conduit!

Volume 2, Number 1 | Department of Computer Science, Brown University | Spring, 1993

GRAPHICS GROUP HAS NEW VIRTUAL REALITY LAB

Andy van Dam and User Interface Researcher Ken Herndon

The Department recently completed a new laboratory for the Graphics Group in which to perform virtual reality (VR) research. The 4th-floor machine room was divided into two. The east side of the room is now a machine room holding several of the larger CPU and framebuffer units previously in the graphics lab. The west side of the room has been converted into space suitable for research in VR. It has been acoustically conditioned with carpet and double-insulated walls, and with acoustic panels. Pass-throughs were put in to facilitate cabling requirements. Track lights were also installed, along with a hologram on the wall produced by Mitch Henrion, formerly of the Graphics Group and now at the MIT Media Lab, and Sarah Lindsley, former CS11 and CS192 head TA, now at Apple.

The renovations also included moving a large air-conditioning unit and several large workstations out of the graphics lab and into the adjoining machine room. The much quieter working environment inside the graphics lab is necessary for our new teleconferencing setup with the other schools (Caltech, Cornell, North Carolina, and Utah) in the NSF/DARPA Science and Technology Center for Computer Graphics and Scientific Visualization. The plan is to collaborate with colleagues at these schools through televideo, transmitting a mixture of live video with workstation feeds.

With sponsorship from Steve Bryson's virtual windtunnel project at the NASA Ames Research Center, the Graphics Group has a variety of VR hardware in the lab. The most significant is the FakeSpace Boom, a stereoscopic display device consisting of two black and white monitors mounted on an armature held by users in front of their eyes. This alternative to a head-mounted display is less invasive and easier to engage and disengage. The boom will be driven by either one or more workstations to display stereo, 3D virtual worlds. As the user swivels the headpiece, the displayed scene is adjusted accordingly to create a feeling of immersion in the virtual world. The Virtex CyberGlove will be used in conjunction with the Boom to record the user's hand postures, which can be recognized by software algorithms and translated into commands for the application. The Ascension Bird tracker will be used with the Boom and Glove to monitor the position of the user's hand in space.

Dan Robbins uses the FakeSpace Boom

Separate from the Boom setup, two Logitech Flying Mice (6D ultrasound tracking devices), one for head tracking and one for hand tracking, will be used in conjunction with a set of StereoGraphics stereo glasses with shuttered LCD lenses to visualize stereo images on a workstation screen. The Hitachi workstation monitor has a very high refresh rate which is synchronized with the shuttered glasses.

The Graphics Group's research will focus on the user interface requirements of computer graphics applications for scientific visualization. They will explore the general areas of navigation through large datasets, interaction with and control of 3D graphical objects in these applications, and immersive environments (including multidimensional input devices and stereoscopic displays). They will work closely with researchers at NASA to develop useful interaction techniques for 3D graphics applications in general, and for NASA's research in computational fluid dynamics in particular.

PERSPECTIVES IN COMPUTER SCIENCE

A recent series of talks "Perspectives in Computer Science," given in the CS Department and summarized in the following articles, presented research ideas of five faculty members in a form accessible to a nonspecialist audience. Eugene Charniak suggested that the standard "knowledge representation" approach to natural language understanding is inadequate and that a statistical approach based on correlations among words in large texts can be remarkably powerful in determining language semantics. Tom Dean explored the use of stochastic automata for robot planning and learning under uncertainty, extending classical uses of automata to embedded systems that capture knowledge of an uncertain world. Franco Preparata discussed the horizons of parallel computing, pointing out the architectural, software, and fundamental modeling problems that stand in the way of its more widespread adoption. Leslie Kaelbling examined refinements of reinforcement learning for robots that continuously interact with their environment via their sensors and effectors, especially the problem of knowing which of its actions are responsible for arriving at a good state. Peter Wegner examined the role of reasoning versus modeling paradigms in problem solving and demonstrated that, because of its greater interactiveness, object-oriented programming is more powerful than logic programming as a component-based paradigm for programming in the large.

Though these talks were prepared independently, they unwittingly struck a common theme, namely, computational empiricism. Each talk focused on the interaction of models with the domain being modeled. Eugene's statistical models, Tom's embedded models, and Franco's parallel models explicitly brought the outside world into the model; so did Kaelbling's models of reinforcement learning and Wegner's preference for interactive over uncommunicative software components. These talks collectively suggest a subtle change of emphasis away from static models that passively specify the inherent structure of a domain towards dynamic models that actively interact with their domain.

"These talks collectively suggest a subtle change of emphasis away from static models....towards dynamic models that actively interact with their domain"

Post-Symbolic AI and Natural-Language Understanding

Eugene Charniak

In artificial intelligence (AI), the standard approach to natural-language understanding assumes that the incoming language is translated into some "semantic" or "internal" representation that represents the meaning of the utterance (or text), and less important things like the particular words or sentence construction are stripped away. This semantic representation is then checked against what the reader already knows about the topic and is used to augment that knowledge. For example, in reading "Joe went to the liquor store. He took a six-pack of beer from the refrigerator unit and paid for it." we use our knowledge of liquor stores, refrigerators, and buying things in stores, etc. to fill in unstated but implied facts like (a) He went to the liquor store to purchase beer and (b) He wanted the beer cold. As can be seen from this example, the program must have at its disposal a "knowledge base" of common-sense knowledge such as the purpose of refrigerators, why one goes to stores, etc.

