I always had smudgy drawings.

> *"My favorite farm job was irrigating...a side job is to whack gophers on the head with a shovel as they get flooded out of their holes; not pretty, but I did it"*

When it was time for college, I applied to Stanford and Berkeley because they were nearby and MIT because it seemed exotic. My



guidance counselor asked me, "What's MIT?" I said, "The Massachusetts Institute of Technology." "Oh," said he, "a *technical* school." I had gotten this far without any idea that being female was an obstacle to pursuing any career I might wish. Then a woman from MIT called to recruit me. One of the first things she said was, "You know, the men here really aren't *that* bad." That really made me wonder.

I ended up at Stanford because I visited there and enjoyed a lecture on Mozart. I started out as a math major, but because my math background was pretty weak, I ended up in huge classes of math for engineers. It was probably useful, but it wasn't inspiring. Philosophy classes, on the other hand, were more intimate and supplied ample fodder for reflection. And philosophy at Stanford isn't all that different from math. Which led me to take and love computer science courses. It was like all the logic I had been playing with, but when you were done, you got a cool artifact that did something. I got a bachelor's in philosophy (because there was no CS major at the time) and a master's in computer science. The TA in one of my computer science classes was a graduate student named David Kaelbling. I ended up marrying him. (We didn't start going out until after the course was over, but I got an A even so.)

When it got to be time to find a job, I thought, "Boy, wouldn't it be great if someone would pay me to sit and think." But I didn't think it would ever happen. I looked for a job in the usual way, signing up for interviews in the career center. After a number of disastrous interviews, including one with a video-game company whose vision was that they should rewrite all their programs to run in less memory (since then memory has gotten cheaper and faster at an astonishing rate), I interviewed with SRI International. SRI used to stand for Stanford Research Institute; it was formerly affiliated with Stanford, but during the '60s and '70s the student body found its large involvement with defense research objectionable and it was divorced from the university. SRI is a think tank; it'll do any research anyone is willing to pay for. It chose Anaheim as the site for Disneyland and even looked into whether Uri Geller could really bend forks. Somehow I got hired as a researcher at SRI's Artificial Intelligence Center. And the great thing was, *they were paying me to sit and think*! I had to do my share of writing proposals and building demos, but mostly I got to sit and think—what could be finer? It was clear that this was the profession for me.

When I was interviewing at SRI, I told Nils Nilsson, the director of its AI Center, that I thought I'd eventually like to go back for a Ph.D. He said, "Do it now, before you get out of the habit of working hard." So I did a Ph.D. in computer science while I worked at SRI.

I had been interested in questions of mind and of intelligence, and of how (or if) it would be possible to make an artificial intelligence. At SRI, I had to roll up my sleeves and try. The AI Center was building a new robot, Flakey, and my job (along with a couple of other people with a similar lack of preparation) was to make it do something intelligent and to understand something about intelligence in the process—a serious first assignment for someone who wants something to think about.

AI and Robots

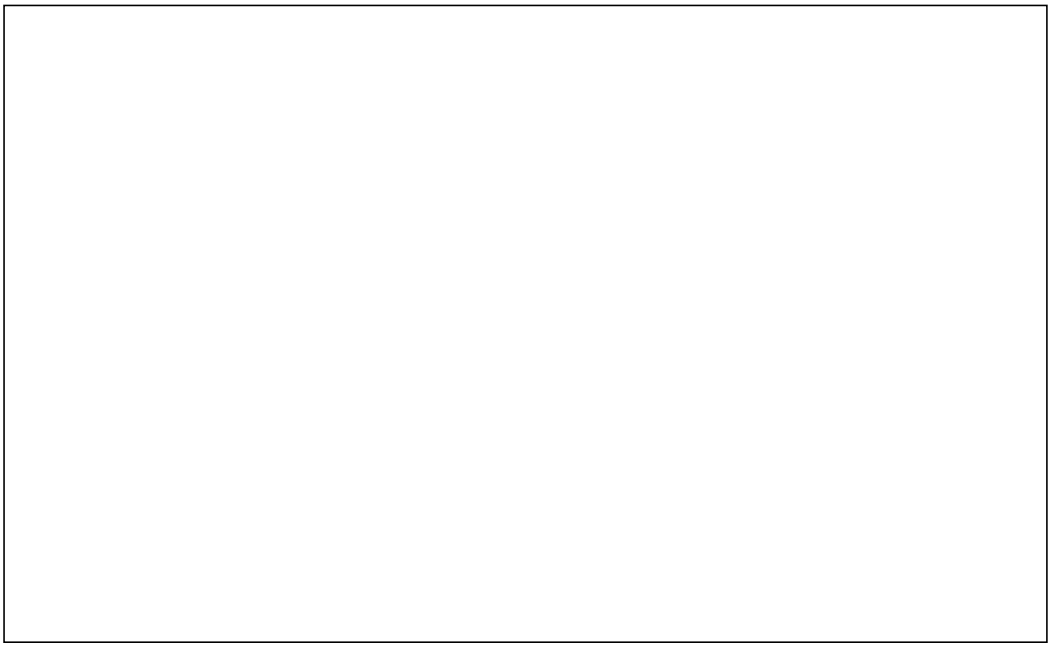
The current state of the art in "intelligent" robotics is pretty dismal. Most people who work on robotics do factory automation. They make robots that do the same thing over and over, with complete repeatability and reliability. The thing about these robots is that they are totally oblivious to the world around them. A robot that welds cars in an assembly line would happily weld an elephant if one were to come down the line. Now, this single-mindedness is fine in a factory environment, because things can be made perfectly predictable and an elephant never *does* come down the line. But I'm interested in robots that can act robustly in a wide variety of situations—that exhibit behavior you might be willing to call "intelligent."

"I'm interested in robots that can act robustly in a wide variety of situations—that exhibit behavior you might be willing to call 'intelligent'"

Most people derive their knowledge of robots from movies; "Star Wars" is a good example. So when they think of robots, they think of R2D2 and C3PO. Real robots are nothing like that. We're happy when we can get them to drive around a building and not run into

things; we're ecstatic when they can go to somebody's office without getting lost. Why is this so hard?

Traditional AI work has centered on the "intellectual part" of intelligence: the things, like playing chess and making detailed plans and using language, that seem to separate human intelligence from that of other animals. Methods for solving these problems are very abstract; they manipulate symbolic representations of the world, almost like thinking by composing sentences in your head. This may be the right strategy for generating "intellectual activity," though I have my doubts. In any case, it's clearly the wrong strategy for the much more pedestrian tasks that robots can't do well. It's much too cumbersome and slow; can you imagine having linguistic thoughts about your position in the lane you're driving in while trying to stay in the middle of the road?



Photo–John Foraste/Brown University

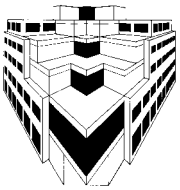
Leslie explores variables in CS31

My work has been in "situated" artificial intelligence. The idea is to think of an agent in terms of its interactions with the environment, rather than as a stand-alone system. There are two ways to go straight down a road: one is to measure exactly where you are, how the road bends, etc., then close your eyes and drive precisely down the road. The other is to have constant interaction with your environment, measuring your current position, making corrections, measuring again, etc. The first approach is attractive to a lot of people, but the second is easier to make work in the long run. In fact, I believe you always have to resort to the second approach, because you can never know your situation perfectly enough to plan exactly what to do in advance. So I concentrate on approaches to intelligent behavior that are based on constant measurement and correction.

Another big question in AI is the architecture of an intelligent agent. The romantic ideal is that there is one elegant mechanism behind intelligent behavior and that all we have to do is find it. Unfortunately, my guess is that intelligence is a lot messier than that. If we're lucky, there's a handful of basic techniques. If we're not, then intelligence is just made up of a big bag of ugly hacks, and it will be hard for humans to figure out.

In any case, intelligent systems will need to learn. We human designers can't anticipate in advance all the knowledge necessary for an agent to succeed in any environment. Artificial intelligence studies a number of different learning scenarios. The most basic one is *supervised learning*. Here the agent is given examples of how to behave and it merely has to learn to do those same behaviors. This is an interesting and hard technical problem, but it doesn't apply very well to robots; it's hard for humans to supply examples of behavior to robots, except at a pretty abstract level. My robot, Ramona, can't learn how to turn her wheels by watching me (and the watching part is pretty hard, too).

I work on a kind of learning called *reinforcement learning*. We give the robot not examples of behavior but "reinforcement" signals telling it when it's doing a good or bad job. I might give the robot a small negative reinforcement for running into the wall and a big positive one for bringing me coffee. This is a much harder problem than supervised learning, for at least two reasons. One is that the robot has to learn to take actions because of their long-term effects; we're all intimately familiar with the idea of suffering short-term pain for long-term gain, but learning to act on that principle is hard, for people and robots alike. Another difficulty is that the robot has to experiment, trying out a variety of different strategies in order to find the best one. Many humans aren't



very good at this, either. I used to have a colleague who was particularly bad at this. He knew a way to get from home to work and a way to get from home to the store. To get from work to the store, he would go via home, a number of miles out of the way. It never seemed to occur to him that there might be a better way.

Learning is such a cool idea that it's tempting to imagine that it will do all the work. But *tabula rasa* learning usually doesn't work very well. If there are no constraints on the thing to be learned, then the universe of possibilities is so large that an impossible amount of experience is required to learn the right thing. Natural systems (humans and other animals) have a lot of innate knowledge. Much of the vision and motor system is hard-wired, and the brain has all sorts of structure. Evolution has solved the problem of building in as much fixed structure as possible, while leaving the system plastic enough to adapt to a varied environment. For artificial systems, human engineers will have to play the role of evolution, building in the fixed structure and finding the right tradeoff between innate knowledge and adaptivity.

