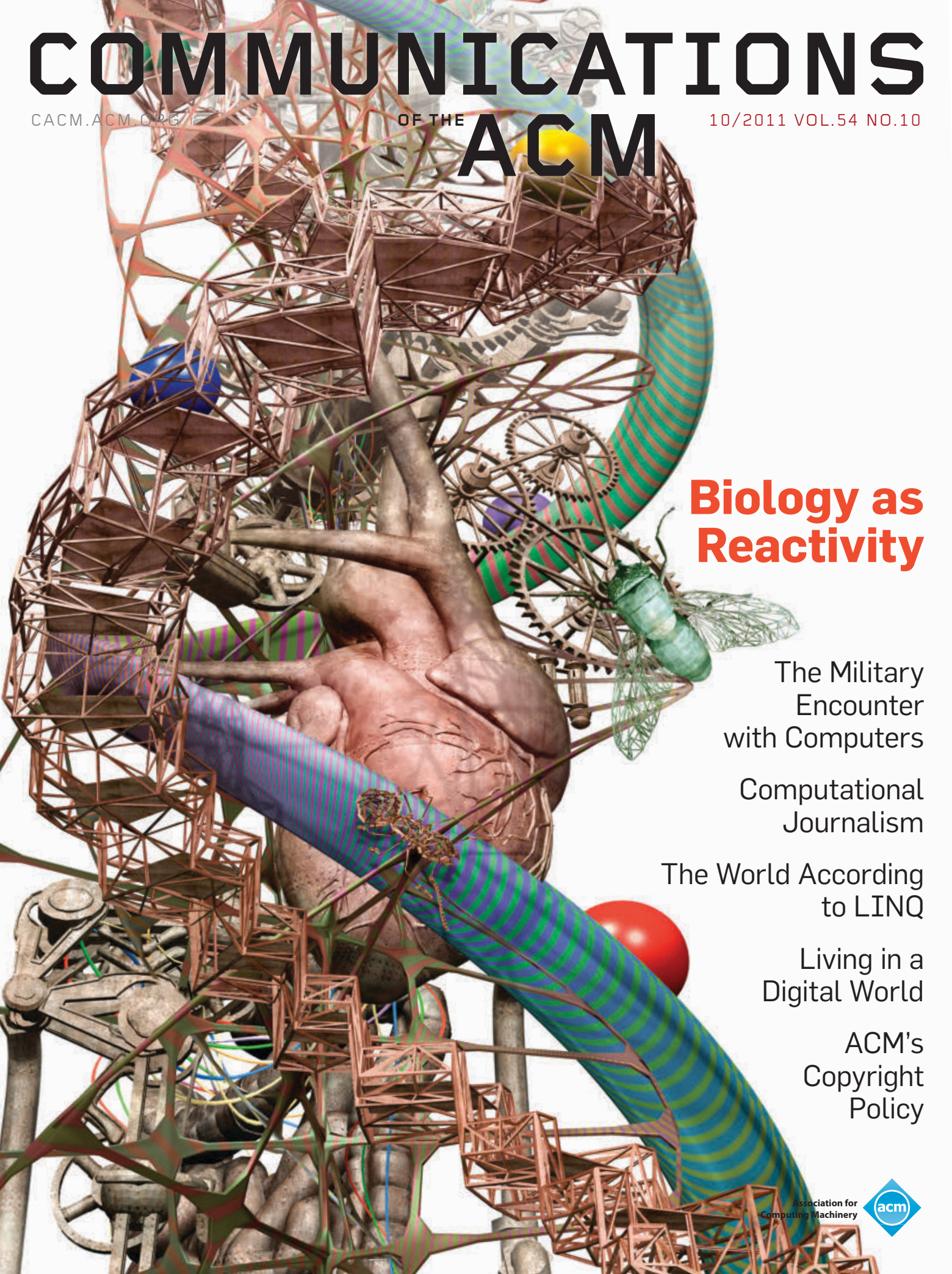
COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

10/2011 VOL.54 NO.10



Biology as Reactivity

The Military Encounter with Computers

Computational Journalism

The World According to LINQ

Living in a Digital World

ACM's Copyright Policy

Association for Computing Machinery

ACM Multimedia 2011

Scottsdale, Arizona, USA

Nov. 28-Dec. 1, 2011

Venue: Hyatt Regency Scottsdale Resort and Spa at Gainey Ranch

http://www.acmmm11.org

ACM Multimedia, the worldwide premier multimedia conference, has an exciting 2011 program that includes:

- 9 half-day tutorials
- 3 keynotes:
  - Genevieve Bell, *Intel Fellow, Director, Interaction and Experience Research*
  - Alex Pentland, *MIT Human Dynamics Lab*
  - Arnaud Robert, *The Walt Disney Company*
- Research presentations, plenary poster sessions
- 11 technical workshops, 1 women mentoring workshop
- 1 doctoral symposium
- 3 thought-provoking panels
- An open-source software program
- The industry-inspired Multimedia Grand Challenge competition
- Technical and industrial demonstrations

Early registration deadline: October 31

General Chairs
K. Selçuk Candan (Arizona State U.)
Sethuraman Panchanathan (Arizona State U.)
Balakrishnan Prabhakaran (U. Texas at Dallas)

Get short, timely messages from ACM Multimedia 2011. Join Twitter and follow @acmmm11.
Acmmm11 is on Facebook. Sign up for Facebook to connect with Acmmm11.

Distinguished Speakers Program

talks by and with technology leaders and innovators

Chapters • Colleges & Universities • Corporations • Agencies • Event Planners

Need a Great Technical Speaker for Your Next Event?

The Association for Computing Machinery (ACM), the world's largest educational and scientific computing society, now provides colleges and universities, corporations, event and conference planners, and agencies – in addition to ACM local Chapters – with direct access to top technology leaders and innovators from nearly every sector of the computing industry.

Book the speaker for your next event through the ACM Distinguished Speaker Program (DSP) and deliver compelling and insightful content to your audience at a remarkably reasonable price. Our program features renowned thought leaders in academia, industry and government, speaking about the topics that matter most in the computing and IT world today. Our booking process is simple and convenient, please visit us at: **www.dsp.acm.org**.

The ACM Distinguished Speaker Program is an excellent solution for:

Corporations Educate your technical staff, ramp up the knowledge of your team, and give your employees the opportunity to have their questions answered by experts in their field.

Colleges & Universities Expand the knowledge base of your students with exciting lectures and the chance to engage with a computing professional in their desired field of expertise.

Event & Conference Planners Use the ACM DSP to help find compelling speakers for your next conference and reduce your costs in the process.

ACM Local Chapters Boost attendance at your meetings with live talks by DSP speakers and keep your chapter members informed of the latest industry findings.

Captivating Speakers from Exceptional Companies, Colleges & Universities

DSP speakers represent a broad range of companies, colleges and universities, including: IBM, Microsoft, BBN Technologies, Raytheon, Sony Pictures Imageworks, National Institute of Standards and Technology, Lawrence Livermore National Laboratory, Siemens Information Systems Bangalore, Stanford University, University of Pennsylvania, University of British Columbia, Georgia Tech, Carnegie Mellon, UCLA, McGill University, Tsinghua University and many more.

Topics for Every Interest

Over 250 lectures are available from nearly 100 different speakers with topics covering Software Engineering, High Performance Computing, Human Computer Interaction, Artificial Intelligence, Gaming, Mobile Computing, and dozens more. Some of our most popular lectures include:

- Electronic Voting in the 21st Century
- Software Engineering Best Practices
- Software Under Siege: Viruses and Worms
- Spatial Databases and Geographic Information Systems
- Careers in Computing – How to Prepare and What to Expect

Quality is Our Standard

The same ACM you know from our world-class digital library, magazines and journals is now putting the affordable and flexible Distinguished Speaker Program within reach of the computing community.

To select a speaker for your next event, get started today at www.dsp.acm.org.
If you have questions, please send them to acmdsp@acm.org.

Microsoft Research
The DSP is sponsored, in part, by Microsoft Europe

Association for Computing Machinery

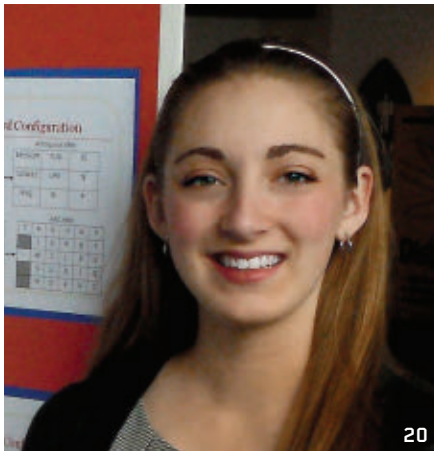
Advancing Computing as a Science & Profession

COMMUNICATIONS OF THE ACM

Departments

Last Byte

News

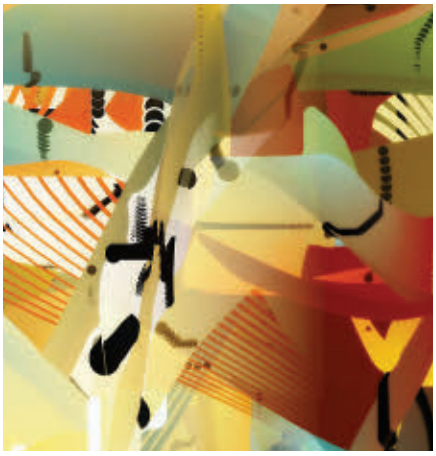


Viewpoints

PHOTOGRAPH COURTESY OF NATALIE NASH

10/2011
VOL. 54 NO. 10

Practice



Articles' development led by acmqueue
queue.acm.org

About the Cover:
This month's cover story, beginning on page 72, explores the connection of biology with reaction systems. The key to these efforts will unlock a far greater understanding of living systems. Cover image by J.F. Podevin. For more on his work, see http://www.podevin.com.

ILLUSTRATION BY DAVE BOLLINGER

Contributed Articles

Review Articles

Research Highlights

Association for Computing Machinery
Advancing Computing as a Science & Profession

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

Subscriptions
An annual subscription cost is included in ACM member dues of $99 ($40 of which is allocated to a subscription to *Communications*); for students, cost is included in $42 dues ($20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is $100.

ACM Media Advertising Policy
Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current Advertising Rates can be found by visiting http://www.acm-media.org or by contacting ACM Media Sales at (212) 626-0686.

Single Copies
Single copies of *Communications of the ACM* are available for purchase. Please contact acmhelp@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to
Communications of the ACM
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA

Association for Computing Machinery

Printed in the U.S.A.

DOI:10.1145/2001269.2001270

Ronald F. Boisvert and Jack W. Davidson

ACM's Copyright Policy

Addressing the needs of authors, readers, and the Association.