The branch of AI known as "knowledge representation" has as part of its charge the creation of such knowledge bases, but it has not been very successful. Indeed, most people have given up trying to create a knowledge base of the "common-sense" knowledge we need to understand simple English. (The sole exception, to my knowledge, is Doug Lenat, but excepting Doug himself, nobody I know believes his approach will succeed.) Thus language understanding researchers are in the uncomfortable position of needing a common-sense knowledge base but not believing that it can be effectively realized any time soon, say in the next 10-15 years.

"Language understanding researchers are in the uncomfortable position of needing a common-sense knowledge base but not believing it can be effectively realized any time soon"

The standard approach described above is "deep" in that it tries to uncover any hidden, unstated relations in the text. In order to avoid the problems of common-sense reasoning, I have adopted a statistical approach to natural language processing that gathers statistics about "surface" properties of English and then uses them as a basis for reasoning. Because the analysis is shallow, we need to lower our expectations about what our programs will do. Consider speech recognition (going from speech signals to written text). This is a very hard problem, but we can break it up into two parts: first, figure out the possible words that might have been spoken, and second, decide among the possibilities. For example, suppose that we heard "The man got some beer." but that simply from the sound waves the program could not decide between this and "The man *cot* some beer." A program that could decide that the former is a more likely English sentence than the latter would enable the speech recognition system to make the correct decision. We call such a program a *language model*. Obviously, a perfect language model is a very difficult thing to construct, but even an imperfect model could help a lot and building it can be a lot easier than trying, say, to answer questions (e.g., "Why did Joe go to the store?"). To give you some idea of how simple such a model can be and still be very useful, consider a model which simply asks which of the possibilities for a given word is the most probable given the two previous words. In the case at hand this would ask, after the words "the man", whether "got" or "cot" is more probable. If we have a large amount of English text we can, in fact, collect such statistics and use them.

But we can do more with the statistical approach. Currently a student (Glenn Carroll) and I are trying to learn an English grammar statistically. I will not go into how this is done, but rather just show how it could help to improve our language model. Consider a sentence like "Jack watered Susan's small garden." If we ask what the probability is of "garden" coming after "Susan's small", the number would presumably be quite low, thus giving little aid to a speech recognition system that needed help figuring out what word this is. On the other hand, suppose we have done a grammatical analysis and ask, instead, what the probability is that "garden" is the direct object of the verb "to water". This is quite likely indeed, and thus much more help.

Furthermore, although the statistical approach gathers statistics about very shallow things, it nevertheless can be used to get at properties concerning the meanings of words. For example, work elsewhere has shown that words can be put into groups that seem to reflect their meanings by looking at very simple things like what words appear around them, what verbs have them as direct objects, etc. One such experiment found groups like (street, block, avenue, corner) and (school, classroom, teaching, math). These techniques can produce silly combinations when there is not much data to go on, as a single example of a word may have atypical properties. Furthermore, some semantic distinctions cannot be captured by these techniques, apparently because they are not reflected in surface phenomena. For example, one study found that the word most similar in surface features to "hot" was "cold", which makes sense if you think about it. Nevertheless, this suggests that there is no *a priori* reason why statistical techniques cannot be pushed to deeper levels of analysis.

Thus it would seem that the statistical approach has a lot to recommend it. Because it takes a shallow view of language, it is not affected by the problems of common-sense knowledge discussed earlier. At the same time, the approach can produce useful byproducts: language models for speech recognition, improved spelling correctors (which might be able to correct misspelled words that are themselves words), improved document retrieval, and even aids to language translation. Finally, the statistical approach allows us to bootstrap to deeper and deeper levels of language as our tools improve.

Robot Planning and Learning

Thomas L. Dean

Traditional computer science has concerned itself primarily with data processing tasks (e.g. sorting, searching, path planning) that ignore the outside world. With the advent of networked computers, satellite communications, and autonomous mobile robots, the focus is shifting to systems embedded in the real world and continually faced with tasks that involve processes not under computer control (e.g., air traffic control). Much of theoretical computer science is based on discrete structures such as labeled graphs and finite-state machines that may not seem relevant to continuous-change, real-time applications.

We demonstrate how discrete, deterministic automata and their stochastic counterparts can play an important role in embedded systems. In particular, we consider how automata can be used as a representation to cope with the uncertainty and time pressure with which real-world robotics applications are fraught. We consider applications in which robots with noisy sensors track other robots in office buildings, build maps of their environment to help in navigation, and respond to unpredictable events in a timely manner. The mathematics of discrete structures and the tools of asymptotic complexity allow the computer scientist to analyze the time and space requirements for applications like map learning, path planning, and a range of lower-level tasks often considered the exclusive domain of control theory.

"We provide a two-stage iterative refinement algorithm for approximating the optimal plan under time constraints"

l to r: George Keith, Gort, Jonathan Monsarrat, Huey and Tom Dean

As an illustration of how to manage real-time planning for robots, we provide a way of specifying goals for domains whose dynamics are modeled as stochastic processes. In our approach, an optimal plan to achieve a state satisfying some proposition as soon as possible corresponds to a policy (a mapping from states to actions) that maximizes the expected value with respect to a particular reward function. Finding the optimal plan can be done using standard methods; however, these methods are at best polynomial in the number of states in the stochastic process, where the number of states is exponential in the number of propositions (or state variables). We provide a two-stage iterative refinement algorithm for approximating the optimal plan under time constraints. The algorithm employs a reduced version of the stochastic process that uses a subset of the set of states describing the entire domain. The first stage extends the state space of the reduced process on each iteration and the second stage computes a policy with respect to the reduced process. We describe a set of experiments involving a mobile robotics application. This talk describes joint work with Ken Basye, Leslie Kaelbling, Jak Kirman, and Ann Nicholson.