"This may be too hard a problem for us to solve. Although I believe that it's in principle possible to build artificial human-level intelligence, we may be too stupid to do it"

This may be too hard a problem for us to solve. Although I believe that it's in principle possible to build artificial human-level intelligence, we may be too stupid to do it. One strategy for approaching AI is to try to understand natural intelligence, then model or replicate it in artificial systems. This approach is attractive because there are real examples of intelligent systems to study. I'm taking a somewhat different approach that assumes that many different kinds of mental structures can give rise to intelligent behavior. Nature's solution has the character of many evolved systems: it seems to be a pile of tweaks and hacks, modifications of modifications, producing an unnecessarily complex architecture. My hope is that there is a simpler solution, one that we human engineers can eventually discover and implement.

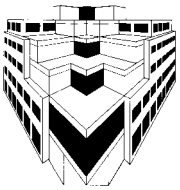
Teaching

I did a lot of TAing and teaching at Stanford and decided that lecturing is an important counterpart to reflection. After all that thinking, you have to tell someone! The best thing is to inspire and to be inspired by students. So, rather than continuing to work in industrial research, I came to Brown to be a professor. This was not just a whim; it takes a good reason for a native Californian (third generation) to move to the East! I had imagined that it was entirely paved from Boston to New York, and was gratified to find out that it's not. We live in the pseudo-country, halfway between here and Boston. We have a few acres of woods and my horse lives on a farm a couple of miles away. I even like the snow (and so does the horse)!

I have taught a number of courses here at Brown; I want to focus here on two slightly unusual ones.

Last fall, I co-taught a university course called "Making Decisions" with Steven Sloman from the Cognitive and Linguistic Sciences Department. I think it's important to make connections to other disciplines, and while I can't even begin to fathom things like literary criticism, I can try to reach as far as another science. In this course, I present some mathematical theories that purport to describe the best possible way to make decisions under uncertainty (and give helpful hints about how to deal with casinos and lotteries); Steven presents the psychological literature, showing that humans make decisions in a very different way. We use this difference to motivate discussions of rationality and what it means to make good decisions.

I also think it's important to engage humanities concentrators in the sciences. Many people take math and science under duress in high school and look forward enthusiastically to college life without science courses. But we should be able to teach science courses that are as superior to high-school science courses as college literature courses are to courses in spelling and vocabulary. Last fall was the first time I taught a non-technical audience. I knew I shouldn't get too mathematical, but I was unprepared for just how much trouble many of the students had with the math in the course. I was surprised, because we weren't using anything more sophisticated than high-school-freshman algebra. But I found that, although most students had pretty good



mathematical manipulation skills (once the right formulas were written on the page, they could get the answer), they were very weak at going from an abstract problem description to the mathematical formulation. That is, they couldn't do word problems. I think this points to a major weakness in high-school math preparation, which tends to emphasize technique over real understanding. I hope this insight will help me do a better job with this course next year.

Another course I teach is called "Building Intelligent Robots." Much of computer science addresses a perfect formal world in which everything is completely reliable and repeatable. In order to debug your program, you run it until it fails, then systematically track down the failure and try again. Students learn this skill early on, and with it comes the attitude, "With enough patience, I can fix anything." Once your programs get connected to the outside world, though, the abstraction fails. Maybe a certain bug in your word processor happens only when you type a particular word, or type that word in a particular rhythm. Then it gets much harder to figure out what's wrong. With robots things are even worse. Students in my class build robots out of Legos, a microprocessor, and some really cheap sensors we buy at Radio Shack (I *do* love going in there, being female, and out-geeking the clerks; so I guess I really am a geek, in disguise I hope). Nothing works the same way twice. Light sensors are unreliable, and the light changes between day and night (students develop their programs late at night, but when it's time to give a demo for class, the sun is up and the programs don't work anymore).

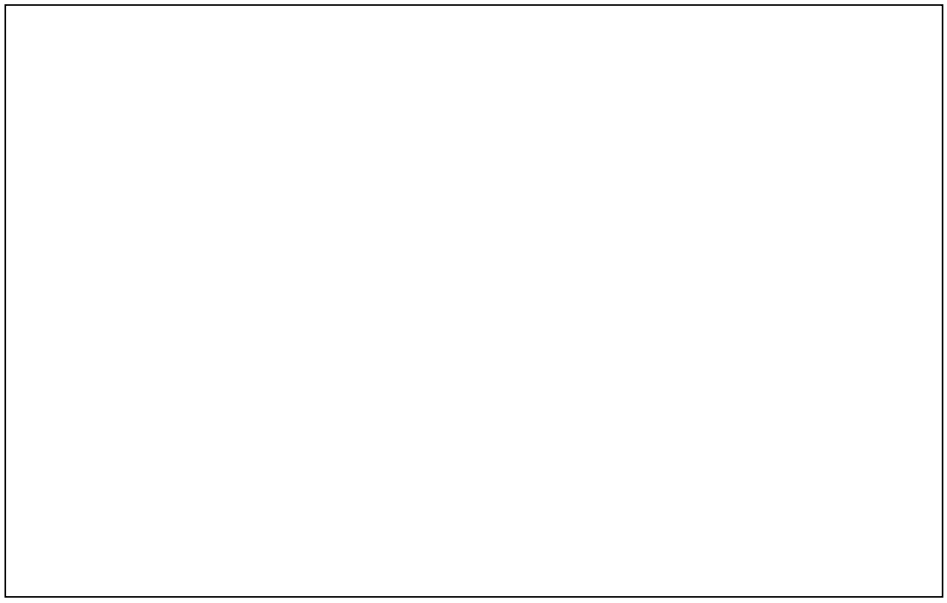
At first, the students are really frustrated, thinking there's no way they can get this crummy hardware to do anything reliable. But we teach them that, with the right kind of programming strategy, focusing on the interaction between the robot and its environment, you can often get very robust high-level behavior from very unreliable low-level components. It's especially useful for scientists trying to build artificial intelligence to see things this way, since it is how natural systems achieve their remarkable robustness.

Closing Thoughts

I never thought of myself as a *female* computer scientist until I came to Brown and everyone started asking me how difficult it was to be one. In fact, I don't think being a woman has ever been an obstacle to my professional advancement. I know that it has been for other women, though, and I'm certainly interested in removing whatever obstacles they may face. The gender imbalance in computer science students does bother me, though. The percentage of women among our graduating CS concentrators has varied between about 5 and 15 percent since I've been here. I'm not sure why it is so low, but I know we're missing out on some great computer scientists.

Part of the reason I'm giving this lecture is to try to dispell some of computer scientists' geekly image. I have to admit that there are a few geeky computer scientists; but they aren't the best ones, or even necessarily good ones. You can have a life and do CS. And it's fun!

In addition to my faculty position, I now have a new job that affords time for reflection. My first child, Elizabeth, arrived in January. Raising children may be something like farming: I mentally composed a lot of this talk while feeding Elizabeth at 4:00 am. And I take her for long walks, telling her about my research ideas as we go. She always falls asleep, but I'm grateful for the company.



l to r: Christopher Cantor & Brett Levine, both '97, work on their robot

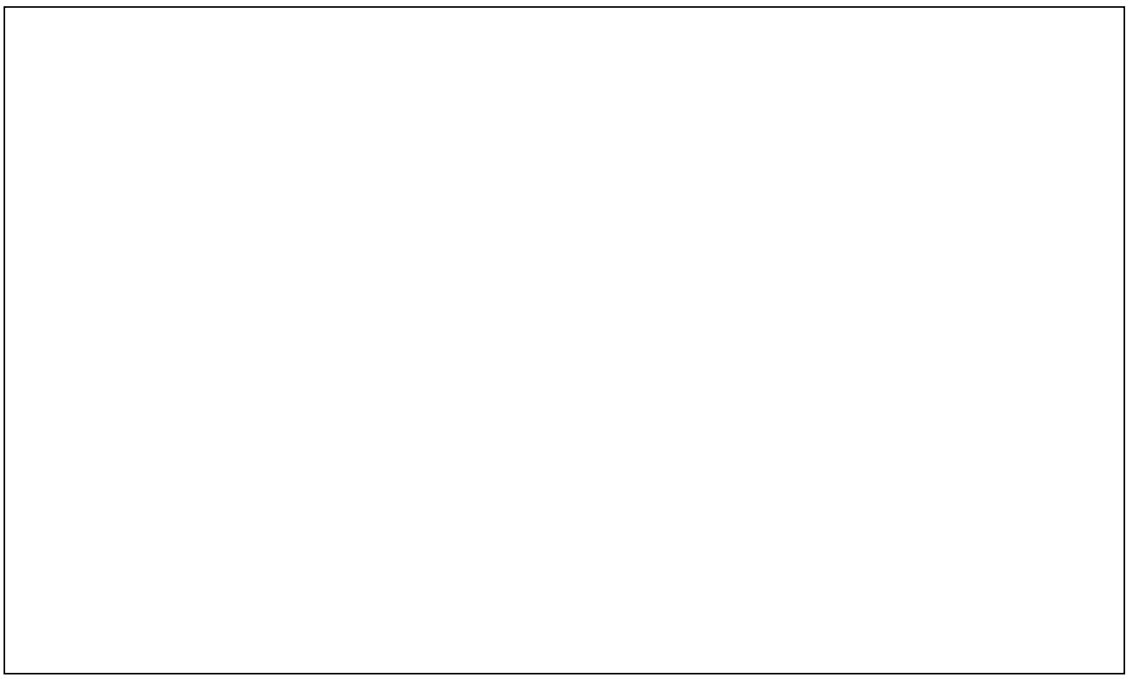
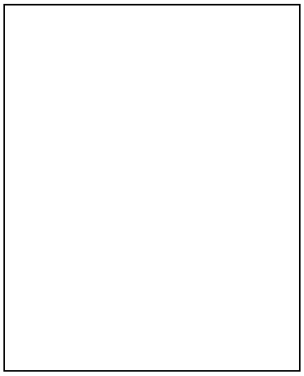


Fig. 1: A directed graph built from a program fragment. The vertices represent procedures and the arcs represent direct invocation

A SOLUTION IN SEARCH OF A PROBLEM



Philip Klein

I start with an illustrative (but not realistic) example. Imagine that you are trying to debug a huge program and need to know whether invoking procedure A could eventually result in the invocation of procedure B.