In 2010 ACM published approximately 18,000 articles in its Digital Library (DL). As a condition of publication, a designated author of each of those articles is required to sign an agreement that transfers copyright to ACM. Many authors execute the transfer with little thought about what rights they are transferring, what rights they are retaining, and perhaps most importantly, why transferring copyright to ACM provides any benefits to them and to the larger computing community. Some authors view the copyright transfer as a "land grab" by ACM of their intellectual property and only grudgingly execute the transfer form. This misperception about the goals and purpose of ACM's copyright policy sometimes appears in blogs and postings to discussion lists on academic publishing. Our intent here is to briefly highlight some key provisions of ACM's copyright policy (the full text can be found at http://www.acm.org/publications/policies/copyright_policy) to dispel some misperceptions, and to describe some recent changes to the policy that benefit both our authors and the Association.

Background

ACM's original copyright policy was reformulated in 1994 specifically to address access and usage rights in the digital environment. The major principles in this landmark policy were widely adopted by other scholarly societies and eventually by commercial publishers as well.

Nonetheless, few could anticipate the rapid adoption of electronic publishing and the accompanying development of enhanced services and functions. To keep pace with and adapt to these changes, the Publications Board continually reviews our copyright policy. These reviews consider input collected from the various stakeholders: authors, subscribers, and the Association. Indeed, the reformulated policy has undergone multiple revisions over the past 17 years. The current version, Version 6, was adopted in January 2011. As part of our most recent review, the Publications Board also carried out a comparative review of our copyright policy with the copyright and licensing policies of several major scientific publishers. We did this not only to determine how ACM's policy compared to those of its peers (we compare very favorably), but also to uncover good ideas that might be useful to incorporate in our policy.

Rights Retained by Authors

Many authors assume they are giving up all rights to use their material in future works (as is common with commercial publishers) when they transfer copyright to ACM. In fact, ACM's policy allows authors to reuse the material in their future works without restriction, without cost, and without any need to ask for permission. The complete list of rights retained by authors under ACM's copyright transfer agreement is:

- All other proprietary rights to the work such as patent,
- The right to reuse any portion of the work, without fee, in future works of the author's own, including books, lectures, and presentations in all media, provided that the ACM citation and notice of the Copyright are included,
- The right to revise the work,
- The right to retain copyright to embedded images (for example, figures) with independent artistic value,
- The right of an employer that originally owned copyright to distribute definitive copies of its author-employees work within its organization, and
- The right to post author-prepared versions of the work covered by ACM copyright in a personal collection on their own home page, on a publicly accessible server of their employer, and in a repository legally mandated by the agency funding the research on which the work is based. Such posting is limited to noncommercial access and personal use by others, and must include the ACM copyright notice both embedded within the full-text file and in the accompanying citation display as well.

The last three items deserve special comment. The right to retain copyright to embedded images with independent artistic value was a change the Publications Board adopted last January. Many of the images that appear in ACM's computer graphics conference proceedings and journals require substantial effort to produce (both creative effort and production effort) and the authors or their companies often wish to separately exploit these images as artworks. Based on feedback from the graphics community, the Publications Board added

the ability for authors to selectively retain copyright to such images.

The rights addressed by the last two items have always been somewhat controversial in the community. That is not surprising as these items expose the tension between ACM providing a means for free and open access to every article and sustaining a robust, publishing program with subscription revenue. This tension was noted in David Wise's June 1999 *Communications* article that described the Publications Board rationale for granting these rights (see http://dx.doi.org/10.1145/303849.303870).

In fact, as a not-for-profit professional society, ACM has priced its DL subscriptions at the low end of the spectrum for commercial and nonprofit society publishers alike. The result is the DL is now available at some 3,000 academic, government, and industrial sites in 75 countries, accounting for an estimated 15–20 million persons with potential access. For calendar year 2011 we are projecting approximately 15 million full-text downloads by 1.5 million active users. So, the "pay wall" erected by subscriptions is not a barrier to the vast majority of research scientists. And, the subscription revenues contribute to DL services that are free to the community at large, such as the DL's embedded *Guide to Computing Literature*, a free, discipline-specific discovery service that indexes the top research publications of the entire field, with Web pages that aggregate citation and usage data in profiles for individual authors, conferences, and institutions.

ACM continues to look for ways in which it can strike an even better balance between open access and the need to generate revenues to extend and sustain its editorial services, databases, and servers. One new feature that ACM will roll out in the fall will enable authors to obtain a special link for any of their ACM articles that they may post on their personal page. Anyone who clicks on this link can freely download the definitive version of the paper from the DL. In addition, authors will receive a code snippet they can put on their Web page that will display up-to-date citation counts and download statistics for their article from the DL.

For calendar year 2011 we are projecting roughly 15 million full-text downloads by 1.5 million active users. So, the "pay wall" erected by subscriptions is not a barrier to the vast majority of research scientists.

This referrer-linking service has major benefits for our authors and for the ACM DL. First, it provides *free* access to the definitive version of the article permanently maintained by ACM in the DL, enabled by the authors through their home-page bibliography. There will no longer be a need to post author-prepared versions. The benefit to the author and ACM is that download statistics maintained and reported by the DL will be more complete as they will reflect activity of users accessing the article from the author's home page, and automatic indexers will point to the article in the DL rather than non-definitive copies hosted elsewhere without any promise of being permanently maintained.

Benefits of Copyright Transfer

One might wonder, given the generous rights retained by authors, why ACM requires authors to transfer copyright to ACM at all. In fact, the transfer of copyright to ACM provides substantial benefit to the computing research community and to authors.

By owning exclusive publication rights to articles, ACM is able to develop salable publication products that sustain its top-quality publishing programs and services; ensure access to organized collections by current and future generations of readers; and invest continuously in new titles and in services like referrer-linking, profiling, and metrics, which serve the community. Furthermore, it allows ACM to efficiently clear rights for the creation, dissemination, and translation of collections of articles that benefit the computing community that would be impossible if individual authors or their heirs had to be contacted for permission. Ownership of copyright allows ACM to pursue cases of plagiarism. The number of these handled has been steadily growing; some 20 cases were handled by ACM in the last year. Having ACM investigate and take action removes this burden from our authors, and ACM is more likely to obtain a satisfactory outcome (for example, having the offending material removed from a repository) than an individual.

Conclusion

ACM's policies regarding author rights, reuse permissions, and copyright transfer, coupled with its business model offering multiple access paths to content—some of them free of charge—and a commitment to keep subscription prices for both individuals and institutions very low—all combine to maximize wide distribution of ACM publications and global access to them. As part of its continued stewardship of this resource, ACM will continue to seek input from its stakeholders and adapt its policies to improve the experience of both the authors that generate content and the many users that access this content. C

Ronald F. Boisvert and **Jack W. Davidson** are co-chairs of the ACM Publications Board.

DOI:10.1145/2001269.2001272

In the Virtual Extension

To ensure the timely publication of articles, Communications *created the Virtual Extension (VE) to expand the page limitations of the print edition by bringing readers the same high-quality articles in an online-only format. VE articles undergo the same rigorous review process as those in the print edition and are accepted for publication on merit. The following synopses are from articles now available in their entirety to ACM members via the Digital Library.*

contributed article

DOI: 10.1145/2001269.2001297

Don't Turn Social Media Into Another 'Literary Digest' Poll

Daniel Gayo-Avello

Content published in microblogging systems like Twitter can be data-mined to take the pulse of society. Indeed, a number of studies have praised the value of relatively simple approaches to sampling, opinion mining, and sentiment analysis. In this article, the author plays devil's advocate, detailing a study conducted in late 2008/early 2009 in which such simple approaches largely overestimated President Barack Obama's victory in the 2008 U.S. Presidential election. The author conducts a post-mortem of the analysis, extracting several important lessons.

Twitter is a microblogging service for publishing very short text messages (only 140 characters each), or tweets, to be shared with users following their author. Many Twitter users do not protect their tweets, which then appear in the so-called public timeline. They are accessible through Twitter's own API, so are easily accessed and collected.

Twitter's original slogan—"What are you doing?"—encouraged users to share updates about the minutia of their daily activities with their friends. Twitter has since evolved into a complex information-dissemination platform, especially during situations of mass convergence. Under certain circumstances, Twitter users not only provide information about themselves but also real-time updates of current events.

Today, Twitter is a source of information on such events, updated by millions of users worldwide reacting to events as they unfold, often in real time. It was only a matter of time before the research community turned to it as a rich source of social, commercial, marketing, and political information.

The goal of this article is to focus on one of the survey's most appealing applications: using its data to predict the outcome of current and future events.

contributed article

DOI: 10.1145/2001269.2001298

Computing for the Masses

Zhiwei Xu and Guojie Li

The fields of computer science and engineering have witnessed amazing progress over the last 60 years. As we journey through the second decade of the 21st century, however, it becomes increasingly clear that our profession faces some serious challenges. We can no longer solely rely on incremental and inertial advances. Fundamental opportunities and alternative paths must be examined.

In 2007, the Chinese Academy of Sciences (CAS) sponsored a two-year study on the challenges, requirements, and potential roadmaps for information technology advances into year 2050. This article presents a perspective on a key finding of that study: A new paradigm, named *computing for the masses*, is needed to cope with the challenges facing IT in the coming decades.

The CAS study focused on China's needs. However, the issues investigated are of interest to the worldwide computing community. For instance, when considering the drivers of future computing profession, it is critical not to underestimate the requirements and demands from the new generations of *digital native* population. As of July 2010, 59% of China's 420 million Internet users were between the ages of 6–29 years old. The time frame of 2010–2050 is not too distant a future for them. These digital natives could drive a ten-fold expansion of IT use.

Computing for the masses is much more than offering a cheap personal computer and Internet connection to the low-income population. It means providing essential computing value for all people, tailored to their individual needs. It demands paradigm-shifting research and discipline rejuvenation in computer science, to create augmented Value (V), Affordability (A), and Sustainability (S) through Ternary computing (T). In other words, computing for the masses is VAST computing.

BLOG@CACM

The *Communications* Web site, http://cacm.acm.org, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

twitter
Follow us on Twitter at http://twitter.com/blogCACM

DOI:10.1145/2001269.2001273 http://cacm.acm.org/blogs/blog-cacm

From Idea to Product; How Schools of Education Can Help CS

Daniel Reed discusses how researchers communicate their project ideas to companies and product groups and get them successfully adopted. Mark Guzdial considers whether schools of education could create more high school CS teachers.

Daniel Reed "Technology Transfer: A Contact Sport"
http://cacm.acm.org/blogs/blog-cacm/99627
October 4, 2010

It is fall and football season in the United States. We are still playing soccer too—football everywhere else—but that is another story. American football has long been called a contact sport, meaning the players intentionally collide to move the ball toward the goal line. However, it remains a far cry from rugby, which to my untrained eye seems indistinguishable from unarmed combat.

How, one may wonder, does football relate to technology transfer? For that matter, what do I mean by the phrase "technology transfer" itself? There are broader definitions, but for the sake of discussion, let's define technology transfer as the mechanism via which research ideas are communicated to and adopted by companies and their product groups.

The success or failure of technology transfer depends on many factors, from the personalities and skills of the people involved, through the timing and appropriateness of the offering, to the risks and costs associated with the new idea. No single mechanism is guaranteed to succeed, although there are many mechanisms that are likely to fail.