Horizons of Parallel Computing

Franco Preparata

Parallel computing—today a major focus in computer science and engineering—is a modern instance of "computing," which, in its most common connotation, is the diligent execution by some agent of a sequence of well-defined "commands." These commands belong to a repertory of actions that the agent (human or mechanical) is capable of carrying out.

Underlying this characterization is a model of (serial) computation, which traditionally consists of (1) a facility to store initial data, intermediate results, and final answers (*memory*); (2) a facility to store the program (also *memory*); (3) a facility to execute the repertory of commands (CPU). Data are "variables," identified by the places in memory where they are stored. This extremely useful model, known as the von Neumann machine (also partly foretold in the work of Babbage and Turing), has in its simplicity its power and its limitation (von Neumann bottleneck): there is a single datapath between memory and CPU.

"Massive parallelism is made possible by an extremely successful technology, which still offers more design latitude than system designers can exploit"

The scene becomes considerably more complicated when technological progress allows the deployment of multiple concurrently operating CPUs and independently addressable memory modules: the interconnection of these modules becomes the focus of a parallel system, with an extremely large space of solutions. When the impressive advances in integration (VLSI) made parallel systems a distinct possibility, the choice of interconnections (the "architecture") became a difficult system question. As we shall see below, several years later the picture is considerably clearer.

Is parallelism a recent innovation in computing? The answer is a qualified "no." One could say that parallelism has always been deployed in computers to the extent permitted by technology. While the arithmetic of the earliest machines was essentially serial due to hardware costs and reliability, parallel arithmetic units were used as soon as safe and economical. Similarly, special-purpose I/O controllers (early examples of parallel co-processors) were used as soon as they were economical.

Today, massive parallelism is made possible by an extremely successful technology, which—a very rare event—still offers more design latitude than system designers can exploit. Since the late seventies, considerable research has been carried out on the design and analysis of parallel algorithms. This effort has not only fostered the understanding of specific applications and of their inherent parallelism (or lack thereof), but has also led to the identification of general computational paradigms (algorithmic patterns specified by their data-exchange structures) and of their supporting architectures. The most significant results of this endeavor are some fundamental relationships between algorithms and architectures and the realization that—contrary to original beliefs (or fears)—very few architectures appear to have computational relevance, among them notably the mesh, the tree, the hypercube, and their hybridizations. Indeed, these interconnections are the basis of the parallel systems that have been marketed in the recent past.

Despite considerable insights into parallel algorithmics and the appearance of some parallel machines, parallel computation is not enjoying the success that had been anticipated. There are several reasons for this slow progress, first and foremost a mismatch between current algorithm design and realistic machine structures, as well as a lack of program portability. Most parallel algorithm design refers to systems where the number of processors is tailored to the problem size (fine-grained systems). A more likely scenario is one where the system deployed is fixed (fixed interconnection of processors) and the problem size is arbitrarily large (coarse-grained systems). In such situations, for large problem sizes the serial task dominates, permitting optimal speed-ups. Very little research has focused on this model. Program portability is crucial to user acceptance of parallelism. Unfortunately, the diversity of the proposed architectures and therefore the lack of consensus on the fundamental system primitives (whose efficiency is obviously architecture-dependent) have prevented convergence on a parallel programming style.

> “In a technology where limits are hypothetically attained, most of the classical tenets no longer hold”

Finally, technological progress has brought us closer to the physical limits of computing (speed-of-light and device-size limits). In a technology where such limits are hypothetically attained, most of the classical tenets no longer hold. For example: (1) the only scalable computing structures are of the near-neighbor type (meshes), and parallelism emerges naturally from the need to maintain a fixed memory/CPU ratio; (2) latency hiding is not feasible; (3) slow-down emulations no longer obey the neat trade-offs known as Brent’s principle; (4) the exploitation of locality may lead to speed-ups in excess of the traditional ones.

This brief synopsis simply outlines the challenges of a research area that will be crucial to the attainment of some of our societal goals, notably the so-called “grand challenge” problems. Indeed, the performance demanded by some of these applications is likely to be afforded only by a judicious use of massive parallelism.

Learning from Trial and Error

Leslie Pack Kaelbling

One of the standard problems in machine learning is *concept learning*, in which a computer program is presented with a set of input-output pairs. The program must find a function (typically of a particular syntactic form) that maps the given inputs to the given outputs. In noisy environments, the problem is often posed as one of minimizing the mean squared error between the given outputs and the ones generated by the function. There are a variety of techniques for solving this problem using methods of statistics and computer science.

> “My interest is in constructing robots that act robustly and autonomously in dynamic, noisy environments”

My interest is in constructing robots that act robustly and autonomously in dynamic, noisy environments. In order to do this, the robots must learn at least some parts of their control programs. Unfortunately, there is no good source of input-output pairs for robots to learn from. Instead, robot learning is more appropriately cast as learning from trial and error, or *reinforcement learning*. In reinforcement learning, the computer program continuously interacts with the environment through the robot’s sensors and effectors. The robot senses a particular world state, then the program must chose what action to take. A scalar *reinforcement signal* is made available to the robot to indicate how good the resulting state of the world is for the robot. We would like to construct computer programs that learn a mapping from perceptual states to effector actions that maximize reinforcement over the long term.