This question can be formulated in terms of a graph representation of the program. The graph has a vertex for each procedure, and an arc X —>Y for each pair of procedures X and Y such that X directly invokes Y. Figure 1, provided by Steve Reiss's FIELD environment, depicts such a graph for a relatively small program.

The question of whether invoking procedure A could result in the invocation of procedure B is, in graph terminology, the question of whether the graph has a path from vertex A to vertex B, i.e., whether A can *reach* B. There are efficient algorithms for searching a graph that can be applied to answer this question. Such an algorithm, however, finds *all* the vertices reachable from the given vertex A, and the time required is roughly proportional to the number of such vertices. Therein lies the disadvantage of this approach: the graph-searching algorithms give us much more information than we needed and exact a proportionately high price in terms of time. All we wanted to know was whether A can reach B, and we ended up learning about all the nodes A can reach.

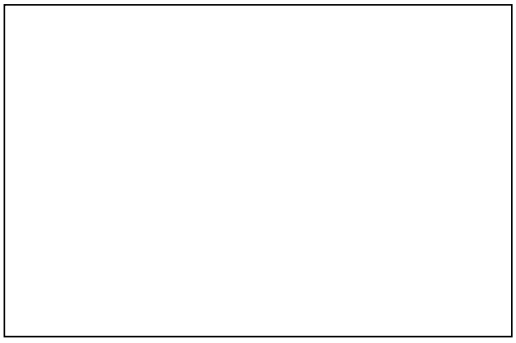
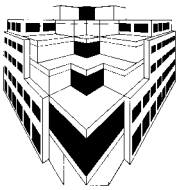


Fig. 2: A simple directed graph and its transitive closure. Vertex A can reach vertices C, F and G; vertex B can reach all vertices.

Is there a shortcut? One can, for example, stop the search early: once one determines A can reach B, one need not continue searching the graph for more vertices reachable from A. However, this trick is not likely to save us much time, especially when B is in fact not reachable from A.

We are led, therefore, to consider *preprocessing*. The *transitive closure* of an *n*-vertex graph is an *n* x *n* table. There is an entry for each pair of nodes; the (X,Y) entry is 1 or 0 depending on whether X can reach Y or not. Thus this table contains answers to all possible reachability queries about the graph.



One can construct the table during a preprocessing step. Once it has been constructed, it is easy to determine whether A can reach B: simply look up the (A,B) entry on the table (below). This takes constant time.

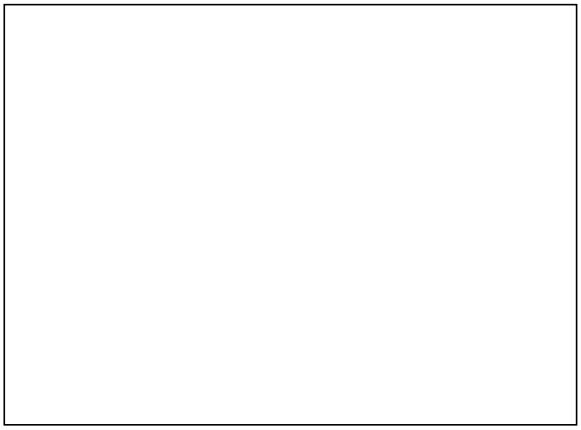


Fig. 3: The transitive closure for the graph depicted in Fig. 2

This cure, however, is worse than the disease —at least if the graph is large enough. For a huge graph, one with, say, ten million nodes, it is infeasible to store the entire transitive closure in primary memory—it is just too big. Storing it in secondary memory would of course defeat the purpose, since access to it would be too slow.

My former student Sairam Subramanian (now with Nortel) and I came across a technique to address this problem (we were really working on something different, but that's another story). We found a way to represent the transitive closure that is much more compact than the straightforward representation. Querying the representation—i.e., determining whether a vertex A can reach another vertex B—is more than simply looking up a single table entry, but is much less time-consuming than searching the entire graph.

The representation technique can also be used for approximate shortest-path distances in a graph with edge lengths. That is, one can construct a compact representation with which one can quickly answer queries of the form: what is the distance from A to B? Here again, one can answer such questions by searching the graph or anticipate them by precomputing a table giving, for each pair of nodes X and Y, the shortest-path distance from X to Y.

The first solution requires lots of time per query and the second solution requires lots of space to store the table. Our technique provides a representation that requires only moderate storage space and allows moderately fast query-answering.

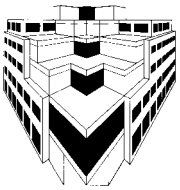
I'm not going to go into the technical details of our representation technique; rather, I'd like to ask readers to suggest uses for this technique. The example with which I started this article, while illustrative, is not really appropriate for two reasons: (1) It's not clear why one would want to know, for particular procedures A and B, whether invoking A results in the invocation of B. (2) More importantly, the nature of the application is such that extremely quick answers are not necessary—a tenth of a second, for example, is fast enough for interactive use. Even a straightforward graph search operating on a pretty large graph could answer a reachability question within that time.

Here are several applications that are slightly more realistic, from the areas of databases, artificial intelligence, and information retrieval. Previous work by H. V. Jagadish and others on compact representation of transitive closure has addressed the first two.

Databases: One way the relational database model has been extended is by the addition of recursive queries (e.g. in Zloof's QBE). A basic tool in recursive databases is the transitive closure of certain relations. In order to handle multiple queries concerning the same relation, one might want to precompute a compact representation of the transitive closure so as to answer these queries quickly.

The boundary between artificial intelligence and databases: In knowledge-representation languages (such as KL-ONE) and structural data models (e.g. CANDIDE and CLASSIC), queries and inferences involve frequent reachability queries in a graph representing the inheritance relation. One would like to be able to represent the transitive closure compactly so as to speed the processing of such queries and inferences.

Fig.4: A simple Is-a hierarchy (from Charniak and McDermott's book on AI)

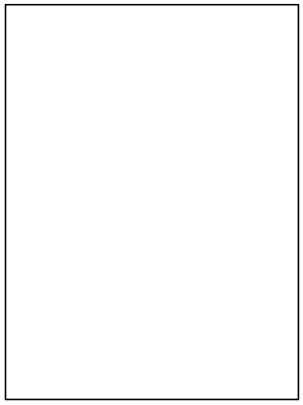


Information retrieval: This final application is somewhat vaguer and more speculative in nature. Imagine a graph in which the vertices represent concepts. For some pairs of concepts, there is an edge representing the relatedness of the concepts whose length indicates how closely related the concepts are. Thus the shortest-path distance between two concepts not directly linked by an edge is an indication of how closely related they are. In organizing and categorizing documents, one might need to make many pairwise comparisons of concepts. Such comparisons would be facilitated by our compact representation of shortest-path distances.

Can readers suggest possible applications for our representation technique? Features of an appropriate application are:

- There is a huge graph in which the number of edges is only four or five times the number of nodes.
- The application specifically needs to perform *pairwise* reachability or shortest-path queries, queries that involve just two nodes A and B. (Our technique is not needed for finding all nodes reachable from A or all nodes reachable within a given distance.)
- The application requires very quick answers to such queries; e.g. hundreds of queries must be handled every second.

SPOCS LIVE LONG & PROSPER



Detail from the SPOC Hall of Fame whiteboard

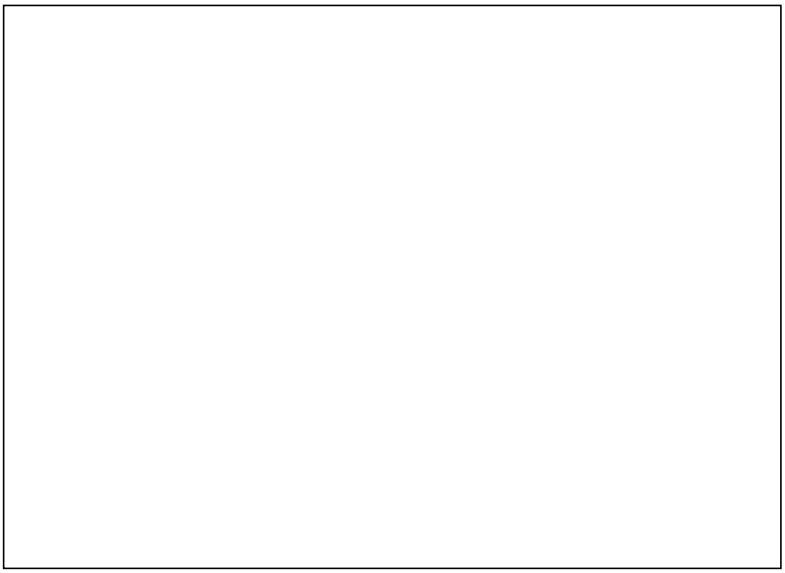
To land a SPOC (Systems Programmer Operator and Consultant) position in CS is to hold the most prestigious, most coveted undergraduate job in the Department. The SPOC tradition is mutually beneficial for students and for the technical staff. Students learn systems programming and management, gain sophisticated hands-on experience, and earn some money too, while tstaff appreciate the remarkable amount of work SPOCs accomplish. Not only do tstaff enjoy working with these exceptionally talented individuals (too expensive to hire once they graduate), they also learn from them. SPOCs say the enjoyment is mutual.

Having survived a rigorous selection process that includes a somewhat intimidating interview by six tstaffers, the current SPOCs, the meta-Head-TA and the Head Consultant all at once, the new SPOC launches into the exacting world of systems administration, writing and maintaining software at a variety of levels, answering questions and resolving problems and crises for students, faculty and staff. Their expertise is available to CSers during tstaff's off-hours, but since they each have their own machine and the luxury of a SPOC office (a perk not likely to be found on the outside), they are generally in residence most of their student lives. Consequently, the SPOC/tstaff relationship is a close one and long-lasting associations often result. Tom Lawrence (SPOC '89-'90) feels that "....the best part of being a SPOC is working with the tstaff. They are just the greatest. Every time I've gone back to visit Brown (alas, very rare these days), the 5th floor of the CIT has been my first stop. I dread the day when none of the old guard are there any more."