DANIEL REED

"In the embedding model, researchers become members of the product team, working side by side on the product in the same environment as the product developers."

Sharing the Love

Opening the technology transfer conversation with a product group by disparaging the current product or questioning the intelligence of the developers is unlikely to create a receptive audience for one's brilliant research result, no matter how scintillating and erudite the subsequent description may be. Nor will woeful obliviousness to the history that led to the current product design, for it is important to understand context and competition. Finally, neither will bad timing. Reporting a better way to engineer a product just before it is finished will accomplish nothing other than depressing the product team, though it may provide guidance for the next product generation.

One needs to build a long-term, productive, and positive relationship between research and product development. It is about creating mutual benefit, sharing the love.

On the positive side, I have observed a continuum of mechanisms that facilitate the transfer of ideas. None is universally successful in isolation, and even in aggregate they may fail. However, the set is far more powerful than the elements, and all can benefit from true contact sport engagement.

Papers. These are the stock and trade of researchers, whether in academia or in government or corporate

research laboratories. Yes, members of many product groups read research papers, but an abstract idea, even with supporting experimental data, is rarely the catalyst for uptake of a new idea. A well-written paper does, however, provide the technical details needed when an idea is ultimately adopted.

Demonstrations. These are the "science fair" instantiations of ideas, the early instances that are little more than mockups of the idea. They work on limited data, are constructed using laboratory rather than product components, and they are unreliable at best. However, their virtue is in allowing nonresearchers to see and experience the idea in concrete form. Often, seeing really is believing, and a prototype can be convincing in ways that a paper is not.

Prototypes. Prototypes are the next stage, where the demonstration is sufficiently robust that it can be used routinely by individuals and groups other than the creators. The robustness allows product groups and developers to test the idea in more realistic conditions and contexts, and see how it fares relative to existing technologies.

Near Products. Near products are products in all but name, suitable for shipment with little more than cosmetic changes. Creating a near product can be effective if the product team is interested and willing but understaffed, or if they lack the specific technical skills needed to create a product-quality instance of the idea.

Contact Engagement

Papers, demonstrations, prototypes, and near products help, and they may sometimes be sufficient to transfer an idea into a product. However, they are pale shadows of the true contact sports, either direct embedding with the product team or long-term engagement and partnership. In the embedding model, researchers become members of the product team, working side by side on the product in the same environment as the product developers. Because the utility of so many new ideas depend on culture and context—things rarely written down—as well as personal trust and connections, contact engagement is often the most successful approach.

Remember, game plans, no matter how elaborate, rarely work without modification. The players on the field innovate and adapt. Technology transfer is a contact sport. It can be contentious at times, but the rewards for successful transfers are enormous, both in a sense of personal accomplishment and pride and in the long-term success of the companies involved.

MARK GUZDIAL

To get 10,000 high school CS teachers in five years, "we desperately need to involve people who know how to prepare and develop teachers—faculty in schools or colleges of education across the country."

Mark Guzdial "Computer Science Needs Education Schools. Desperately."

http://cacm.acm.org/blogs/blog-cacm/100067

October 13, 2010

In the new National Science Foundation (NSF) "Computing Education in the 21st Century (CE21)" program solicitation, proposers are explicitly asked to "design, develop, and evaluate the impact of pre-service and in-service efforts and strategies that enhance K–14 teaching expertise in computing." In Education-speak, "pre-service" means (mostly) undergraduate teacher education programs and "in-service" means professional development for teachers in the classroom. The solicitation points to the "CS 10K" goal: To have 10,000 high school teachers capable of teaching the new Advanced Placement (AP) exam in computer science (CS) by 2015. Today, we have about 2,000 AP CS teachers in the U.S.

How are we going to get to 10,000 high school CS teachers in five years? Here's a quick answer: We're not going to get there alone. We desperately need to involve people who know how to prepare and develop teachers—faculty in schools or colleges of education across the country.

Let's quickly set aside the idea of computer science faculty teaching all those 10,000 future high school teachers ourselves. What do we know about preparing lifelong career teachers? There are employers in IT who argue that we are not particularly good at producing our existing graduates for today's software development careers. High schools are a completely new kind of employer, with new kinds of stakeholders, and new kinds of jobs. Yes, we know computer science. They know the rest. We need a partnership.

How do we convince the schools of education that they want to work with us? That's a harder problem. Budgets are tight. Education schools are not eager to develop new teacher education programs. The just-released "Running on Empty" report shows that few states are teaching anything about computing, so there has to be a parallel effort to create demand for new computer science teachers. From the education schools' perspective, they already teach STEM teachers, and they probably already teach pre-service courses about "technology," although often that means "How to teach a class how to use Excel." Why should they branch out into computer science?

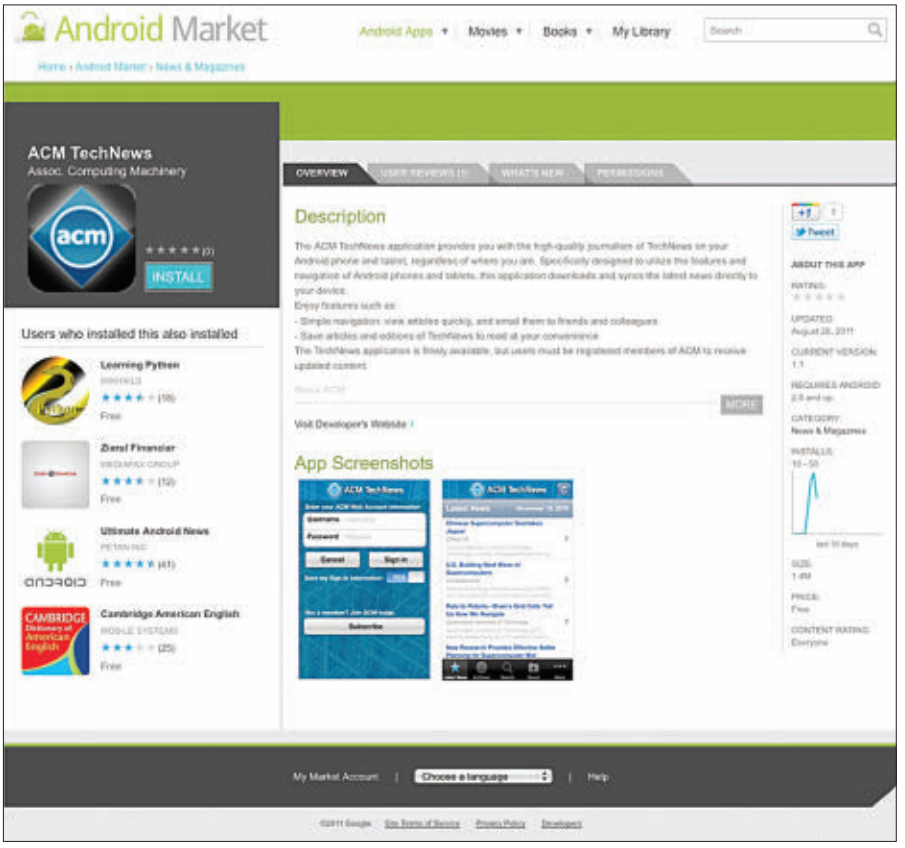
There is a good, mercenary answer: There's money in it. Besides the new NSF CE21 program, the current U.S. administration is funding more efforts in STEM education. The Department of Education and the White House have said clearly that computer science is part of STEM, but little of the STEM education funding is aimed at computer science. That's an opportunity with little competition right now. The time is ripe to form partnerships between CS and education to improve computing education in high schools. C

Daniel Reed is vice president of Technology Strategy & Policy and the eXtreme Computing Group at Microsoft. **Mark Guzdial** is a professor at the Georgia Institute of Technology.

DOI:10.1145/2001269.2001274 Scott E. Delman

ACM TechNews Now Available in the Android Market

After the successful launch of the ACM TechNews iPhone and iPad apps, which have now been downloaded by thousands of ACM members from the Apple iTunes store since their launch last year, ACM now introduces an Android version available for both Android smartphones and tablets. As an increasing number of ACM members consume information on their mobile devices, ACM will continue to provide new and compatible options for accessing content on these devices.



The design and functionality of this new app are nearly identical to the iOS versions. Like the existing versions, this app is free to download but one must have a current ACM membership to receive newly published issues of ACM TechNews three times each week.

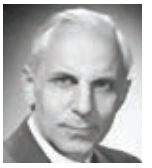
For those interested in downloading the app directly from the Android Market, please visit http://bit.ly/q4sONr.

We hope you will enjoy using this new app and please be sure to send us your feedback directly through the app itself. Thank you.

ACM Member News

FORMER ACM PRESIDENTS REMEMBERED

Two former ACM presidents died in July—Franz L. Alt, 100, on July 21 and Daniel McCracken, 81, on July 31.



Born in Vienna, Austria, on November 10, 1910, Alt fled the Nazis to come to the U.S. in 1938. He was a mathematician who studied set-theoretic topology and econometrics. Drafted by the U.S. Army during World War II, Alt worked at the Aberdeen Proving Ground, where he was a member of the Computations Committee, and helped the army adopt automatic computers. After the war, he moved to the National Bureau of Standards, from 1948 to 1967, where he helped lead the federal government's early use of computers.

In 1947 he was a founder of the ACM, and served as its third president, from 1950 to 1952. He was editor of the *Journal of the ACM* from 1954 to 1958, and was the first recipient of the Distinguished Service Award in 1970. In 1994, he was among the first group to be inducted as a Fellow of the ACM.

McCracken was born on July 23, 1930, in Hughesville, MT, and earned degrees in mathematics and chemistry, as well as a master of divinity degree from Union Theological Seminary. He worked at General Electric starting in 1951. McCracken wrote *Digital Computing Programming*, the first programming textbook, in 1957. He left General Electric in 1959 to be a consultant and write a series of books on Fortran and Cobol; his Fortran books were the standard textbooks for more than two decades. McCracken taught at City College of New York from 1981 until his death.

McCracken was president of the ACM from 1978 to 1980, and chaired the ACM Committee on Computers and Public Policy. He was inducted as an ACM Fellow in 1994.