In the last few years, reinforcement learning has received a great deal of attention. One important question is how to control the robot’s exploration of its environment. In a noisy domain, simply trying a particular action in a particular situation once is not a reliable indication of its true value. Instead, the robot must repeatedly sample different actions, in order to determine which ones have the best results. But if the robot spends too much of its time exploring its domain, it will spend too much time performing actions that do not gain high reinforcement. If, on the other hand, the robot does not experiment enough, it may not discover the best strategies. I have developed an algorithm, based on statistical confidence interval estimates of underlying probabilities, that provides a good solution to this problem.

Another crucial problem in reinforcement learning is *temporal credit assignment*. When the robot must endure a series of states with very low reinforcement value in order to get to an especially good state, how is it to know exactly which of its actions was crucial to arriving in the good state? Algorithms from dynamic programming have been adapted to this problem with very good results on small problems. Unfortunately, they are very inefficient in large domains. Inefficiencies occur both in the amount of time taken to process one input instance and in the number of learning instances required before behavior becomes good.

My most recent work has introduced a notion of hierarchy into reinforcement learning with temporal credit assignment. This work applies to the special case of goals of achievement, in which there is some small penalty for being in every state and a large reward for being in a "goal" state. This model is appropriate for navigation and other task-achieving behaviors. In this case, it is possible to learn strategies that are only approximately optimal, but to gain considerable efficiency in both the time to process a single instance and in the number of learning instances required to achieve a good behavior. Detailed technical papers are available on request.

Reasoning Versus Modeling in Computer Science

Peter Wegner

Deductive reasoning involves proving theorems (goals) from axioms (facts) by rules of inference and is exemplified in computer science by logic programming (LP). Models describe objects of an application domain by their relevant observable properties and are exemplified by object-oriented programming (OOP). Reasoning and modeling represent two very different paradigms of problem solving, deriving respectively from philosophical rationalism and empiricism.

"Logical agents cannot interact flexibly because they must consider too many alternatives before making a definite commitment"

Though LP and OOP have equivalent computing power (both can simulate Turing machines), LP is weaker than OOP as a component-based interaction paradigm since logic programs are not reactive. Because of potential backtracking, logical agents must be able to retract their actions and cannot commit to sending irrevocable messages. This supports the intuition that logical agents (whether human or computer) cannot interact flexibly because they must consider too many alternatives before making a definite commitment. Logic programs may be viewed as autistic in their inability to interact, while object-oriented programs are interactive.

Component-based and state transition models are complementary paradigms of problem solving, just as particle and wave theories are complementary ways of describing the properties of matter. Communication power is a more relevant metric than Turing computability for modeling application domains and more generally for programming in the large. It has fundamentally different notions of observability, composition, and equivalence and gives rise to different models of computation. The importance of communication and interaction as a basis for computing is eloquently described in Robin Milner's Turing lecture in the January 1993 *Communications of the ACM*.

Prolog programs have a *procedural reading*, but the converse mapping from interactive to reductive programs is not generally possible. Hobbes was correct that *Reasoning is but Reckoning* since reasoning is easily convertible into reckoning. He would have been incorrect in asserting that *Reckoning is but Reasoning* since reckoning is not generally reducible to deductive reasoning.

Sets of rules of the form "goal :- sequence of subgoals" with the same predicate in the head can be viewed as *components* of a Prolog program. A computation in Prolog progressively reduces goals to sequences of subgoals until the goal is solved or proved unsolvable. Alternative clauses of a component, representing alternative ways of reducing a given goal to a set of subgoals, are tried and then retracted (by backtracking) if they cannot succeed. Components are not reactive because the need to revoke all effects on backtracking precludes sending of irrevocable messages. Concurrent logic languages abandon backtracking (and logical completeness) because of its incompatibility with reactiveness, adopting a committed-choice selection mechanism similar to that of object-oriented programming. This sacrifice of logical completeness to imperative commitment suggests that reasoning should be supplemented by (or replaced by) interaction and commitment in modeling the real world.

Rob Netzer’s graduate class listens in the lobby

THE 11th INDUSTRIAL PARTNERS PROGRAM SYMPOSIUM

The 11th IPP Symposium on Progress in Distributed Object-Oriented Computing presented researchers from industry—Sun, IBM and DEC—and from Brown. Peter Wegner opened by discussing research needed to extend object-oriented programming from single-user, shared-memory systems to scalable, multiuser distributed systems. Steve Gadol of Sun Microsystems discussed Spring, an experimental software platform designed to support secure distributed computing. Joshua Auerbach and Robert Strom of IBM presented Concert, a portable distributed software platform based on Strom’s Hermes system, and explored the development of a language and operating environment for multiapplications within the Hermes system. Andrew Black of DEC examined the role of types in object-oriented systems, focusing on distributed systems like Emerald. Stan Zdonik and Steve Reiss reviewed their research on distributed object-oriented databases and object-oriented environments. Several Brown graduate students working in this area also presented their research progress.

There was some lively discussion of open questions in distributed object-oriented computing, and diverse opinions were expressed on the relatively slow acceptance of distributed, object-oriented computing as a practical, widely used technology of application programming. The information-packed day was followed by our usual reception, where participants could thaw out over *hors-d’œuvres* and wine.

Because of the large number of participants, we arranged a video presentation of the lectures outside the conference room. The picture above shows members of a Brown graduate class on distributed computing listening to the presentations.