Together with the legendary prestige of SPOC-hood (to undergraduates they are computer gods with root access and wizardly knowledge) comes serious commitment and sheer hard work. Says Tom Lawrence, "The hardest I ever worked as a SPOC was the day I installed all 63 new Sunlab SPARCstation1s in a single marathon nine-hour thrash. I had two or three master disk drives with the OS installed on them. I had to unpack and assemble each machine, install a master disk, boot single user and do a disk-to-disk copy, change the hostname of the machine on the new root disk and reboot." One summer, Tom and John Kraft ('89-'91) put in a 36-hour stint. They were given to cranking up their stereo (serious industrial music) to perform their weekly 2:00 am backups; it would shake the entire half of the building.

"The hardest I ever worked as a SPOC was the day I installed all 63 new Sunlab SPARCstation1s in a single marathon nine-hour thrash"

Steve Drucker ’83-’84 was the original Meta Technical TA/SPOC. SPOC legends include Adam Buchsbaum (’85-’89, coiner of the SPOC acronym) and Mary Fernandez (’86-’88), who worked as SPOCs in ’86-’87; they went on to graduate school at Princeton, married and have produced two SPOClets, Erena and Shira, whom Jeff Coady has declared SPOC legacies. Artistic Mike Galvin ’88-’89, an avid Trekkie, started the tradition of SPOC caricatures on the machine room white boards; they are preserved there in perpetuity. Mike went to work for EBT (now part of Inso Corporation) where he co-founded a rock band ‘Q’ (“It’s difficult to work in a group when you’re omnipotent”). Tom Lawrence recalls Mike’s naming the SPOC office node (a Sun 3/60) ncc1701e after the then-fictional next USS Enterprise. “It should be noted that with the release of Star Trek, First Contact, this ship now exists on film! Mike was always way ahead of his time and still, to this day, is the only person I have ever heard tell a successful stack-backtrace joke.” Luther Kitahata, ’84-’86, was a “regular hang-out guy” whose Topsiders were held together



The SPOC Hall of Fame whiteboard

with duct tape; much to everyone’s astonishment, when he went to SGI he was soon voted best dressed! Tom Lawrence remembers doing his first backup with SPOC Lee Mighdoll (’88-’89) on call. Lee, who sported a bushy beard of noble proportions, was in the process of shaving it off that evening when he was summoned to help with the backup. He arrived with only half the job done! His half-bearded image was added to the white board next to the drawing of Mike Galvin with Spock ears.

There are several versions of the now mythical story about Zoran Popovic (’89-’91) and the spilled milk, but the olfactory upshot is always the same—after several weeks in the refrigerator, Zoran’s gallon of milk finally exploded, spewing its curdled contents all over the carpet. Doubtless, to the other SPOCS in office at the time, the memory still wrinkles noses; but memories of food can be pleasant too: Dave Durfee, the original manager of the CS technical staff, remembers working with Max Salvas on the Apollos in Foxboro Auditorium during the Department’s early days; in order to sustain them on the job late into the evening, Andy van Dam would ply them with sweet and sour soup—hand-delivered at 1:00 am!

Ron Palmon, ’93-’95, was a pre-med student majoring in CS during his stint as a SPOC. He is now in medical school at NYU interested in primary care. Mike Shapiro, ’94-’96, is now tstaff Systems Programmer but will be leaving for Sun in August to join the Solaris performance group. Amy ‘Tiny SPOC’ Flynn (’93-’94) came by her nickname via an Apple Newton which Systems Manager Peter Galvin had won. The Newton read Amy’s handwritten name as ‘tiny’ and it stuck—fortunately, Amy is petite. She is currently working in London with Adam Stauffer (’91-’93, he of the Eddie Bauer wardrobe) at D.E. Shaw, an investment company.

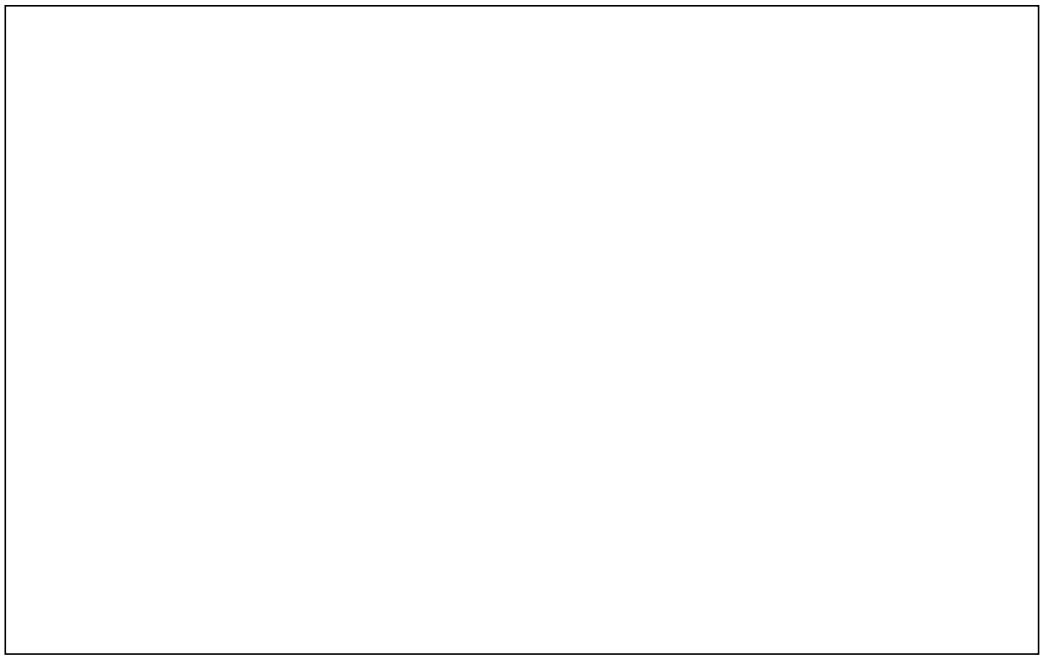
It is the vigilant SPOC’s duty to report wrongdoing and breaches of security. On one occasion a SPOC sent email to Director of Computer Facilities Jeff Coady that a student was illegally accessing a systems lab. Since such an infraction is a disciplinary matter, chairman Eugene Charniak (ec) was copied on the email Jeff sent to reprimand the violator. The student, hoping to elicit empathy from a perceived fellow sinner, emailed ec— “...were you caught in Syslab A, too? ....this Coady guy is a real jerk...”! While the value of the SPOC program is unquestioned, anomalies in the space-time continuum have been known. Peter Galvin, former Systems Manager, remembers when a SPOC wondered what the key in the door of the server did—he turned it and powered off the machine; another time a SPOC aborted the entire nightly system backup to allow a professor to finish some work. The SPOC interview now includes a question about the latter scenario, the better to weed out the meek.

SPOCs are usually hired as juniors prepared to commit two years to the job. They begin with more routine tasks but really come into their own their second year. Aside from the work they perform, the SPOCs also remind us that the University exists for students, a fact easy to lose sight of when handling computing facilities. As substitutes for tstaff during off-hours, a tremendous amount of trust is bestowed upon them—they know the root password and also have access to a master key (the chairman himself has only a master key, not the password!). The compute-power available to SPOCS is enormous—Tom Lawrence once used all 180 machines to create a Mandelbrot image. The entire contingent of machines was also used to run a cracker program that checks for breakable password entries. Because of the success of the SPOC program, John Bazik (Manager, Research and Educational Software) was able to create a student programmer job, Systems Programmer (fortunately, all attempts at Trek acronyms failed).

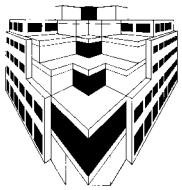
One project that began as a SPOC project and ended up a Department-wide boon was Quahog. According to Bazik, "Quahog *(http://www.cs.brown.edu/software/quahog/)*, a polite distributed processing system, runs jobs on otherwise idle workstations, ensuring that the jobs stop if someone sits down to work. Quahog has enabled researchers in the Department to queue up and run thousands, even tens of thousands of jobs on machines throughout the department without disturbing anyone. Several researchers, including Chairman Eugene Charniak, have used Quahog extensively, getting results in a fraction of the time it would have taken on one or a handful of machines. The original spec was done by Boris Putanec ('90-'93), John Kraft and Zoran Popovic, although fulltime tstaffers brought it to fruition."

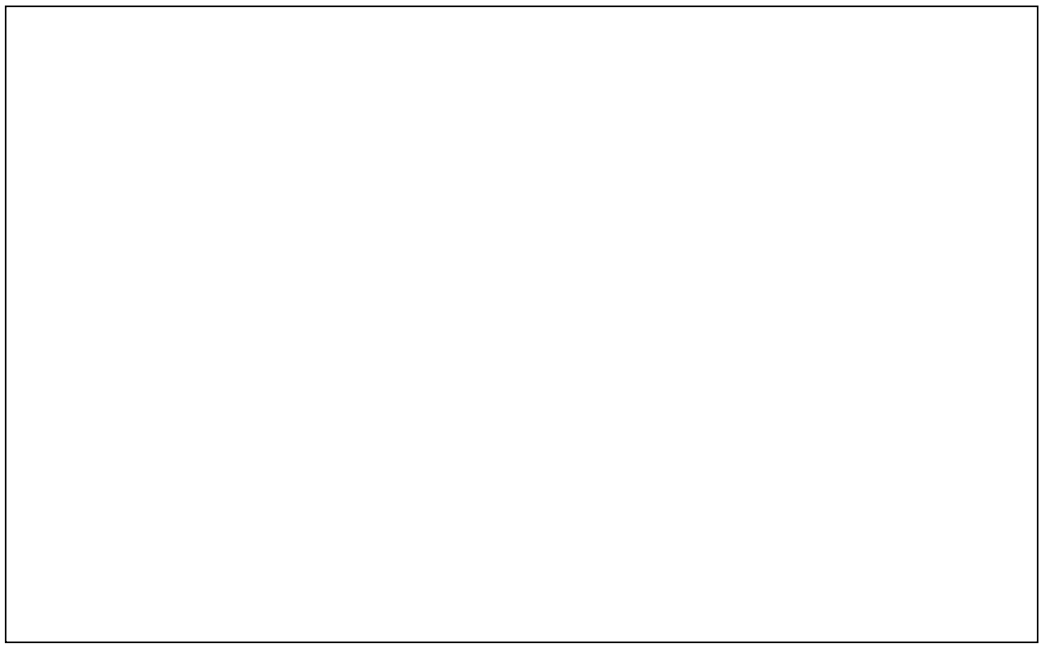
Tstaff often receive kudos from departed alumni, faculty and visitors who miss the way the Department is run. The SPOCs are an essential part of this; as John Bazik points out, "tstaff don't wear beepers." The complexity of their tasks and the responsibility of their positions are sufficient to count as "real-world" experience—SPOCs have little trouble finding employment after graduation. Says John Kraft, "...being a SPOC certainly helped me acquire a job here at Silicon Graphics."