—Neil Savage

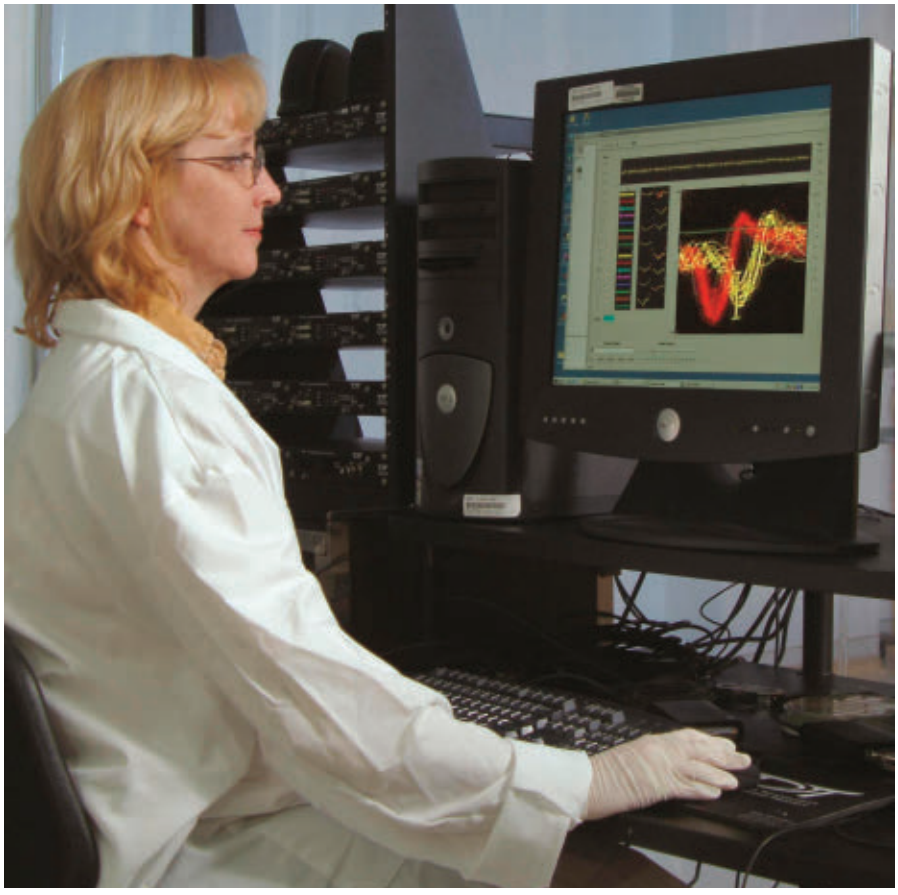
Science | DOI:10.1145/2001269.2001275 Kirk L. Kroeker

Improving Brain-Computer Interfaces

Researchers are demonstrating advances in restorative BCI systems that are giving paralyzed individuals more effective ways to communicate, move, and interact with their environment.

Speculative fiction has long entertained the idea of humans interfacing with machines at the level of thought, resulting in enhancement technologies that not only sidestep the limitations associated with the fragile human body, but also supplement the brain's own shortcomings in processing information or accessing data. While fictional renderings of human-machine interfaces typically take the form of supplementary enhancements for healthy individuals, scientists doing research in brain-computer interface (BCI) technologies have been developing innovative restorative strategies for those who have lost basic functions, such as sight, hearing, and movement.

BCI research, which draws on several fields, such as neuroscience, computer science, physics, and electrical engineering, has led to new developments in nontraditional approaches to the physiological problems that have been resistant to traditional medical solutions. These developments include deep brain stimulators for those who have Parkinson's disease, cochlear implants



Brain-computer interface researcher Dawn Taylor at the Cleveland VA Functional Electrical Stimulation Center.

for those who have lost their hearing, and communication devices for those who are paralyzed. Today, researchers are demonstrating real-world restorative BCI systems, both invasive and noninvasive, that are giving paralyzed individuals more effective ways to interact with their environment and even move.

The complex issues that scientists deal with in this area are numerous, and include challenges ranging from practical lab logistics, sensor hardware, and data-processing systems to team members who have come to the work from very dissimilar disciplines, making it difficult for the field to establish and maintain consistent terminology. For Dawn Taylor, a research scientist working in this area, such problems are not insurmountable. "With a close-knit lab, everyone learns a common vocabulary fairly quickly," says Taylor, who conducts her research at the Cleveland Clinic Department of Neurosciences and the Cleveland VA Functional Electrical Stimulation Center.

Despite the ongoing challenges, research in BCI technologies is resulting in real help for people with severe disabilities, says Taylor. In her current work, Taylor is focusing on using brain signals to trigger movement in paralyzed individuals, bypassing damage in the spinal cord. "We are getting pretty good at decoding someone's intended arm and hand movements from recorded brain activity," she says, noting that her colleagues are now able to restore arm and hand function in paralyzed individuals by using implanted stimulators to activate paralyzed muscles. The main goal for her at this point is to link these two technologies—on the one hand effectively decoding the brain's movement intentions and on the other hand successfully stimulating paralyzed muscles to produce movement—to enable paralyzed people to move their limbs just by thinking about doing so.

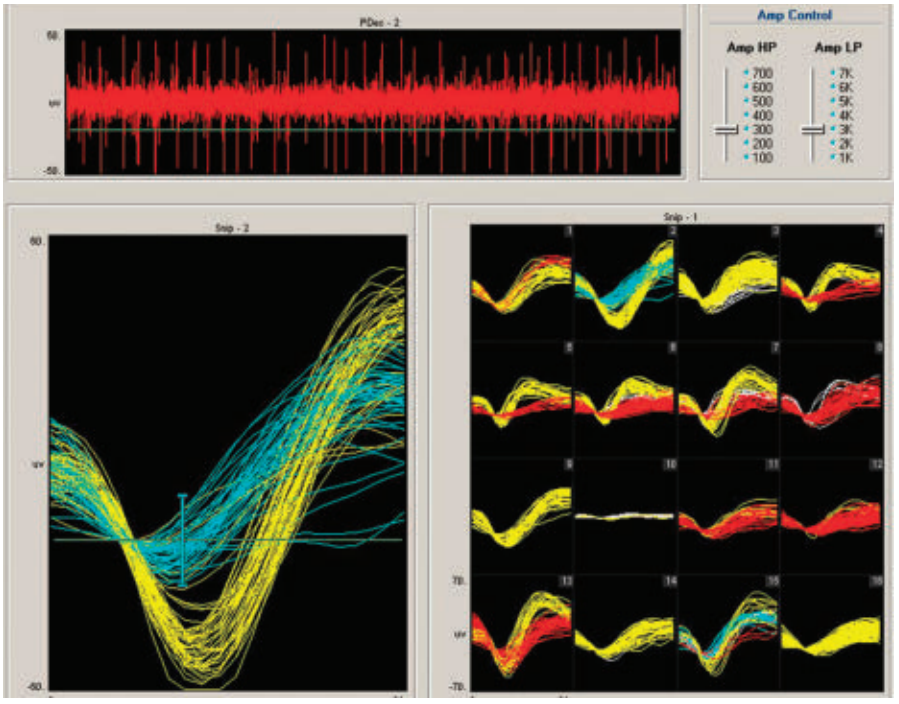
Despite the ongoing challenges, research in BCI technologies is resulting in real help for people with severe disabilities, says Dawn Taylor.

To devise a brain-controlled typing system for a patient who is unable to communicate, one method would be to display a keyboard on a screen and have the patient think about moving his or her fingers to each letter. While it would be possible to decode the movement path the patient is imagining and use that data to control a mouse on the screen, Taylor says it might be more efficient to decode the final goal of each pointing movement and select each letter directly. Or, Taylor explains, decoding each muscle's activation level might be the most efficient strategy when using the brain to control an implanted stimulation system that activates paralyzed muscles to restore arm motion.

Depending on where the electrodes are implanted in the brain's information-processing stream, BCI researchers can decode the goal of the movement itself, the trajectory in 3D space that a hand follows during a movement, the angular motion of the joints, or even the force in each muscle. The type of device that the BCI system controls will impact which aspect of movement to decode and where it would be best to acquire data from the brain's processing stream.

What is clear from Taylor's research and other BCI studies is that users can improve their device-movement skills with practice. "The brain is an adaptable learning machine," says Taylor. "We learn to do a new dance move or play tennis through practice. Learning to control the movements of a device directly with the brain is no different."

Taylor says such learning can be accelerated by using smart algorithms that learn in parallel with the brain. Taylor's latest algorithm, which she calls "co-adaptive," is designed to decode muscle activation levels from the brain to restore arm and hand function via implanted stimulators. The algorithm, which must be set up in an initial supervised training phase where it is clear what the patient is trying to accomplish, is designed to modify itself on the basis of how accurate recent past movements were. Taylor likens this update process to how supervised learning occurs in neural-network algorithms.



Neural activity from a 16-channel electrode array for a BCI system. Each spike indicates a firing neuron. The software determines which neuron generates each detected spike by attributing each spike to the neuron whose wave shape is most similar to its own.

However, unlike most neural-network applications, the algorithm's input (that is, the brain activity) is also learning in parallel to the decoding algorithm. "This co-adaptation process can be powerful in that adjusting the decoding algorithm allows the person to explore new and potentially easier ways to think about moving that will generate stronger brain activity patterns and ultimately result in more robust decoding of the appropriate muscle activity," explains Taylor. "In any parallel adapting system, finding an appropriate adaptation rate is key to ensuring stable rapid improvement."

While decoded brain activity has been used to activate muscle stimulators in a few studies and the groundwork is now being laid for U.S. Food and Drug Administration endorsement, suggesting that the techniques and technologies associated with BCI systems are finally maturing, Taylor notes there is still a long way to go. She cites two main technical challenges, in particular, as bottlenecks for further progress. One is that researchers working in this area need better brain-recording technologies to extract high-resolution brain activity reliably. The other is the actual physical components that go into a complete BCI system must be made smaller and more portable, and must also be simple enough to use so they can be adjusted without the aid of a research engineer.

"The brain is an adaptable learning machine," says Dawn Taylor. "We learn to do a new dance move or play tennis through practice. Learning to control the movements of a device directly with the brain is no different."

Platform Standardization

Another major issue facing the BCI field as a whole is the lack of platform standardization and interoperability. One scientist who has been working to address this issue is Gerwin Schalk, a researcher at the Wadsworth Center in Albany, NY, and director of the BCI2000 project. Schalk says he became interested in BCI technologies in the late 1990s, when most BCI teams were writing custom software because there was no common platform to facilitate BCI implementations. In 1999, Schalk and his colleagues began working to develop such a platform, which they named BCI2000.

At present, the software is being used by some 600 laboratories around the world. "BCI2000 has supported experiments in about 150 peer-reviewed publications or theses," says Schalk. "Some of these papers are among the most influential studies in the field of BCI research."

At its core, BCI2000 is a general-purpose software system designed to support many data-acquisition and feedback formats through an interface that allows for interaction with external programs. For example, a robotic arm application that is external to BCI2000 can be controlled in real time with brain signals processed by BCI2000. As another example, BCI2000 can be used to store behavioral-based inputs such as eye-tracker coordinates. The software is free, and is available with full source code. While the application is currently Windows-based, Schalk says he and his team are developing BCI2000 code for other operating systems, including Linux and Mac.

With labs regularly developing both new sensor technologies and new methods for interacting with the brain's data stream, work on BCI2000 remains ongoing. "We are working hard to prepare BCI2000 for the growing complexity of the experiments in BCI research, for the increasing number of clinical applications of BCI technology, and for new applications of BCI technology in areas other than communication and control, such as clinical diagnosis," says Schalk, noting that he and his team are committed to keeping the software evolving to the latest developments in BCI research.