(Next symposium schedule on back page)

Symposium speakers, l to r: Dave Langworthy, Steve Gadol, Steve Reiss, Andrew Black, Stan Zdonik (background), Rob Strom, Josh Auerbach, Peter Wegner, and Scott Meyers

activities@cs.brown.edu

Thomas Dean. Besides giving an invited talk at Columbia University entitled “Robot Planning in Uncertain Environments” at the Symposium on Intelligent Systems, Tom presented an overview and chaired a panel on “Learning in Navigation” at the DARPA/NSF Workshop on Machine Learning and Computer Vision, in West Virginia.

Thomas Doeppner. Tom gave an invited talk at the CERN Conference on Computers in High-Energy Physics in Annecy, France, last September. The talk was entitled “Open Software: UNIX/OSF/UI, etc.”

Leslie P. Kaelbling. Leslie’s latest book, *Learning in Embedded Systems*, published by MIT Press, will appear in May. According to Richard S. Sutton of GTE Laboratories, quoted in *AI Magazine*, “This is likely to become a foundational, problem-establishing book in the rapidly growing area of reinforcement learning. It includes significant new results, is self-contained and scholarly, and includes excellent references and coverage of related work.” Last March Leslie was an instructor at a NATO Advanced Study Institute on the Biology and Technology of Intelligent Autonomous Agents in Trento, Italy.

Paris C. Kanellakis. Paris is one of three co-chairmen coordinating the First Workshop on Principles and Practice of Constraint Programming, sponsored by the Office of Naval Research, in Newport, RI, on April 28-30. This will be an interdisciplinary meeting focusing on constraint programming as a general paradigm for computation.

Philip N. Klein. Phil organized a workshop this March on Approximation Algorithms for Hard Optimization Problems, sponsored by DIMACS (Center for Discrete Mathematics and Computer Science), at Rutgers University. He returns from sabbatical soon.

Franco Preparata. Franco delivered plenary addresses at the International Workshop on Parallel Computing in Paderborn, Germany, and at INRIA’s 25th anniversary symposium in Paris, in addition to several seminars at various universities. He was also invited to deliver three public lectures on Parallelism in Computational Geometry at the meeting of the French Computational Geometry Community in Grenoble. He was pleased at the publication of the Japanese translation of his computational geometry text, following the Russian and Chinese translations.

John Savage. During his recent sabbatical leave, John gave five invited talks including one at the British Colloquium on Theoretical Computer Science. He was also co-chair of the Brown/MIT Conference on Advanced Research on VLSI and Parallel Systems. He is currently a member of the MIT Corporation Visiting Committee for the Department of Electrical Enginering and Computer Science.

Roberto Tamassia. Roberto was on sabbatical from January to May, 1992. He was an organizing committee member for the International Work Meeting on Graph Drawing held in Rome last June, and guest editor of a special issue of *Algorithmica* on Graph Drawing. Over the past year, Roberto was an invited lecturer at nine institutions, worldwide. In June, ’93, he will be on the program committee for the 19th Workshop on Graph-Theoretic Concepts in Computer Science, in Utrecht; in August, he will serve in the same capacity for the Third Workshop on Algorithms and Data Structures (WADS ’93) in Montreal.

Andries van Dam.
Andy joined the Technical Advisory Boards for Microsoft Corporation and Ithaca Software. He was the keynote speaker at the Graphics Visualization & Usability Center Convocation and the First Annual PHIGS Users Groups Conference, and an invited lecturer at the University of Toronto, the NECUSE Workshop at Harvard University, and Brown University's San Francisco Campaign Celebration. He was on the program committee and a senior reviewer for SIGGRAPH '93, the program committee for the Third Eurographics Workshop on Object-Oriented Graphics, and a member of the National Research Council Committee on Virtual Reality Research and Development.

Pascal Van Hentenryck. Pascal is an invited speaker at the Third International Workshop on Constraint Logic Programming (WCLP '93) in Marseille and the Third International Workshop on Static Analysis (WSA '93) in Padova, both held in March. He is also sitting on five program committees in three different areas, AAAI '93, ICLP '93, LPAR '93, PPCP '93, and WSA '93, and is a local organizer of the First International Workshop on Principles and Practice of Constraint Programming. Three of his papers will appear as chapters in two books on Intelligent Scheduling and Constraint Logic Programming.

Peter Wegner. In October, Peter chaired a panel on megaprogramming at OOPSLA '93. He was an invited speaker at the TriAda conference in November, and presented a paper on Logic Versus Object-Oriented Programming in Hawaii in January. He also organized a workshop at Harvard on the Introductory Computer Science Curriculum.

(See article on p. 13)

DATA-SWAPPING TECHNIQUE USED BY U.S. CENSUS BUREAU

A data-swapping technique developed in 1978 by Steve Reiss and Tore Dalenius, of Brown's Division of Applied Math, for controlling the risk of invasion of privacy in a statistical database, was used by the U.S. Census Bureau in the 1990 census. They applied this technique

"The objective was to maximize the level of useful statistical information provided subject to the condition that confidentiality not be violated"

to release actual census data (not just statistics) to researchers, while maintaining privacy and without compromising the confidentiality of the database.