Qapla'



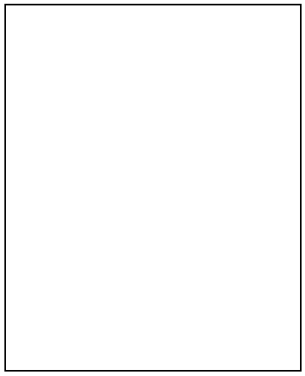
The current cadre of SPOCs: l to r: Stephanie Schaaf, Mike Shapiro, James Todd, Jesse St. Laurent and Dan Price





The February IPIP/SCOOP technical job fair was held in Sayles Hall rather than in the Department, the better to accommodate the 47 companies and over 450 students who participated

FIRST KANELLAKIS AWARD PRESENTED AT ACM '97



Peter Wegner

The first Kanellakis award was given at the annual ACM conference in San Jose on Sunday, March 2nd, for public-key cryptography. The award ceremony, which also included the presentation of the Turing award and the Karl Karlstrom education award, the installation of ACM Fellows, and the announcement of winners of the ACM programming contest, was memorable. Our thoughts and hearts went out to the Kanellakis family, and they would have been proud of what was said of Paris on this occasion.

The citation for this award read as follows:

"The first ACM Kanellakis Award for Theory and Practice is awarded to Leonard Adleman, Whitfield Diffie, Martin Hellman, Ralph Merkle, Ronald Rivest, and Adi Shamir, for the conception and first effective realization of public-key cryptography. The idea of a public-key cryptosystem was a major conceptual breakthrough that continues to stimulate research to this day, and without it today's rapid growth of electronic commerce would have been impossible."

The idea of an award recognizing the impact of theory on practice has been well received by the computing community. Each of the nominations received by the Awards Committee thoughtfully addressed what it means for a theoretical contribution to have an impact on practice, and included letters from both theorists and practitioners attesting to the value of the contribution. Indeed, as hoped, the existence of this award seems to be causing researchers to place greater value on building bridges between theory and practice. We were pleased with the high quality of this year's nominations, and they will be a part of the pool from which awards in future years will be chosen.

The endowment for the award includes a generous contribution from the Kanellakis family, support from PWS Publishers, and contributions from the ACM Special Interest Groups for computing theory, databases, and programming languages. We have reached $80,000 of the $100,000 needed to make this award every year. Our request for individual contributions, announced in a previous issue of ***conduit!***, was delayed and we therefore repeat our invitation at this time. We invite you to contact *shp@cs.brown.edu* (Susan Platt) to specify the amount you wish to pledge. The ACM will then invite you to send in your contribution, and will send you an acknowledgment attesting to its tax-exempt status.

EMAIL TO THE EDITOR *et al*

MARC ABRAMSON AB '95.5

Just wanted to say hello, Andy. I was reading the *New York Times* about a week ago, an article about Vartan, when I came across a quote of yours. It made me realize that I had not been in touch with you for some time. I am interested to hear how CS15 was this past semester, especially since I heard you made the switch to Java. I have been working with IBM, for what used to be called ISSC, now IBM Global Services. It is funny too because the client I have been assigned to has me programming in Object-Oriented Pascal. I also thought you might like to know that on many more than one occasion I have been thankful for the great education I got in the Computer Science Department. I was more than prepared to conquer what the real world had to throw at me, and I feel comfortable enough with my foundation to jump at opportunities to learn new things.

marc_abramson@vnet.ibm.com

ERIC ALBERT AB '80

Eugene, it was a pleasant surprise, while reading the latest issue of ***conduit!***, to come across the writeup about the *New York Times* crossword story that mentioned me. It was an even greater pleasure to have you remember that I took your AI class. Not only did I take it, but I and my classmates Neil McBurnett and David Auty (who were also my apartment-mates) had you over to our campus apartment for dinner one evening. I thought it was gracious and good-humored of you to accept our invitation. It meant a lot to me then, and it means a lot to me now. And speaking of good humor, I'm glad to see that you have retained yours—your musings in ***conduit!*** are funny enough that I often read them aloud to Rose, and she has no CS background whatsoever. P.S. I'm more in the entertain-ment business than you know—I have the leadoff story in the anthology *Best American Erotica 1996*, edited by Susie Bright, published by Simon & Schuster, available at large bookstores everywhere. Feel free to note this in ***conduit!***, since writing the story required a 133 Mhz Pentium clone with 64 MB RAM and two 1.2 GB disks running Microsoft Word 6.0a.

ealbert@world.std.com

Eugene's response: Eric, I certainly do remember going to your apartment for dinner. I also remember that I brought a bottle of wine and none of you folks drank it (or any other alcohol). In retrospect this fact strikes me as even more unusual, given the recent spate of alcohol problems on campus. Also, thanks for the nice words about my column. My 14-year-old son complains about my bad jokes. I will show him your letter in a no doubt vain effort to convince him that he just has no sense of humor.

STEVE EHRLICH AB, ScB '77

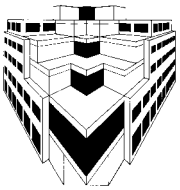
Dear Professor van Dam: From 1975 through 1977 I was one of the many students who took your CS classes. To get an exact date, I started your AM101 class and by the end of the semester, it was renamed CS101. At the start of AM101, you told us that you "expected excellence from us because when we hit the real world and people heard we had you as a teacher, you did not want us to embarrass you."

Now, about 20 years later, I want to thank you for pushing us to excellence, for not accepting anything but perfect work. I continued my education after Brown, getting an MS in Applied Statistics from Purdue University and an MSE in Computer and Information Sciences from the University of Pennsylvania. But it was your teaching and pushing for excellence that taught me the analytical and programming style which has made me successful.

Now, I was just one of the many faces in the crowd to you. I was never a "grader" or one of your mentored few. In fact, if you had passed me on campus back then there is only a 20 percent chance you would recognize me, and I sat in the front of your classes. I am now Director of Technology in the Information Services Department for Mercy Health System, a Philadelphia-based hospital and health care corporation.

As one of the handful of professors who most influenced me, every day I wonder if my work would meet your standards. I strive never to embarrass you and owe a big thanks for what you taught me.

smetch@waterw.com



ERIC GOLIN ScB '81

Trina, Hi—nice to hear from an old familiar voice. I left Illinois in '93 to pursue more applied (and lucrative) interests. I'm on my second Silicon Valley startup, a company called BroadVision that makes application systems software for the Internet. My title is Director of Software Development—basically I am the engineering director responsible for the software end of product development. On a personal note, things are very well with me. My wife (Marion Abrams, ScB '81 like me) and I have slowly grown accustomed to the sunny weather, plethora of activities and outrageous housing prices. We live in Menlo Park, a short walk from downtown Palo Alto. We have two kids (James, 5-1/2, Anna, 1-1/2).

egolin@broadvision.com

BRIAN KNEP ScB '90, MS '92

Andy, just want to let you know some great news: I got two Scientific and Technical Academy Awards. One is a technical achievement award for the Direct Input Device (formerly known as the Dinosaur Input Device). The other is a scientific and engineering award for a 3D paint system. Details can be found at *http://www.oscars.org/pressreleases/97.01.06-html*. It's all very exciting for me, and I owe some of this to you and Spike and my time in the graphics group—thank you!

bk@nearlife.com

John Hughes (Spike) forwarded the same good news about Brian from the Academy Awards home page; he remembered that an initial version of the 3D paint system was part of Brian's CS224 final project!

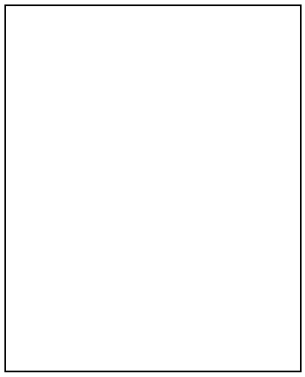
JAMES CIMINO ScB Biology '77

Andy, over the years, I have somehow abandoned the tradition of sending you a program for the annual computers-in-medicine meeting. I thought I would resurrect the tradition one last time since this year it's *my* meeting (program chair). I can trace my success with this project and many others (I am now tenured in medicine and medical informatics) directly back to the education you provided me at Brown. Thanks!

james.cimino@columbia.edu

ERRATUM for last fall's issue: Dilip D'Souza received his MSc in '84, not '88.

THE 17TH IPP SYMPOSIUM



Leslie Pack Kaelbling

On October 24, 1996, the Department hosted its 18th Industrial Partners Program technical symposium, focusing on the emerging field of data mining and its connections to machine learning.