The biggest challenge at this point in the BCI community, says Schalk, is the lack of a noninvasive sensor that can robustly measure brain signals at high fidelity. For noninvasive options, Schalk says he expects this sensor issue to remain a substantial problem for the foreseeable future. But he says he remains optimistic. "Without the limitations associated with current sensors, and with a more complete understanding of brain function, it may be possible to create brain-computer interfaces that seamlessly connect our nervous system with machines," says Schalk. "We are far from this goal, but many in the research

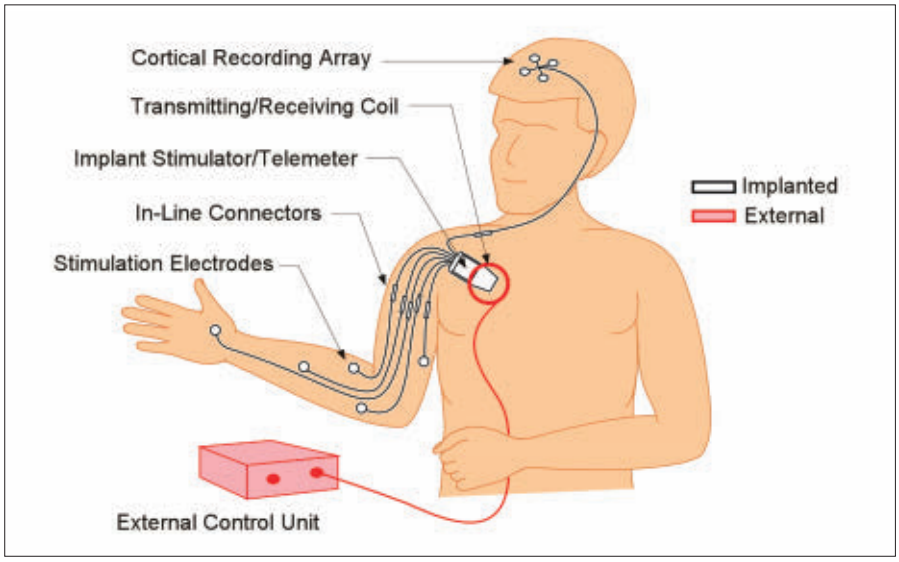


Diagram showing a brain-computer interface setup with implanted and external components.

community, myself included, dream about this possibility every day."

Such seamless interactions might not be achieved for many years to come, but Schalk says it is realistic to see impressive demonstrations of functional restoration, such as full control of robotic arms with actuation of individual fingers, in the near term. Despite the challenges, the potential benefits to ongoing work in this area remain clear. "Using conventional methods, our brain's normal input and output pathways limit the amount of information that can be communicated to about 50 bits per second, such as for spoken speech," says Schalk. "Using BCI technology, it may be possible to substantially increase this rate of communication, and thus allow for an ideal symbiosis of the human brain and artificial machines."

Taylor also offers a similar perspective on the future of BCI technology, suggesting that more adaptive systems will be emerging in the near term. "When you have a less-than-perfect means of controlling your computer, it is imperative that your computer interface make the most of the limited information coming in from the user," she says. In this sense, necessity may lead to significant interface improvements designed to be more intelligent and dynamic in responding to human input.

The prospect of more intuitive input mechanisms has been a key motivation for interface designers, who have long expressed excitement about the potential for BCI research to change the way humans interact with computers. Still, most BCI scientists are careful to offer only guarded optimism, cautioning that the field is still relatively young in its methods and results, making it unlikely the average person will be using BCIs to interact with computers in the near term. At this point, noninvasive ways of recording brain activity provide only low-fidelity data. But for somebody who is paralyzed, even gaining a slow method of interacting with a computer can make a significant improvement in quality of life.

The biggest challenge in the BCI community, says Gerwin Schalk, is the lack of a noninvasive sensor that can robustly measure brain signals at high fidelity.

"In a truly ideal future, BCI technology will make it so that you wouldn't be able to tell if someone has a physical disability," says Taylor. "Paralyzed individuals would move and interact with their environment just like everyone else."

Further Reading

Berger, T.W., Chapi, J.K., Gerhardt, G.A., McFarland, D.J., Principe, J.C., Soussou, W.V., Taylor, D.M., and Tresco, P.A.
Brain-Computer Interfaces: An International Assessment of Research and Development Trends.
Springer, New York, NY, 2008.

Schalk, G.
Brain-computer symbiosis, *Journal of Neural Engineering 5*, 1, March 2008.

Schalk, G. and Mellinger, J.
A Practical Guide to Brain Computer Interfacing with BCI2000.
Springer-Verlag, London, U.K., 2010.

Taylor, D.M., Helms Tillery, S.I., and Schwartz, A.B.
Direct cortical control of 3D neuroprosthetic devices, *Science 296*, 5574, June 2002.

Wolpaw, J.R., Birbaumer, N., McFarland, D.J., Pfurtscheller, G., and Vaughan T.M.
Brain-computer interfaces for communication and control, *Clinical Neurophysiology 113*, 6, June 2002.

Based in Los Angeles, **Kirk L. Kroeker** is a freelance editor and writer specializing in science and technology.

Technology

Skin-like Electronic Patch Unveiled

In an emerging field called epidermal electronics, a multidisciplinary team of University of Illinois researchers has crafted a skin-like, wearable electronic patch that could transform such diverse fields as medical monitoring, wound treatment, covert communication, and human-computer interfaces.

The researchers demonstrated a system that integrates tiny sensors, transistors, radio-frequency inductors, capacitors, and other electronic devices into an ultra-thin, flexible artificial substrate with physical and chemical properties that make the patch barely distinguishable from the user's own skin. The work was first reported in the Aug. 12 issue of *Science*.

The practice of affixing electronic sensors and probes to the human body—for monitoring heart activity, for example—dates back decades. But it has relied on bulk electrodes attached by adhesive tapes, conducting gels, and mechanical fasteners, connected by wires to external boxes of computing and power devices.

A key to making a comfortable and durable wearable alternative was to fashion a skin-like substance that is so thin and so flexible—it stretches, compresses, and resists punctures—that it tightly conforms to the tiny bumps and craters on the skin's surface. The patch developed by the scientists, which can be as thin as 1 micron resting on a 30-micron elastomer substrate, adheres closely to the skin solely via van der Waals forces, those that bind surfaces at the molecular level.

In a demonstration, the researchers showed it was possible for their device, attached to the throat, to identify muscle activity with sufficient accuracy to form the basis of a speech recognition system, possibly then controlling computer games, PCs, or surreptitious communication devices.

"We've figured out a way to configure silicon electronics—conventionally built on the rigid, brittle surfaces of silicon wafers—into formats that are soft, curvilinear, and stretchy," says John Rogers, a professor of materials science and chemistry at the University of Illinois and leader of the project. "The outcome blurs the distinction between electronics and biology and opens up new modes for integration, with significant potential benefits to human health."

—Gary Anthes

Technology | DOI:10.1145/2001269.2001276 Tom Geller

Seeing Is Not Enough

A new DARPA program is teaching cameras visual intelligence—how to spot and understand human behavior.

A man waits by the public square of a war-torn city. A woman walks by and hands him something, then quickly walks away. He waits there another 15 minutes before someone else gives him another package and sprints off. The subject hurriedly puts together the two pieces, walks to the center of the square, puts down the new assemblage, and leaves.

What are the pieces? What did they compose? Why did the man leave after putting them together? The answers to these questions could presage something horrible—like a bomb explosion—or something ordinary, like a city worker installing signage. A human observer would know to follow up, perhaps by examining the placed object or detaining the man who put it there. But for a surveillance camera, these actions are no more suspicious than those of an ice cream vendor or of kids playing soccer. The camera sees, but it does not understand.

A program from the U.S. governmental agency Defense Advanced Research Projects Agency (DARPA) aims to change that with its Mind's Eye program, which first sought participants in March 2010, launched that September, and announced its 15 contractors in January 2011. The five-year program provides funding for 12 research teams to develop "fundamental machine-based visual intelligence" (VI), as well as three implementation teams that will integrate VI technologies with portable, camera-bearing, combat-ready unmanned ground vehicles. Funding for the first year totals about $5 million, or an average of about $333,000 per contractor. It will be followed by $10 million the next year and $16 million the next, with further funding to be determined as the program progresses. To ensure the final products have military usefulness, DARPA has engaged the Army Research Laboratory as its "customer" throughout the process.

The Mind's Eye project has the ambitious goal of teaching cameras to recognize and name the "nouns" and "verbs" of actions, such as one person giving an object to another person.

IMAGE COURTESY OF DARPA

According to DARPA Program Manager James Donlon, action recognition research has traditionally focused on narrowly defined problems, solved with incrementally higher degrees of performance. By comparison, the Mind's Eye project aims to markedly advance the field by converting video streams to simple descriptions of the actions they depict. "DARPA's in the perfect role to identify problems that are almost ridiculously difficult, compared to the current state of the art," Donlon says. "We know that performance will be lower than you'd have on a set of tightly controlled data. But then the questions are, What did we learn as a result? What needs to be developed next to get better performance?"

Seeing Things as They Are

The Mind's Eye project requires researchers to attack four tasks: Recognition of actions in a scene; description of the actions being performed; gap-filling to make accurate assumptions of what's left out of a scene, including predictions of what came before and what will follow after it; and anomaly detection to identify actions that are unusual in the context of the entire video. It builds on past achievements in object recognition—the "nouns" of VI—to establish methods to recognize the "verbs" of action. To focus efforts, DARPA has identified 48 specific verbs of interest such as "approach," "fly," and "walk."

Action recognition is a surprisingly difficult task for a VI system, although humans do it without thinking. First, the system must separate active objects from the background—a task the world makes difficult with such distractions as tree branches blowing in the breeze. Even after a VI system discounts irrelevant movement, the scene could contain multiple active objects that require examination, and the crucial action could depend on one, some, or all of them. As Lauren Barghout, founder of the vision technology firm Eyegorithm, describes it, "You can refer to something as a group of objects or nouns—'two cupcakes'. If you employ the 'spotlight theory of attention,' you pay attention to the area they occupy, but you might find that the center of that area is just an empty space. Or you could follow the 'object-based' theory, in which case you have to determine whether the 'object' is one cupcake or both cupcakes."

The visual system must also recognize and deconstruct those objects. Takeo Kanade, a professor at Carnegie

Mellon University's Robotics Institute, points out that some objects are easier to recognize than others. "We can train for limited sets of objects pretty well, once we know their expected appearances—the human face, the human body, cars, things like that. But there are lots of objects whose appearance is completely unknown or unspecifiable. Take 'package'. People who use that word only know that it's three-dimensional, probably box-like, and made of paper. It tends to be from hand-size to body-size, because we call it something else if it's bigger." The human face, on the other hand, is patterned after a well-defined template and is therefore easier for VI systems to identify.

From Object to Actor

Once the VI system recognizes objects with some accuracy, it must understand how their movements and interactions comprise actions. Bruce Draper, an associate professor of computer science at Colorado State University, believes a successful system will need to learn that without explicit training. "You cannot go in and predefine every object, every action, every event," he says. "You don't know where the system's going to be deployed, and the world changes all the time. We need to build systems that learn from watching the video stream without ever being told what's in it." The key, he says, is for the system to recognize repeated actions. "If we had only one video of someone throwing a football, there'd be no repeated pattern to learn from," says Draper. "But with several, it can learn that pattern as unique."