The Bureau is required by law to release data in a way that does not identify an individual, and thus its objective was to maximize the level of useful statistical information provided subject to the condition that confidentiality not be violated. Three types of procedures for dealing with disclosure risk for 100 percent data were investigated: Suppression, Controlled Rounding, and Confidentiality Edit. The last is the Bureau's name for the Reiss/Dalenius data-swapping technique (perhaps the original name sounded somewhat immoral, suggested Tore!). After considerable study and testing of the three methods, Confidentiality Edit was determined by the Census Bureau to be a means of providing sufficient uncertainty in the data to allow for adequate protection against disclosure while at the same time providing reliable data.

Confidentiality Edit is based on selecting a small sample of census households and interchanging their data with those of other households that have identical characteristics on a set of selected key variables, but are in different geographic locations. The matching and interchanging operations are controlled on the key variables of number of persons in household, population characteristics of race, Hispanic origin and age, and on housing characteristics of units in building, rent/value and tenure. The result is that census counts for total persons, totals by race, Hispanic origin and age 18 and above, as well as housing counts by tenure, are not affected by the confidentiality edit. According to Richard A. Griffin of the Census Bureau, the technique worked very well. Because the data need just be swapped once, no further study or manipulation is needed to protect the data. Secondly, although swapped, *all* of the data can be made available to researchers. The only disadvantage noted by the Census Bureau is that the technique is so unobtrusive people might not realize their confidentiality has been maintained!

“Data-Swapping: A Technique for Disclosure Control,” Tore Dalenius, Steven P. Reiss, *Journal of Statistical Planning and Inference*, vol. 6, pp. 73-85, North-Holland Publishing Company, 1982.

FACULTAS LUDENS

The Department chairman, **Eugene Charniak**, became interested in art and art collecting in graduate school after reading his roommate’s (an architecture major) art magazines. There is, however, a family penchant for painting—his

Eugene and the ceramic collection in his office

father painted representational pieces during his retirement, and Eugene painted abstract works as a hobby during his years in graduate school. The highpoint of his art career was the honorable mention of a piece in the Boston Visual Artists Union show. Eugene has numerous abstract prints and sculpture at home, including a table with a complex linear structure made by a Rhode Island School of Design graduate. Casually he began acquiring ceramic mugs until he realized he was starting a collection—it now totals 31 pieces, varying from utilitarian to whimsical and artistic. These make a colorful display in his office, together with one of his more entertaining possessions, a piece called “The Pitchfork Chair” (banned from their home by his wife, and perhaps the only piece they haven’t agreed on!); this is a very serviceable chair with the head of a pitchfork for the back and axe handles for legs.

There always seemed to be an artist in **Tom Dean**’s family, and he always enjoyed working with tools, the more involved and complicated, the better. Before starting his career in computer science, he tried to make a living as a sculptor. His work was both abstract and representational, the latter largely figurative. He progressed from metal sculpture to large wood pieces, for which he created his own tools. He also developed special-purpose carving tools for antique reproduction work—creating ball and claw feet, finials, etc. Solving the problems associated with machine tools was as rewarding for Tom as the creation of an actual sculpture. Never a traditionalist, he was always looking for new techniques and considered himself a frustrated engineer using art as an outlet. On one of his trips to used machine-tool shops in Washington, DC, Tom found a numerically controlled milling machine—a manufacturing tool driven by a primitive computer. His fascination with this very large piece of equipment led him to electrical engineering school to learn how to build computers. His first degree was in mathematics, then to graduate school, and ultimately to his faculty position in the Computer Science Department. Should he return to sculpture, Tom would create light-hearted robotic art in the style of Jean Tinguely—an avant-garde sculptor who created a machine which actually destroyed itself (a bit of the Luddite in him, Tom notes!). While he appreciates art, he sees himself not as an artist, but as someone who loves building things.

Tom Doeppner learned scuba diving in 1985 at a Brown scuba course. In the summer of 1986 he bought his first underwater camera, a Nikonos 5 non-SLR camera. Taking fish pictures with this manual-focus camera was fairly difficult but certainly rewarding; coral was always a good subject—it didn’t move! His recent purchase of an autofocus camera (a Nikon 8008S inside a Tussey housing) with two lenses (28-85-mm zoom and a 60-mm macro) has simplified picture taking considerably, particularly judging distances. Tom prefers diving in the South Pacific—it’s more exotic than the Caribbean, with fewer people and greater varieties of underwater life—giant clams, lionfish, clownfish and pelagic creatures. One of the biggest thrills happened

Tom at 60 feet in the Caribbean

recently in the Galapagos when the guide he was with suddenly took off, swimming flat out; Tom followed and saw a line of 6-foot eagle rays; he counted 20. Of all the out-of-the-way places he has visited—the Maldives, Hawaii, Grand Cayman, Bonaire, Roatan, Cayman Brac, the Great Barrier Reef, Fiji, Tasmania, Belize, Saba, Solomon Islands, Turks & Caicos Islands, New Guinea, and the Galapagos—his favorite is the Solomons, where he lived aboard a dive boat for ten days diving four to five times a day. The only way to dive that often is by using a dive computer which keeps track of how long you have been down and how much nitrogen is in your bloodstream—Tom hasn't yet had any close calls—he puts his trust in computers!