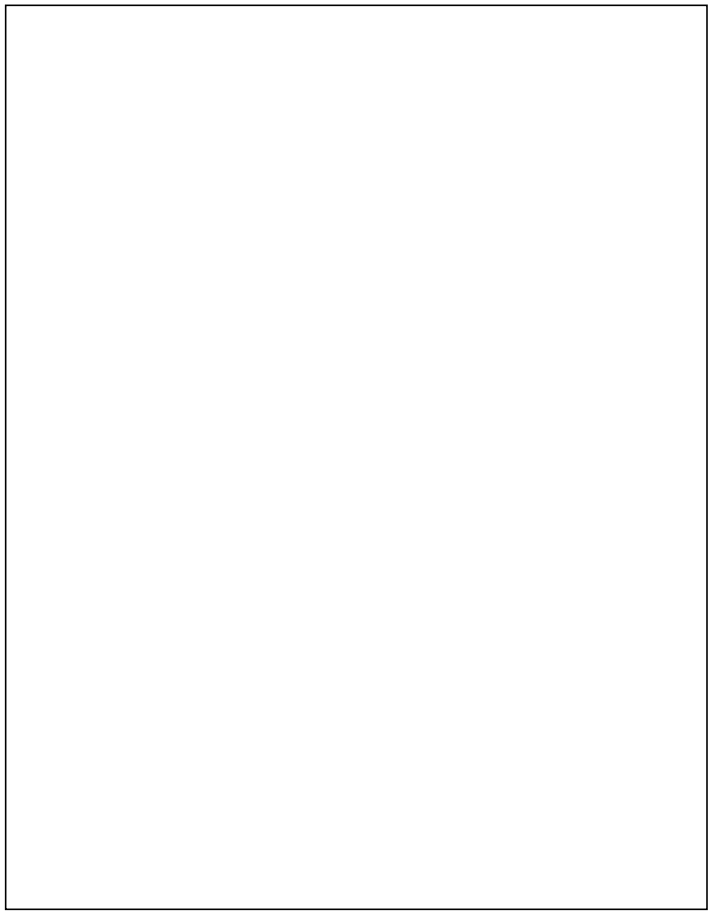
Most areas of industrial practice have seen a huge growth in the information collected about product sales and performance, manufacturing quality and efficiency, and marketing effectiveness. The vast majority of this data sits unanalyzed because no one knows quite how to tackle it. Data mining and machine learning address the problem of discovering useful knowledge from very large databases. The symposium featured five industrial researchers working at the forefront of this new area, which will have broad practical impact throughout industry.

Usama Fayyad of Microsoft Research, a pioneer in data mining and knowledge discovery in databases (KDD), was to have given the first talk, an overview of the area. Unfortunately, he was suddenly unable to attend, so Leslie Kaelbling gave an impromptu talk with his slides. It emphasized the KDD process as an interactive one with a human in the loop: automated data-mining techniques can help a human understand a large data set, but the human must participate in the process by doing feature selection and evaluation of the knowledge extracted by data mining. The talk concluded by decribing an extremely successful application, done by Usama at JPL, that learned to categorize astro-nomical objects (stars, galaxies, etc.) from very blurry data; its ultimate performance was considerably better than human performance.

The next talk, by Ronny Kohavi of Silicon Graphics, described Silicon Graphics' data-mining tool, MineSet. MineSet includes both interactive data-visualization techniques and analytical data-mining techniques. The visual techniques include: interactive scatterplots of multidimensional data and mapping attributes to visual attributes, data shown on maps (sales by region/state/zipcode), hierarchical breakdowns of information (sales by product area, product group, UPC) and interactive fly-throughs over the data. The analytical techniques include decision trees and simple Bayesian networks (evidence charts), both

tightly coupled with appropriate visualizations to help users understand the classifiers built. During the lunch break, Ronny gave demos of MineSet.

After lunch, Foster Provost of NYNEX Science and Technology gave a talk on automated design of fraud detection systems. In addition to teaching us about data mining and machine learning, Foster taught us a lot about cellular-phone fraud! Fraud detection is based on profiling customer behavior and checking for anomalies. The domain of Foster's case study was cloning fraud in cellular telephony, but his methods are applicable to any domain



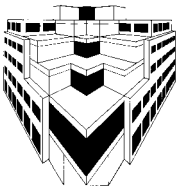
Symposium speakers from l to r: Foster Provost, NYNEX Science & Technology; Kamakshi Lakshminarayan, Honeywell; Ronny Kohavi, SGI; Leslie Pack Kaelbling, host; Yoram Singer, AT&T Research

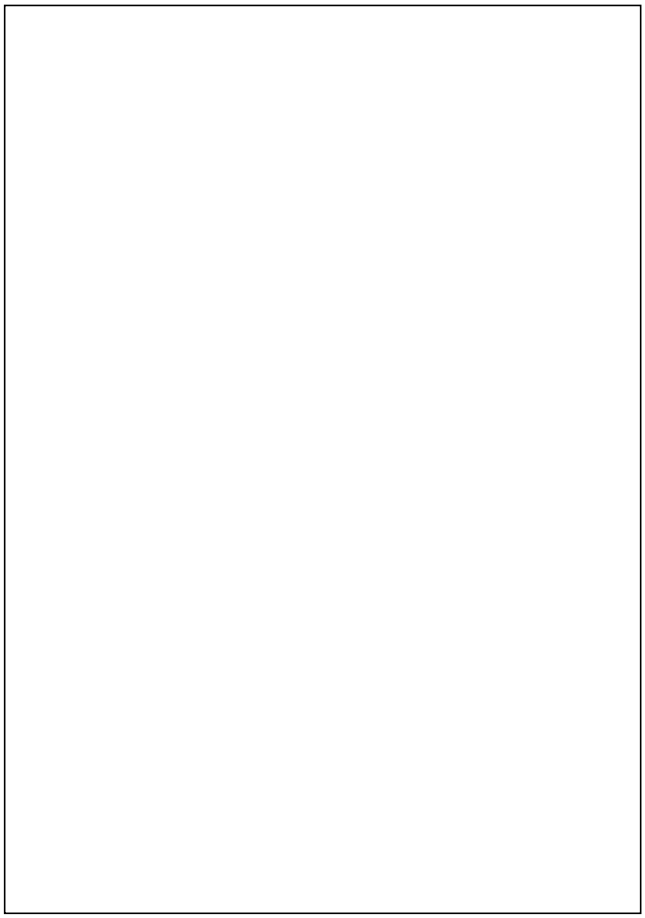
in which fraudulent usage is superimposed upon legitimate usage (as in credit card fraud). A rule-learning program is used to uncover indicators of fraudulent behavior from a large database of cellular calls. These indicators are used to construct profilers that then serve as features to a system that combines evidence from multiple profilers to generate high-confidence alarms. In a fielded system, the alarms would either trigger further investigation by a human or cause a hold to be put on an account.

Next, Kamakshi Lakshminarayan of Honeywell Technology Center spoke about performing data mining in databases with many missing or erroneous values. She presented a case study of data entered by field service personnel in which nearly half the values were missing, with no single complete record. Researchers at Honeywell (including S.A. Harp, R. Goldman, and T. Samad) approached the data-completion problem using two well-known machine-learning techniques. The first is an unsupervised clustering strategy that uses a Bayesian approach to cluster the data into classes; the classes so obtained are then used to predict multiple choices for the attribute of interest. The second technique involves modeling missing variables by supervised induction of a decision-tree-based classifier so as to predict the most likely value for the attribute of interest. Tests on the industrial database showed that both approaches are useful and have advantages and disadvantages that make them appropriate for different contexts.

The last talk of the day was given by Yoram Singer of AT&T Research, on machine-learning approaches in natural-language processing. Although humans are experts at natural-language processing, they typically cannot describe their abilities in such a way as to hand-craft automatic natural language systems. Another alternative is to develop computational models and algorithms for automatic machine learning from past experience. There have been several recent successes by machine-learning researchers in applying ideas from machine learning to natural-language processing, notably in deterministic and probabilistic automata learning, online prediction, and automatic rule induction. Yoram described a few such successes, based on new machine-learning paradigms developed at AT&T Labs.

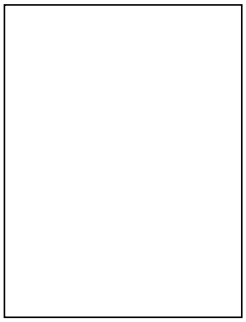
The symposium concluded with a panel discussion on the general applicability and utility of machine-learning and data-mining methods. It was concluded that these techniques are ready to be used outside the research lab and that, although a great deal of development is still to be done, they will provide an exciting new technological tool.





Fragments of the view from the 5th-floor deck remain as the W. Duncan MacMillan science teaching facility takes shape

50 YEARS OF COMPUTING



Peter Wegner

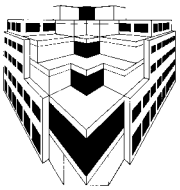
*Computing Surveys**, an ACM journal, commemorated the 50th anniversary year of the Association for Computing Machinery (ACM) and of the discipline of computing with two special issues: the March 1996 issue on Perspectives in Computing that examines current technology, and the December 1996 issue on Strategic Directions in Computing that looks towards the future. Brown faculty participated in both these projects. The issue on Perspectives on Computing included articles by Roberto Tamassia on data structures, Tom Dean on automated planning, Tom Doeppner on distributed file systems and distributed memory, Steve Reiss on software tools and environments, and Peter Wegner on interoperability. Brown faculty chaired or co-chaired four of the 19 working groups on strategic directions—on artificial intelligence (Tom Dean), computational geometry (Roberto Tamassia), constraints (Pascal Van Hentenryk), and databases (Stan Zdonik)—and participated in a number of other areas, such as theory of computing (John Savage).

The "Perspectives" and "Strategic Directions" issues respectively classify the field from the viewpoint of content and research directions. The "Perspectives" issue contains 68 short articles classified into the ten topic areas listed below, including the nontraditional areas of computational science, databases, human-computer interaction, and graphics (the number of articles in each area is given in parentheses):

- Algorithms and data structures (10)
- Architecture (2)
- Artificial intelligence and robotics (6)
- Computational science (3)
- Database and information retrieval (10)
- Graphics (6)
- Human-computer interaction (8)
- Operating systems and networks (11)
- Programming languages (7)
- Software engineering (5)

These subdivisions of the discipline reflect a broadening of the self-image of computing that is closely related to the most recent ACM/IEEE curriculum (Curriculum '91). The profound technological changes from mainframes to personal computers and networks will no doubt cause further changes of balance among subdisciplines. However, the above categories provide a contemporary classification of computing in the mid- to late-1990s whose substance is clearly presented in individual articles. Longer versions of the papers in this issue appear in the 2500-page *Handbook of Computer Science and Engineering*, published by CRC Press in December, 1996.