After the VI system can recognize actions, it still has to name them. NASA Jet Propulsion Laboratory Principal Investigator Michael Burl believes that a VI system should do more than just flash "run" and "approach" on the screen when those actions occur. "We want to go beyond text descriptions by extracting and transmitting a 'script' that can be used to regenerate what happened in the video," he says. To that end, Burl and co-investigator Russell Knight are using planning-executing agent (PEA) graphical models to provide an abstract representation of various behaviors. "Take 'throw,'" says Burl. "It takes two arguments: The agent doing the throwing, and the object being thrown. For the action itself you'd expect to see transitions between several states, such as a windup, forward motion of the arm, then the concepts of 'separate' and 'fly' being applied to the object. PEA models are hierarchical so complex actions can be composed from simpler ones. By identifying the PEA models being used by the agents, we obtain a compact, generative script that provides a summary of the full video."

Beyond the Battlefield

DARPA's Donlon says the target verbs were chosen to be both relevant and wide-ranging. "Some verbs are what soldiers would need to know on the battlefield, but the list has a lot of diversity," he notes. In fact, all of the listed verbs have some applicability in non-military situations, raising the question: What will VI technology be like when it reaches the civilian sphere?

"We're getting a lot of interest in non-military applications" says Draper. "For example, the National Institutes of Health wants to figure out what kids are doing on the playground. Not what any individual child is doing, but which pieces of equipment are encouraging them to be active. And there's an existing surveillance market that detects motion and determines whether it's caused by a person. That's wonderful to protect a perimeter, but what if someone grabs their chest and falls down in the middle of a public square? That's an activity that's wrong, not just a matter of 'someone who shouldn't be there.' "

However, the presence of intelligent cameras in the public sector could have a chilling effect, says Jay Stanley, senior policy analyst in the Speech, Privacy and Technology Program of the American Civil Liberties Union. "It's fine to use this technology in military applications on overseas battlefields or in certain law enforcement situations where there's probable cause and a warrant," he says. "And perhaps it could be used by individuals if it becomes integrated into consumer products, the way face recognition has in a limited way. But there are two classes of concerns. The first is that it works really well, and so intensifies existing concerns about surveillance. The second is that it works very poorly. False alarms can be just as bad for people as accurate ones, and there are all kinds of gradations in what a false positive is. Perhaps the computer correctly interprets your behavior but there's a perfectly innocent explanation. Or perhaps the computer totally misunderstands your behavior. And there are a lot of points in between."

Donlon believes the benefits of the Mind's Eye project will outweigh such risks. "One of the things inspiring about visual intelligence is that, if we can solve this range of tasks on this range of verbs—even partially—there will be a wide range of both military and commercial applications," he says. "We use one potential military outcome as the ultimate goal, but it's just one exemplar, really, of the benefit we'll get. It just seems intuitively obvious that there's a very rich potential commercial market for smart cameras—for commercial security applications, or for loss prevention in retail. For all of them, attending to alerts and dismissing false alarms is a lot less eyeball-intensive than staring at a video feed 24/7." C

Further Reading

Barghout, L.
Empirical data on the configural architecture of human scene perception, *Journal of Vision 9*, 8, August 5, 2009.

Draper, B.
Early results in micro-action detection, http://www.cs.colostate.edu/~draper/newsite/index.php/research/visual-intelligence-through-latent-geometry-and-selective-guidan/early-micro-action-detection/, Colorado State University video, January 2011.

Lui, Y., Beveridge, J., and Kirby, M.
Action classification on product manifolds, *2010 IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, June 13–18, 2010.

O'Hara,S., Lui, Y., and Draper, B.A.
Unsupervised learning of human expressions, gestures, and actions, *2011 IEEE Conference on Automatic Face and Gesture Recognition*, Santa Barbara, CA, March 21–25, 2011.

Poppe, R.
A survey on vision-based human action recognition, *Image and Vision Computing 28*, 6, June 2010.

Tom Geller is an Oberlin, OH-based science, technology, and business writer.

Society | DOI:10.1145/2001269.2001277 Samuel Greengard

Living in a Digital World

Technology has created new opportunities to connect and interact. Yet, researchers are increasingly concerned that heavy technology usage is changing people's behavior in less than desirable ways.

It is no secret that humans have an innate urge to connect with one another. In fact, research shows that well-adjusted people spend more time engaged in social interaction and activities. However, in the age of always-on digital technology, the notion of connectedness—and the definition of friendship—is changing radically. Increasingly, the route to human interaction is through a digital device.

Approximately two billion people now tap into the Internet. About five billion people use mobile phones and a growing number of these devices offer sophisticated computing and communications capabilities. There's cell service atop Mt. Everest and in remote South Pacific atolls. Incredibly, the average 13- to 17-year-old in the U.S. sends about 110 text messages per day. In fact, it's become increasingly difficult to go anywhere without getting caught in the tractor beam of digital technology.

Not surprisingly, as people use these devices more frequently—and for more hours each day—researchers are studying the effects with growing interest. Add to this the extreme multitasking that we increasingly engage in, either by choice or necessity, and it is clear that society is venturing into a brave new frontier. "We're seeing people so absorbed in digital media that it's becoming their primary reference point for life," observes Clifford Nass, a communications professor at Stanford University and author of *The Man Who Lied to His Laptop: What Machines Teach Us About Human Relationships*.

What is the impact of digital immersion? How is it changing people's thinking and behavior? And how does it affect the way we view the world and interact with others? It's a complex equation that researchers are only now beginning to understand. "Digital technology brings people together," says Michael Suman, research director at the Center for the Digital Future at the University of Southern California. "It allows people to connect in ways that were never before possible. But it also creates new sets of questions and potential problems."

"For many of us, it is becoming increasingly difficult to control the impulse to check the inbox yet again," notes Yair Amichai-Hamburger.

Net Losses

There's no disputing that digital technology has thoroughly invaded our lives. A 2011 study conducted by telecommunications giant Ericsson found that 35% percent of iPhone and Android users check their email or Facebook account before getting out of bed in the morning. In addition, 40% use their phones in bed before they go to sleep at night. The average American is digitally connected between 2.5 and 3.5 hours a day. Nielsen reports that social networking, online games, and email are the biggest attractions.

Few people have examined the topic more closely than Sherry Turkle, a professor of social studies of science and technology at Massachusetts Institute of Technology and author of *Alone Together: Why We Expect More from Technology and Less from Each Other*. She believes that digital immersion is seductive because it seemingly addresses our human vulnerabilities. "As it turns out, we are very vulnerable,"

PHOTOGRAPH BY BILLY QUACH

she says. "We are lonely but fearful of intimacy. Constant connectivity offers the illusion of companionship without the demands of friendship. We can't get enough of each other if we can have each other at a distance and in amounts that we can control."

The heavy use of digital technology trains society to have less patience for the particular skills, pace, and sensitivities of face-to-face interaction. "We become used to the volume and velocity of the digital medium," explains Turkle. "We adapt to it and, over time, become more comfortable with its simplifications." The upshot? "People complain that they are too busy communicating to think, too busy communicating to create, and, in a final paradox, too busy communicating to fully connect with the people who matter. We are in continual contact but we are alone together."

The ripples of digital technology also make it easier to hide. Turkle says many people admit they are relieved to leave a voicemail message rather than reach the intended person. Some say that texting lets them avoid the time commitment of phone calls. "We are using technologies to dial down human contact," she says. "People are comforted by being in touch with a lot of people whom they also keep at bay." The result? "We imagine that email and texting will give us more control over our time and emotional exposure" but, eventually, "anything but staccato texts seems too exhausting."

While digital technology can connect families and friends over geographic distances, it is critical to recognize that Facebook pokes and postings aren't equal to actual conversation.

Master or Slave?

The allure of digital technology impacts people in other ways. Suman says the students he teaches have more trouble than ever focusing on lectures and learning. Text messages and emails arrive in gibberish, he says, and students end up asking the same questions over and over. Even when they have switched off their devices they are too often unable to think through concepts and ideas. "They're increasingly challenged to engage in deep and meaningful thought," he says. "Sequential, logical, rational thinking seems to be severely compromised."

A breakdown of social etiquette—if not outright rudeness—is also more pervasive, Nass says. "Today, people think it's okay to text in the middle of dinner, at a meeting, in class, wherever. They text while you're talking to them and then they look up and say, '*What?*' " Humans, he says, aren't good at multitasking. In fact, studies show that multitasking doesn't exist. We simply switch back and forth from one task to another very quickly. The heaviest "multitaskers" show signs of diminished short-term memory. In other words, they're forgetful.

Yair Amichai-Hamburger, director of the Research Center for Internet Psychology at the Sammy Ofer School of Communications, says it is time to consider whether we are served or enslaved by today's technology. "For many of us," he notes, "it is becoming increasingly difficult to control the impulse to check the inbox yet again." Worse, he says, we're constantly surrounded by advertisements urging us to find fulfillment through the acquisition of material goods. The latest acquisition of a shiny new gadget gives us a quick fix but does nothing to feed the soul.

For many, the net effect is an always-on digital lifestyle. In a world where time is often equated to money, society increasingly buys into the notion that being technologically tethered is essential and even unavoidable. What's more, as employers

Society

Gender Bias at Wikipedia?

A substantial gap in the number of male versus female editors at Wikipedia may be creating a gender-oriented disparity in the popular online information source's content, according to research by the University of Minnesota's College of Science and Engineering.

Only 16% of the new editors who joined Wikipedia in 2009 identified themselves as female and those females made only 9% of the edits by the editors who joined that year.

The findings of Minnesota's GroupLens Research Lab were recently published in a paper, "WP: Clubhouse? An Exploration of Wikipedia's Gender Imbalance," and discussed in a YouTube video, "Research Proves Gender Imbalance on Wikipedia."

"We think people who use Wikipedia as a resource need to be aware our results suggest there are disparities in the quality of its content coverage and those disparities seem to be related to the gender gap," says lead researcher Shyong (Tony) K. Lam. "For example, we found Wikipedia's coverage of movies of particular interest to females tends to be lower in quality than its coverage of movies with large male audiences."

The research gives no indication as to why the gender gap exists. "Our research was entirely quantitative and based on data made publicly available by Wikipedia," Lam adds.

"What surprised us most was this gap hasn't changed in five years. It was 16% five years ago and it's 16% now," says Lam. "This compares with other areas of the Web, especially in other social media like Facebook and Twitter, where the gender gap has not only closed but has become female dominant."

Lam has received acknowledgment from Wikipedia that it received the research results but has gotten "no indication as to whether any action will be taken," he says.

The research will be presented at the 2011 WikiSym conference in Mountain View, CA, in October.

—Paul Hyman

require individuals to check digital devices and respond 24/7, there's no clear separation between home and work. These pressures, Amichai-Hamburger says, put modern society at danger of "swapping standard of living for quality of life."