Cyclamen grace the skylights on either side of Franco's front door

Franco Preparata's particular liking for arts and crafts and his fascination with the effect of light on glass led him, in the mid-1980s, to take up the painstaking craft of leaded glass work, an art form that became popular at the turn of the century in Belgium, France and the United States. Before the leading can be applied, copper banding must first be affixed around each piece of smoothed and shaped glass—a very delicate technique. He first saw someone doing this work and asked how it was done; he then developed his own 'tricks' and, after many unsuccessful attempts, perfected his technique. Finding a good source of colored and interestingly textured glass is often difficult, and knowing what curves are feasible with which kind of glass and what cuts a piece will take comes only with years of practice. Franco creates works purely for use as windows—an ideal place is a bathroom, where the colored glass creates privacy and prevents boredom! The handsome corner pieces in his office windows are his own handiwork. Up until now his work has been figurative—mainly flowers—but, as soon as his *Kale Sansui* garden is finished, he plans to try abstract forms.

Franco's other hobby is *Kale Sansui*, his Zen garden. *Kale Sansui* translates literally as 'dry, water, mountain.' It is created from gravel, stones, and small plantings—ground covers, mosses, and dwarf varieties of plants and shrubs—and evokes the feeling of a miniature landscape. He has two such gardens at his East Side home. His interest was piqued on a lecture trip to Kyoto in the 1980s where he saw these lovely gardens and decided the creation of such beauty was a worthwhile endeavor. He read extensively and learned to avoid certain unpleasant symmetries and adjacencies in planning the garden, a process which takes considerable time. If the garden is successful, a little raking and weeding is all it takes to maintain a source of continuing relaxation and a place for endless contemplation.

Steve Reiss's office looks more like a greenhouse, as vegetable seedlings sprout from atop the filing cabinets. Besides strawberries and raspberries, he grows a wide variety of vegetables on his Rehoboth, Massachusetts, farm. Those of us left in the Department during August share his abundant tomato harvest. Snap peas are his favorite crop, but last year's *pièce de resistance* was a giant pumpkin, too huge to be weighed. Steve is also an avid baseball fan, playing on the departmental softball team (the team's best player) and coaching Little League since '86.

Biking ten miles each morning on his trusty ten-speed is **John Savage**'s way of starting the day. He particularly enjoys biking in the state forest on Martha's Vineyard, where he has a house.

Besides listening to classical music, especially Bach and Vivaldi, **Roberto Tamassia** enjoys good food, travel, and sports. He feels fortunate that he can combine business with pleasure on his travels, which have included such exotic places as Bali—future trips will take him to Sicily, France, and Australia. Roberto used to play chess competitively, winning several regional tournaments in his native Italy. Table tennis and soccer ('It's in the blood!') are his favorite sports; however, he recently became addicted to downhill skiing.

When asked if he had any hobbies, **Andy van Dam** said "No, I work!"

Peter Wegner's hobby is making puns. He particularly likes the one about the tunes President Clinton will play on the saxophone during his presidency—of course, they will be Al Gore Rhythms. We all hope the Vice President will call the tune!

In the early '70s, while walking in Harvard Square, **Stan Zdonik** heard fiddles playing and stopped to listen—his efforts to identify the music led him to discover bluegrass, and so began his 20-year love affair with bluegrass music. With its roots in Appalachian music, the blues and southern gospel three-part harmony, Bluegrass came into its own in the late '30s when Bill Monroe defined the current form—a five-piece band with mandolin, guitar, bass, banjo, and fiddle. In 1973 Stan started playing mandolin, and in 1976 started The Boston Bluegrass Union, a non-profit group which organizes monthly concerts in Cambridge with bands from all over the country; once a month members bring their instruments to jam at 'picking parties.' Stan has been president for the past 16 years. Before joining the Brown faculty, he occasionally played (and still does) in a four-man band whose other members included Click and Clack of National Public Radio fame. (He once appeared on their auto repair show billed as their long-lost brother Cluck!) Stan's other avocation is auto maintenance—no one but he has worked on his 1983 Toyota Cressida, which now has 210,000 miles on it! When it was just about to hit the 200,000 mark, Stan got up early with a couple of family members and drove until 200,000 rolled around; then they all got out in party hats throwing confetti and treated themselves to a celebration breakfast! There's something about oil-encrusted-hands-on tinkering that contrasts most satisfactorily with the more cerebral pursuits of the computer scientist. To hear some bluegrass, just call Stan's home phone and listen to his answering machine—The Nashville Bluegrass Band, of course.

PEW-SPONSORED NECUSE WORKSHOP

The New England Regional Workshop on The Introductory Computer Science Curriculum at Harvard on January 23-24 1993 included four speakers from Brown among the representatives of over 20 New England computer science departments attending. Andy van Dam described his experiences in teaching an object-oriented introductory computer science course. Ted Sizer, in an after-dinner speech, presented his innovative ideas for high school education that are being tried out in his Coalition for Essential Schools. Peter Wegner presented the results of a questionnaire concerning current undergraduate computer science teaching. Leslie Kaelbling described curriculum reform at Brown, which is in the process of developing a more coherent and modern freshman and sophomore curriculum starting from an object-oriented approach and working out the implications of this change in the introductory curriculum on the later curriculum.

The first day of the workshop included four talks by Alan Tucker of Bowdoin, Kim Bruce of Williams, Andy van Dam of Brown, and Peter Denning of George Mason University, followed by a panel discussion by writers of innovative introductory books, including Hal Abelson of MIT, Eric Roberts of Stanford, and Alan Bierman of Duke University. The second day included a panel on curriculum issues, such as the role of laboratories and the role of theory versus practice, as well as two working groups on the first and second courses in computer science. Though no definitive conclusions were reached, it was generally felt that the right issues had been discussed and that some progress had been made towards understanding the issues. The workshop was sponsored by the Pew Foundation under a grant to the New England Consortium for Undergraduate Science Education (NECUSE), and was organized by Peter Wegner.