*Peter Wegner is the Editor-in-Chief of *Computing Surveys*



The "Strategic Directions" issue was based on a workshop held at MIT last June that grew from its original size of 100 researchers in five areas to include 300 researchers in 22 areas. The working-group reports developed at the workshop were refined over several months and the 19 published contributions were classified into subareas that overlap with the ten 'perspectives' areas but view the discipline in a different though complementary manner.

Foundations:

- Theory of computing
- Computational geometry
- Concurrency
- Formal methods
- Programming languages
- Artificial intelligence

> *"the field is moving increasingly towards applications as it matures, suggesting there may be a mismatch between the self-image of researchers and the expectations of funding agencies and society at large"*

Systems:

- Computer architecture
- Networks and telecommunications
- Object-oriented programming
- Constraint programming
- Software engineering and programming languages
- Software quality
- Real-time and embedded systems
- Database systems
- Storage I/O issues in large-scale computation

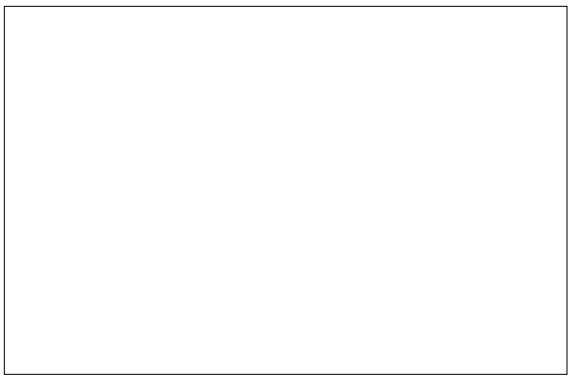
Applications and infrastructure

- Human-computer interaction
- Computational science and engineering
- Electronic commerce and digital libraries
- Computer science education

These strategic directions reflect both traditional research and emerging areas of application-oriented research. Much thought went into this three-way classification. Several working groups initially classified as application areas requested reclassification as foundations or system areas, perhaps because researchers prefer to be seen as working on foundations or systems rather than on applications. However, the field is moving increasingly towards applications as it matures, suggesting there may be a mismatch between the self-image of researchers and the expectations of funding agencies and society at large. This accords with the observations of Thomas Kuhn in *The Structure of Scientific Revolutions* that researchers tend to resist rapid disciplinary change. The pressure to become less concerned with inner foundations and more outward-looking is evident in ongoing debates both within and outside the computing community.

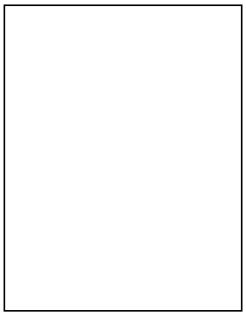
The working-group reports provide an excellent analysis of strategic directions in major subfields of computing, though they do not completely cover every topic. They are supplemented by about 200 electronically accessible individual position statements that were initially prepared for the June workshop and were reviewed and refined for publication. More than 650 pages of online position statements are available through *http://www.acm.org/surveys*. This experiment in joint hard-copy/online publication makes it economically feasible to publish the equivalent of over 1000 pages as a single issue of *Computing Surveys*. The ACM's electronic repository in *www.acm.org* will increasingly supplement its hard-copy publication, and this *Surveys* issue is one of several ACM experiments in extending user services electronically.

Though there is considerable overlap in subject areas between the "Perspectives" and "Strategic Directions" issues, the goals and levels of presentation are complementary. Browsing for a few hours through these two issues is an excellent way of catching up with current trends and future directions in the computing field. While browsing, look out for contributions by Brown faculty to this comprehensive two-pronged attempt to take stock of current technology and understand future strategic directions.



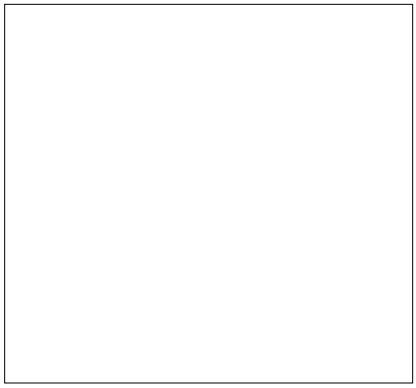
Cogitations on the atrium whiteboard—Danah Beard is a member of the graphics team

FROM THE CHAIRMAN, Eugene Charniak



Eugene Charniak

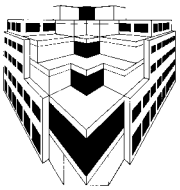
When the architects were designing our building, one of the things we asked for was a common area that would draw people to it, as opposed to the usual common room stuck off in a corner and unused because it is out of everyone's way. Their solution was the delightful atrium area at the center of the department—an area on everyone's way, and one getting a lot of use. We realized how much use when Trina Avery pointed out how the material on the atrium whiteboard changes almost daily. This made us actually pay attention to the material put there by students illustrating technical points in their discussions. A couple of examples are shown above.



Undoubtedly the goriest lecture of the year was given by Dr. Arnold-Peter Weiss of the Medical School, an expert on carpal-tunnel syndrome. (He also operated on Andy van Dam for this condition.) He gave lots of useful tips on reducing the likelihood that the problem will hit you. However, what probably impressed the audience most was his slides, including many of patients with their arms slit open to show the condition, and the surgical methods used to repair the problem. I noticed that quite a few of the audience turned away. Of course, you may well ask, why was I looking at the audience and not at the slides? (Actually, ever since my anatomy and physiology course in college, I am somewhat inured to such pictures, though without constant repetition my immunity is wearing off.) I conjecture that the pictures were put in there deliberately to make us take the warnings seriously. Too many of us tend to think that this will only happen to someone else. (In retrospect, though, I wonder if any of those hands belonged to Andy.)

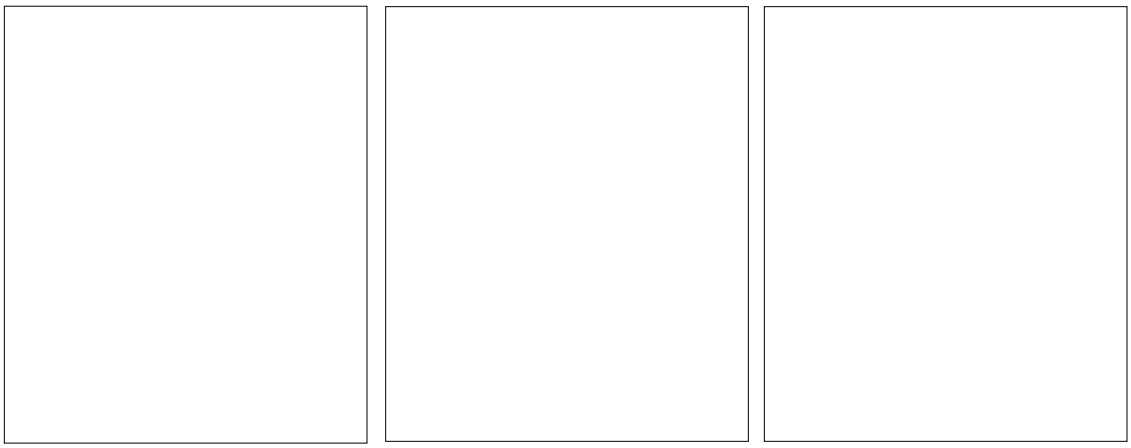
The major conclusion of this lecture was that stopping typing for at least five minutes every hour is the simplest and most effective prophylactic measure one can take. Thus we now set up all new accounts with a timer program that notes typing times on the machine and lets the user know when he/she has been typing for an hour. The program does this by putting up some graphics of hands and fingers encouraging the user to rest them or give them some (non-typing) exercise. One of the graphics is shown on this page. The program also has a "–rude" option that shows the fingers in a slightly different configuration!

Two graduate students have successfuly defended their theses since the fall. **Dina Goldin**, whose topic was "Constraint Query Algebras," is now teaching at UMass Boston. **Dzung Tien Hoang** returned to the Department to deliver his defense from Duke, where he has been completing his Ph.D. with Jeff Vitter. His topic was "Fast and Efficient Algorithms for Text and Video Compression." Dzung is doing research and development of algorithms for



video compression, specifically for the MPEG-2 standards for applications such as DVD disk and networked video. He is working for Digital Video Systems.

As I have remarked before in the column, one of the fun parts of being chair is the responsibility of talking annually with all our faculty members in order to report to the university how they have been doing. (Writing the report, however, is not fun.) My favorite academic story this year is one Roberto Tamassia told me. As a member of the new Center for Geometric Computing, Roberto has been working on architectures for web-based algorithm animation. When he was showing off his wares at a recent conference, the author of the particular algorithm being displayed started to play around with the simulation, trying it out by placing points on the plane and watching the algorithm do its stuff. In the course of this he, and the people around him, realized that the algorithm being simulated had a bug! Admittedly it was for a degenerate case and not too hard to fix, but even so, the author was appalled. The ability to improve algorithms through the use of animation is an often touted reason for pursuing algorithm animation work. This is, however, the first time I have actually heard of a published algorithm being fixed because its author got to play with the animation.