In fact, studies show that heavy technology use can result in higher levels of loneliness and depression—and the U.S. and other countries are trending upward. Irena Stepanikova, an assistant professor of sociology at the University of South Carolina, examined various digital tools in a 2010 study, "Time on the Internet at Home, Loneliness, and Life Satisfaction." Researchers found that people who spent more time at home browsing the Web and using instant messaging, chat rooms, and newsgroups felt lonelier and less satisfied. Email, on the other hand, neither increased nor decreased mental well-being.

The thing that's easy to overlook, Stepanikova says, is that we frequently use digital tools in isolation as a way to connect with others. While digital technology can connect families and friends over geographic distances, it's critical to recognize that Facebook pokes and postings aren't equal to actual conversation. Too often, "we use the Internet alone, and even if others are present, we do not actively interact with them," she says. Consequently, "the solitude of these activities may counter some of the potential social benefits."

It's no small problem. "The more time people spend at home on the Internet," says Stepanikova, "the less time they spend on social activities, parties, conversation, attending sports and cultural events—and essentially on any activities performed together with family members and with friends."

Switching Off

A growing number of researchers and social scientists believe it is important to take steps to regain control of the technology and our lives. One approach that is gaining popularity is the concept of switching off electronics and taking clearly defined breaks. In 2010, the Sabbath Manifesto project emerged. It promotes the idea of unplugging every seventh day, "deadlines and paperwork be damned" and creating more defined boundaries in order to avoid "destroying the fabric of your life."

Amichai-Hamburger says that being unplugged at least one day a week "gives you a chance to be with those you care about." No less important: It changes the flow of life and provides perspective about what's really important. "It reminds us that we have to lead technology and not be led by it. It gives us space to think."

Turkle also believes that venturing offline can be refreshing. "It can be a reminder of the importance of solitude that refreshes and sustains." Nevertheless, she believes that "unplugging" is not the way of the future and that we've only begun the process of adapting to digital technology and learning how to manage our actions and reactions effectively.

"We must begin to make corrections," says Turkle. "We are not doing justice to the complexity of the problems we face, just as we are not doing justice to each other. We need to learn how to be on a digital diet so that we can make healthy choices about the kind of life we want to lead, the kind of life that will make us productive, and how we can be content and fulfilled individually and in relationships." C

Further Reading

Amichai-Hamburger, Y. and Hayat, Z.
The impact of the Internet on the social lives of users: A representative sample, *Computers in Human Behavior 27*, 1, January 2011.

Amichai-Hamburger, Y. and Vinitzky, G.
Social network use and personality, *Computers in Human Behavior 26*, 6, November 2010.

Nie, N.H., and Hillygus, D.S.
Where does Internet time come from? *IT & Society 1*, 2, Fall 2002.

Stepanikova, I., Nie, N.H., and Xiaobin, H.
Time on the Internet at home, loneliness, and life satisfaction: Evidence from panel time-diary data, *Computers in Human Behavior 26*, 3, May 2010.

Turkle, S.
Authenticity in the age of digital companions, *Interaction Studies 8*, 3, 2007.

Samuel Greengard is an author and journalist based in West Linn, OR.

Milestones

CS Awards

Microsoft Research, ACM SIGPLAN, IEEE, and Women Entrepreneurs in Science & Technology (WEST) recently honored a select set of computer scientists for their innovative research and leadership.

MICROSOFT RESEARCH FACULTY FELLOWS

Microsoft Research recognized eight new faculty members from Austria, Australia, Brazil, and the U.S. as 2011 Faculty Fellows. They are Maria Florina Balcan, Georgia Institute of Technology; Krishnendu Chatterjee, IST Austria; Jure Leskovec, Stanford University; Alistair McEwan, The University of Sydney; Shwetak Patel, University of Washington; Anderson de Rezende Rocha, University of Campinas; Keith Noah Snavely, Cornell University; and Brent Waters, University of Texas.

SIGPLAN SOFTWARE AWARD

ACM SIGPLAN awarded the 2011 SIGPLAN Programming Languages Software Award to Simon Peyton-Jones and Simon Marlow of Microsoft Research, Cambridge, for their authorship of the Glasgow Haskell Compiler, the preeminent lazy functional programming system. Peyton-Jones and Marlow donated their $2,500 prize to Haskell.org.

EMANUEL R. PIORE AWARD

IEEE presented the Emanuel R. Piore Award to Fred B. Schneider, Samuel B. Eckert Professor of Computer Science at Cornell University, for "contributions to trustworthy computing through novel approaches to security, fault tolerance, and formal methods for concurrent and distributed systems."

WEST LEADERS

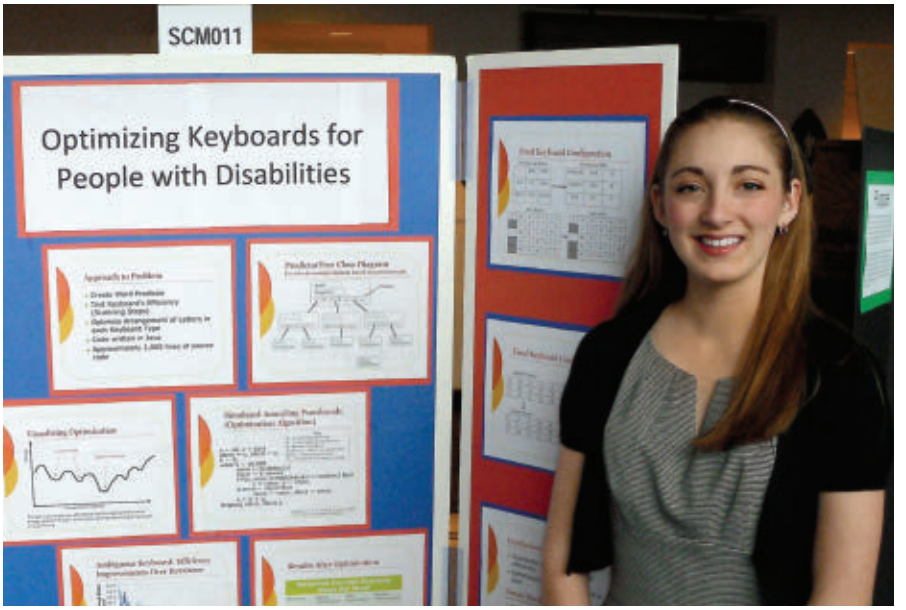
WEST celebrated its 2011 Leadership Awards at the Microsoft New England Research & Development (NERD) Center, and presented Leadership Awards to Lydia Villa-Komaroff, chief science officer for Cytonome; Jennifer Tour Chayes, distinguished scientist and managing director of Microsoft's NERD Center; Laura Fitton, founder and CEO of OneForty.com; and Joanna Horobin, CEO of Syndax Pharmaceuticals.

—Jack Rosenberger

Milestones | DOI:10.1145/2001269.2001278 Marina Krakovsky

Success at 16

A high school student wins first prize from ACM for developing a faster keyboard layout.



PHOTOGRAPH COURTESY OF NATALIE NASH

USING A SOPHISTICATED optimization algorithm, Natalie Nash, a 16-year-old high school junior from Pennsylvania, designed a software interface to help severely disabled users communicate significantly faster than is possible using currently available alternatives. In May, her project, which was Nash's entry in the Intel Science and Engineering competition, wowed the three judges representing ACM, who awarded her the Association's first prize.

"The project was both socially relevant—it was clear that it meant something to her—and it had some very impressive computer science pieces to it," says one of the judges, Robb Cutler, founding president of the Computer Science Teachers Association and a graduate student in computer science at Purdue University.

People who can't speak use augmentative and alternative communication (AAC) devices that generate speech from users' input. Those who can't type, such as people with cerebral palsy, use AAC devices with a single switch through which they painstakingly make choices via an onscreen keyboard. The computer scans the AAC keyboard, highlighting one row at a time until the user activates the switch (by pressing a button, puffing, or blinking), and the device slowly zeroes in on the desired letter. "Sometimes it will take a good five minutes to type one sentence, and by that time everybody else in the conversation has moved on to another topic," Nash explains. "My project was designed to improve the speed of communication using this single-button method."

The first question Nash tackled was the fastest style of keyboard: the standard QWERTY keyboard; an AAC keyboard, which arranges the 26 letters in a 5x5 grid; or a standard ambiguous keyboard, which has more than one letter per key. Nash wrote a program to compare the three keyboards' efficiency at processing the 20,000 most commonly spoken words in American English. Contrary to her hunch, the AAC keyboard was fastest even though the ambiguous keyboard requires traversing only nine keys instead of 26.

Knowing that the AAC keyboard reduces scanning steps by placing the most commonly used letters, such as A and E, at the top left, Nash suspected she could reduce scanning steps by rearranging the letters on an ambiguous keyboard. What's more, all the devices use word prediction—guessing the entire word from partial input—which gave Nash an insight that helped her optimize the AAC keyboard. "It's not necessarily the most common letters in the English language [that should be near the front], but the most common letters in the first part of the most common words, because rarely does the user have to type the actual whole word." G, for example, is very common because many words end in -ing, but users never have to enter the final g.

An exhaustive search of all possible keyboard arrangements would take forever, so Nash used the simulated annealing algorithm to find the optimal keyboard layout. "It actually turns out that I was correct. The ambiguous keyboard, when it was optimized, turned out to be a lot better than the other ones," says Nash, who calculated it is about 15% faster than an AAC keyboard.

Always interested in science, Nash got her first taste of computer programming as a third-grader participating in Carnegie Mellon University's (CMU's) C-MITES program for talented students. Soon enough, she was learning from her father, a CMU-educated programmer. By the time Nash could take computer science classes in high school, when she was a sophomore at Vincentian Academy, she was surprised by how much she already knew.

Although users would have to learn a new keyboard layout, an acquaintance with cerebral palsy told Nash the AAC community would embrace the new design. "Their main struggle is to be faster," says Nash, "and if it means memorizing a new keyboard, they would do it." C

Based in San Francisco, **Marina Krakovsky** is the co-author of *Secrets of the Moneylab: How Behavioral Economics Can Improve Your Business.*

DOI:10.1145/2001269.2001279

Michael A. Cusumano

Technology Strategy and Management

The Platform Leader's Dilemma

Studying the lessons learned from past and present platform leaders.

Platform leaders are companies that do not just sell standalone products. They have a foundation technology that is sufficiently open so that outside firms can provide complementary products and services, ranging from prerecorded videotapes to software applications and downloadable digital content. The value of the platform and complements can grow exponentially with positive feedback loops. These "network effects" make the platform, and the complements, increasingly valuable (and profitable) as more users, application developers, service providers, content providers, device makers, and other ecosystem players such as advertisers adopt the same platform.