Technical Staffers, l to r, Max Salvas and Peter Galvin install new Sparc 10 boards in the AI Lab (Cleese looks on!)

FROM THE CHAIRMAN, Eugene Charniak

There was one sad occurrence during these last six months. Jeff Vitter, who has been with the department practically since its inception (he came in 1980), has been named the chairman of the Computer Science Department at Duke University. While officially Jeff is just on ‘‘leave of absence’’ from Brown for a short period, it does not seem likely that he will be returning. Those of you familiar with the history of the department may remember that a few years ago Robert Sedgewick left to become chairman at Princeton. While we are pleased that the rest of the field thinks so highly of our faculty members, we hope this does not become a habit.

One of the unanticipated pleasures of the chairman’s job is that it makes me do something I otherwise might not find time for—in this case talk in detail to all of the faculty about what they have done in the past year. One thing I have noticed about our faculty is how in demand they are as speakers, both in this country and abroad. So this year I have started an informal ‘‘contest’’ to identify the professor who gave invited talks in the most countries. The winner is Roberto Tamassia, who gave talks in six countries this past year: Canada, Indonesia, Italy, Japan, the Netherlands, and the US. The runners-up were Pascal Van Hentenryck and Peter Wegner, each with four countries.

In this last semester we have found a new use for our atrium—a robot playpen. As the accompanying photograph shows, Tom Dean

Gort negotiates obstacles in the atrium

and his students have been using the atrium to test the new robot they're building. (See the last issue of *conduit!* for more on Tom's robotics work.) Making room for this requires moving the plants around, which occasionally means having to dodge leaves in unexpected places, but this is more than made up for by seeing the robot trundle around.

Since the fall issue of *conduit!*, Scott Meyers and Gail Mitchell have successfully defended their doctoral dissertations. Scott's topic is "Representing Software Systems in Multiple-View Development Environments." He will be staying on as a post-doc until the end of the summer, after which he will write a successor to his well-received book, *Effective C++*; he will also be engaged in private consulting. Gail's topic is "Extensible Query Processing in an Object-Oriented Database." She will be going to GTE Labs in Waltham, Massachusetts. Two other dissertations will be defended over the summer—Marion Nodine's topic is "Interactions: Multidatabase Support for Planning Applications," and Paul Howard's topic is "The Design and Analysis of Efficient Lossless Data Compression Systems."

In the last semester the Department changed the undergraduate curriculum in several key ways. CS11, Computer Programming, Problem-Solving and Applications, is now taught in an object-oriented style. This change alone has forced significant revisions in several courses downstream. The old CS12, Fundamentals of Programming, now has a new number and name—CS31, Introduction to Computer Systems. This reflects its transformation from an assembly language programming course to a machine organization course. CS12 and CS21 (now CS31, Introduction to Computer Systems; and CS16, Algorithms and Data Structures) have switched places in the curriculum. The new CS16 now follows CS11, since the Department felt it made more sense to continue the programming of CS11 into CS16, and put off machine organization (CS31) to second year. There were several other changes as well, and the fall issue of ***conduit!*** will have an article about our new curriculum.

As we noted in the previous *conduit!*, the department is replacing all of its "old" Sun Sparc 1s with Sparc 10s. This is being done in stages: the servers came in first, and then the research machines. (The educational machines for the teaching lab have special graphics requirements and will not arrive until this summer.) I must say that the Sparc 10s have made quite a difference in my personal research. I have been working (with Glenn Carroll, a graduate student) on grammar learning using statistical techniques, and our programs used to take several days to complete. Now with the Sparc 10s this has been reduced to 24 hours. Next we hope to exploit Quahog, a program developed by John Bazik of our technical staff that distributes processes around on all of our machines in such a way as to be invisible to users. By using many machines we hope to get the time down to an hour. Ah, progress!

Graduate student Cindy Grimm and Tia her Lesser Sulphur-Crested Cockatoo are a familiar and appealing sight around the Department

conduit!

A publication of The Computer Science Department
Brown University

Suzi Howe . . . . Editor-in-Chief
Katrina Avery . . . . . Writer & Editor
Jeff Coady. . . . Technical Support

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

Photography: Ken Herndon and Suzi Howe.

The 12th IPP Symposium

Object-Oriented Database Systems

April 27, 1993

8:30	Registration and Continental Breakfast 4th Floor, CIT Building
9:00	Introductions by Eugene Charniak, Chairman
9:15	Overview by Stan Zdonik, Host
9:30	**The Promise of Distributed Computing and the Challenges of Legacy Information Systems** Michael Brodie, GTE Labs
10:30	B R E A K
11:00	**Standards for Object-Oriented Data Management** Thomas Atwood, Object Design, Inc.
11:45	**Applying an Object-Oriented Approach to Integration of Distributed Applications** Mike Renzullo, DEC
12:30	B U F F E T L U N C H
1:30	**Querying in Object-Oriented Databases** Stanley Zdonik, Brown University
2:00	**Indexing Techniques for New Data Models** Paris Kanellakis, Brown University
2:45	B R E A K
4:00	**Using Locality to Improve Query Processing Efficiency in Object-Oriented Databases** Shamim Naqvi, Bellcore
5:00	Wrap-up
5:15	R E C E P T I O N (5th Floor)

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

conduit!

NON-PROFIT
U.S. Postage
PAID
Providence, RI
Permit #202

Printed on recyled paper

Address changes welcomed