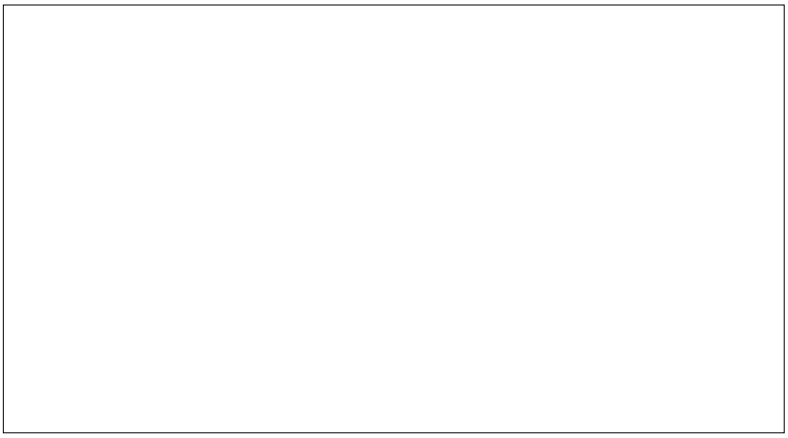


As you can see in the l to r sequence, the anomaly occurs in the third diagram when the points become co-linear

Leslie Kaelbling, noted at the front of this issue for her recent academic accomplishments, was the guest of honor at the department's first professorial baby shower. Dedicated readers of ***conduit!*** will remember that Leslie is also one of two departmental equestrians. (Who is the other? Answer on p. 20.) Thus it was understandable that, when presented with a playpen, she responded, "Oh great—a corral!" Leslie tried putting another gift, a baby blanket, around her neck, but found that "It's got the wrong aspect ratio." Noted Robert Givan, a post-doc working with Leslie and Tom Dean, "...probably the first time that phrase has ever been used at a baby shower—and even more unusual, everyone understood!"

The graphics lab recently had a face lift. The proximate cause was problems with the tables—students frequently would bump their knees on the underworkings of the old ones. Replacing them required moving all the computer equipment, and then one thing led to another: if we are moving all the stuff out of the lab, why not give the lab a paint job while we are at it? So now it is all spiffed up, though I dare anyone to tell me if the picture on the following page was taken before or after the work was done.

Andy van Dam, noted for his leadership, is also noted for his attention to detail. Quite often this means that things are reworked until the last moment. While Andy was chair of this department one departmental proposal only

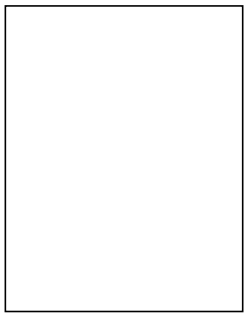


Photo–Kathy Kirman, tstaff

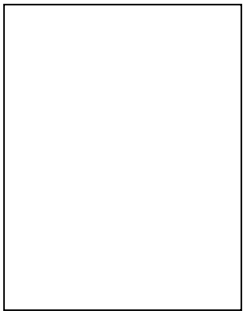
Leslie responds to some good-natured ribbing for showing up an hour late to her surprise baby shower!

made it to Washington on time because then graduate student Dick Bulterman, an amateur pilot, agreed to fly it down on the day it was due. Of course the department picked up the rental tab for the airplane. (Andy, Trina and I all remember this story; unfortunately Tom Doeppner, who has the best memory of us all, says it's apocryphal, and we haven't dared check with Dick.) I was reminded of this recently when upon leaving the elevator here I was almost flattened by one of Andy's UTAs: he was attempting to deliver a hot-off-the-printer overhead transparency to the lecture hall before Andy got to that point in the lecture.

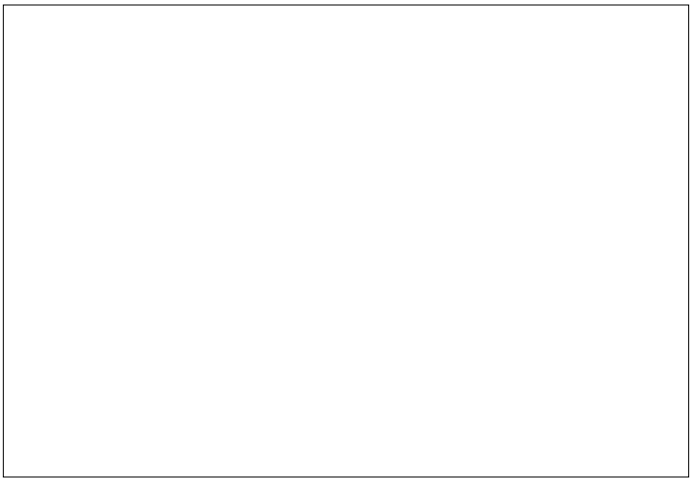
Finally, this is my last Chair's column for ***conduit!***. I have been Chair now for five years, nine months and 25 days. Our two previous chairs, Andy van Dam and John Savage, both did six years and then passed the job on. I am happy to go along with this tradition, and as of July we will have a new chair. Who the lucky person will be has not yet been decided. The choice is made by the Dean of the Faculty, Kathy Spoehr, in consultation with the faculty of the department. About two weeks ago Kathy came and talked with all the faculty (as well as Trina Avery, on the grounds that departmental managers know at least as much as faculty do). Nobody has asked me how this job has changed me, but I figure I will tell you anyway. Mostly my hair color, as the accompanying pictures illustrate.



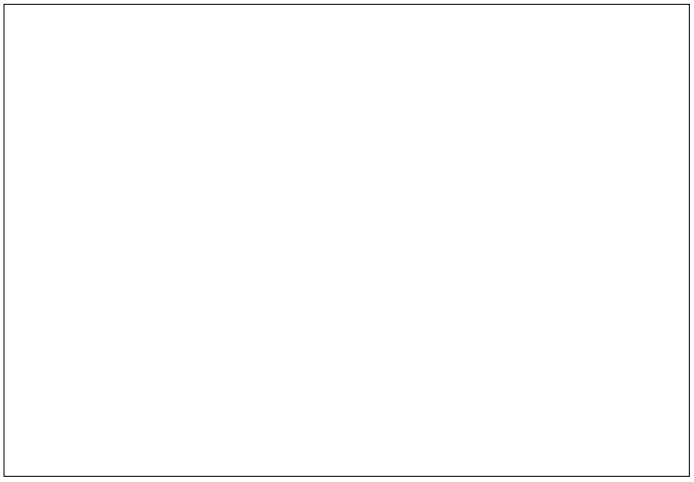
Before



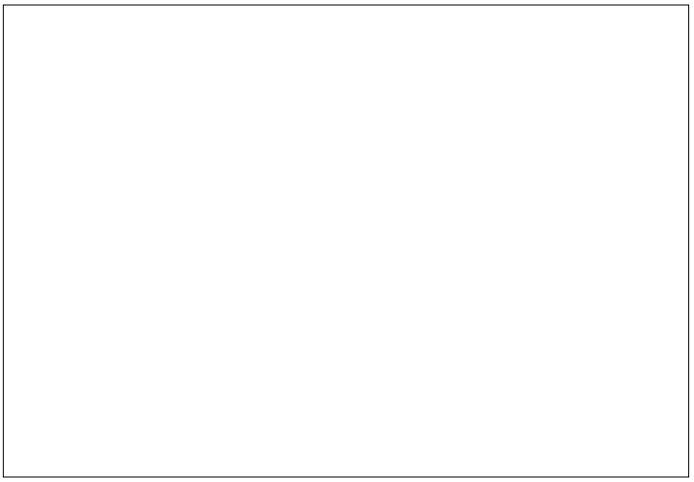
After



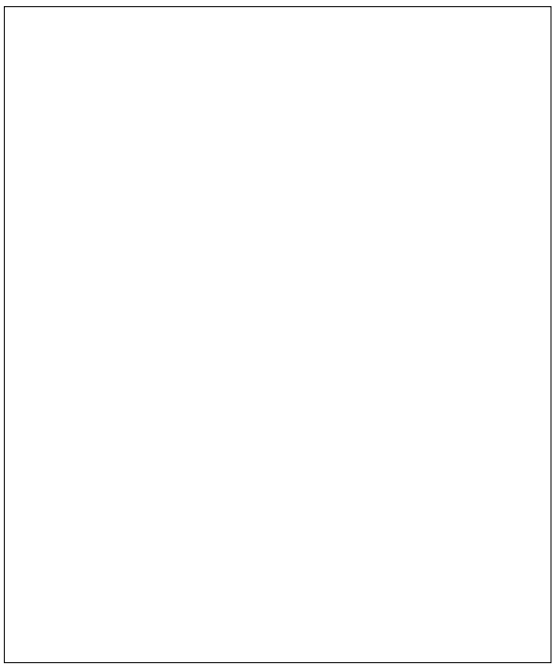
Equipment from the Graphics Lab finds a temporary home in the Lubrano during the recent revamp

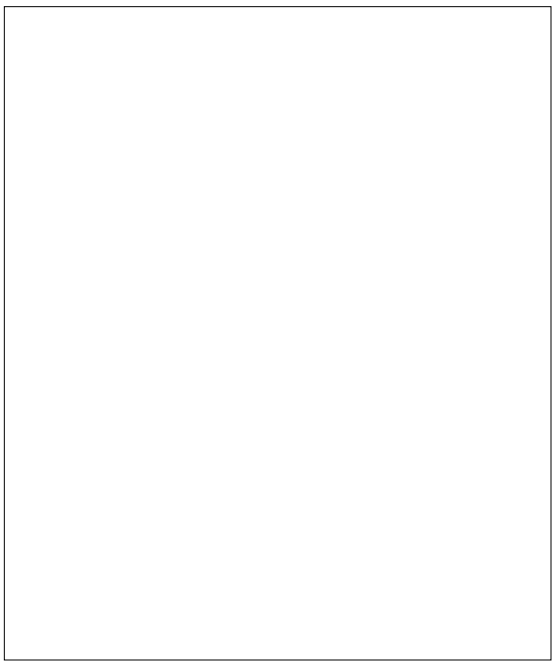


Emptied of contents and clutter, the Graphics Lab was freshly painted before the new tables were installed



Restored to its original ?splendor





Winners from Peter Wegner's CS2 MacPaint competition

conduit!

A publication of
The Computer Science Department
Brown University

Inquiries to: ***conduit!***

Department of Computer Science
Box 1910, Brown University
Providence, RI 02912
FAX: 401-863-7657
PHONE: 401-863-7610
EMAIL: sjh@cs.brown.edu
WWW: http://www.cs.brown.edu/publications/conduit/

Suzi Howe is also an equestrian.

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

conduit!

Printed on recyled paper

Address changes welcomed