While "hit" products come and go, platforms supported by a global ecosystem of complementors and strong network effects should be more difficult for competitors to dislodge. But, as Annabelle Gawer and I noted in our book *Platform Leadership*, even the

ILLUSTRATION BY MICHAEL AUSTIN

best firms face a common dilemma best described by Clay Christensen in his book *The Innovator's Dilemma*: Their success makes it difficult to change, even though their technology must evolve or become obsolete.[a] As new platform wars emerge around mobile phones, video games, cloud computing, and social networking, it seems like a good time to reflect on some past examples of platform leadership and draw some general lessons. In this column, I begin with a few companies we all should know well: IBM, JVC, Sony, Google, and Nokia. Then I turn to Microsoft and Apple.[b]

IBM

This venerable company created the first global platform in the modern computer era, based on the IBM/360 mainframe software introduced in the mid-1960s. Antitrust initiatives pressured IBM to release information to independent maintenance providers. This eventually led to an ecosystem of hardware "clone" makers like Amdahl and Fujitsu as well as software product and service companies focused on IBM customers. But IBM had the deepest knowledge of its market. It had sold primitive electronic computers since the early 1950s and for decades before that dominated in electromechanical tabulating machines and other office equipment. In the 2000s, IBM still rules the mainframe world and does pioneering work in high-performance computing (consider the Watson computer system that beat the "Jeopardy!" game champions). But enterprise computing has evolved to a much more heterogeneous world of machines and software of different shapes and sizes.

To their credit, by 1980, a few key IBM executives had realized that a platform shift was occurring and they introduced their own personal computer design in 1981. The operating system and microprocessor turned out to be the two key components of this new PC platform, and IBM ceded control over these elements to its suppliers, Microsoft and Intel. To its credit again, though, after absorbing billions of dollars in losses, IBM found a way forward. Under new CEO Louis Gerstner, hired from RJR Nabisco in 1993, it became the champion of "open systems" (Linux, Java, the Internet, ubiquitous computing, and the cloud). Gerstner and his successors also sold off commodity hardware businesses and rebuilt the company around services and middleware that help customers utilize different technologies.

Google has challenged the modus operandi of the computer industry—proprietary technology.

The lesson here? Platforms inevitably evolve and the leader of one generation may lose control over the next. But all is not lost if the erstwhile leader truly has distinctive capabilities that keep it linked to its customers. In this case, IBM had decades of experience that helped it understand—like no other company—the data-processing needs of enterprise users and other large organizations. This is where the firm kept its focus. The shift in platforms away from the mainframe and the loss of control over the PC were both highly damaging financially. But these changes created a new beginning for a service-oriented IBM.

JVC and Sony

In the 1970s, VCRs became the highest volume consumer electronics product as everyone with a television set became a potential customer. Sony actually won the race to create a viable home-video recorder but JVC ended up as the market winner. Several Japanese firms studied Ampex's technology for broadcasters in the late 1950s, and both JVC and Sony found ways to miniaturize and improve the technology for the mass market. They beat out their rivals in Japan, the U.S., and Europe. Sony introduced the Betamax in 1975 and JVC countered with the VHS in 1976. By 1978, however, VHS had passed Betamax in sales. It became a global platform in that JVC licensed the VHS technology widely, allowed other companies like RCA and GE to influence feature development (mainly recording time), and cultivated a large set of outside firms for video content and distribution. Sony may have built a better product and was first to market, but it did not do as good a job cultivating ecosystem partners. JVC went on to become a multibillion-dollar company, based mainly on the VHS platform. It must also be noted that, while it diversified from audio and video to computer storage products, JVC never dominated another market. Today it survives after merging with another Japanese audio equipment producer, Kenwood.

Compared to JVC, Sony always has had broader technical skills and deeper pockets. After the Betamax experience, it also learned to cooperate better with other firms when it came to digital video standards, the PlayStation platform for video games, and its Blu-ray format for DVDs. But both Sony and JVC failed to grasp how new software and networking technologies were changing the world of consumer electronics. The Walkman, for example, introduced in 1979, brought enormous revenues to Sony while the Betamax gradually faded to zero market share (for consumers) by 1989. Two decades later, Sony had the technical skills as well as the content to evolve the Walkman into a new type of platform, like Steve Jobs did with Apple's iPod and then the iPhone. But Sony did not have the visionary capabilities of Apple and chose instead to focus mainly on standalone hardware products.

The lesson here? Platform leaders must prepare for the future, even when they are focused and highly successful with their present businesses. This means creating a flexible and

a See A. Gawer and M. Cusumano, *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation* (Harvard Business School Press, Boston, 2002). This column also plays off the dilemma noted by Clayton Christensen, who observed that established companies may pay too much attention to their established customers and fail to see challengers emerging with initially inferior technology. See C. Christenson, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail* (Harvard Business School Press, Boston, 1997).

b Some of the cases discussed in this column are based on material in M. Cusumano, *Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Unpredictable World* (Oxford University Press, 2010).

creative organization, with people who can learn about new technologies and markets. Companies also need to nurture deep, enduring skills, such as by understanding how to marry high technology with consumer electronics. JVC could have done better after the VCR era had it evolved its skills more quickly from analog to digital technology, and then to networked systems and hardware driven by software, rather than software driven by hardware. Sony faced the same challenges and did slightly better but not good enough. Though it still makes Walkman multimedia devices as well as PCs, smartphones, and video game consoles, and owns its own music label and movie studio, Sony continues to look for hit hardware products and always seems to find itself trailing in the newer platform markets.

Google

Google's platform was initially the Internet, based on a better search engine. But Google made its technology nearly ubiquitous on PC desktops with the downloadable and free toolbar. It then built an Internet portal, replete with email, maps, applications, storage, and other features, to surround and feed the search engine. Google monetizes its leadership position by selling targeted ads that accompany searches. But Google has not stopped there. The company realized years ago that most computing would one day be on mobile devices. So Google bought and then refined the Android operating system (which is based on Linux) and created the Chrome browser to facilitate mobile computing (and mobile searches as well as advertising). Google is now the largest smartphone OS provider and plans to challenge Apple directly by acquiring Motorola's mobile phone business. But not even Google has done everything right. It was slow to see the importance of social networking. It has been trying for years (with limited success) to challenge Facebook and create a coalition of partners to gain access to more social networking and social media content—again, presumably, to sell more search and advertising.

The lesson here? Again, platform leaders must force themselves to think broadly about their platforms and business models while extending their technical and marketing capabilities. Google has always focused on search, but computing has been moving beyond the desktop for years and even beyond the Internet—to multiple devices as well as applications and content that reside within both open (such as the Internet) and closed (such as Facebook) networks. Moreover, Google has challenged the modus operandi of the computer industry—proprietary technology. Its software platform for mobile phones and other devices such as Netbooks and tablets is both free and open. It is difficult for companies that charge for their technology and do not have large advertising income—like Apple and Microsoft, as well as Nokia—to beat free and open.

Nokia

This Finnish company remains the largest producer of cellphones, and its Symbian software has been a dominant platform for basic handsets. However, mobile sales are quickly moving to smartphones that require more sophisticated software. Not surprisingly, Nokia has seen its market share, market value, and financial performance suffer dramatically as RIM's BlackBerry and Apple's iPhone handsets, and a variety of devices from different companies running Google's Android software, have come to dominate the market. Nokia removed its CEO and is now led by a former Microsoft executive, Steven Elop, who recently announced plans to abandon the Symbian operating system as well as another joint OS project with Intel. Instead, Elop wants to use Microsoft's Windows phone software for Nokia's next generation of smartphones.

The lesson here? Once more, we see that platform leaders must be prepared to evolve and even discard their technologies and sometimes their business models as well. If they fail to develop new technology internally or find suitable acquisitions, they may well find themselves adopting the platform technology of a competitor. We shall see what happens to Nokia, but the future does not look very bright.

Microsoft and Apple

Steve Ballmer, Microsoft CEO since Bill Gates handed over the reins in 2000, is often criticized for not being able to

Calendar of Events

October 16–21
International conference on Model Driven Engineering Languages and Systems,
Wellington, New Zealand,
Sponsored: SIGSOFT,
Contact: Thomas Kuehne,
Email: tk@ecs.vuw.ac.nz

October 16–19
The 24th Annual ACM Symposium on Interface Software Technology,
Santa Barbara, CA,
Sponsored: SIGGRAPH, SIGCHI,
Contact: Jeff Pierce,
Email: drjpierce@mac.com

October 17–19
Symposium on Algorithmic Game Theory,
Amalfi, Italy,
Contact: Giuseppe Persiano,
Email: giuper@dia.unisa.it

October 17–21
The ACM Conference on Computer and Communications Security,
Chicago, IL,
Sponsored: SIGSAC,
Contact: Yan Chen,
Email: ychen@northwestern.edu

October 19–21
Creativity and Innovation in Design,
Eindhoven, Norway,
Contact: Martens Jean-Bernard,
Email: j.b.o.s.martens@tue.nl

October 19–22
Research in Applied Computation Symposium,
Miami, FL,
Sponsored: SIGAPP,
Contact: Jiman Hong,
Email: jiman@ssu.ac.kr

October 19–22
ACM Special Interest Group for Information Technology Education Conference,
West Point, NY,
Sponsored: SIGITE,
Contact: Edward J. Sobiesk,
Email: edward.sobiesk@us.army.mil

October 23–26
ACM SIGOPS 23rd Symposium on Operating Systems Principles,
Cascais, Portugal,
Sponsored: SIGOPS,
Contact: Edward Pearce Wobber,
Email: wobber@microsoft.com

move much beyond the PC platform. Indeed, Windows desktop and server and the Office suite still account for nearly 80% of Microsoft's revenues and almost all its profits. Ballmer is under particular pressure because Microsoft's share price is lower today than it was a decade ago (though this is also true of Intel, Cisco, Nokia, and a host of other high-tech firms). And archrival Apple, despite the small (but rising) global market share of the Macintosh personal computer, and despite its near bankruptcy only a few years ago, has been growing at 50% a year and vaulted past Microsoft in market value during 2010. Apple is growing so fast because it has become a major player in consumer electronics as well as cellphones and digital content distribution. On the strength of its high-margin digital service platforms (iTunes, App Store, and iCloud), Apple may someday match or surpass Microsoft in profitability. Reproducing digital bits is much less costly than reproducing hardware boxes.

But we tend not to give Microsoft enough credit for its accomplishments. It remains the most profitable of the high-tech giants, including Apple and Google. It has survived radically disruptive technological transitions and daunting business-model challenges (character-based to graphical computing, the Internet, the Software as a Service model, cloud computing, mobile computing, and social networking). It has survived antitrust scrutiny and violations (remember Netscape?). Nevertheless, Microsoft continues to "print money," relying on the enormously profitable gross margins of the packaged software business. And change is always in the works at Microsoft, albeit slowly. Billions of dollars in losses ("investment") from MSN and Bing over the past 15 years has prepared Microsoft for the online world funded by advertising revenue. It learned from the Longhorn/Vista debacle how to break up Windows into smaller, more manageable chunks, which can also help deliver new Internet and cloud services. Its Windows Azure cloud offering and SaaS versions of some products have had good receptions in the marketplace and are competitive, though not dominant.

What Microsoft needs to do is what IBM, Google, and Apple have done—evolve beyond the slow-growing PC industry and move to newer, faster-growing markets, as well as integrate the new technologies. Microsoft's decision in early 2011 to buy Skype is one move, although an expensive one, to get access to new customers. Other moves include Microsoft's alliance with Nokia to take over its future smartphone software and an earlier alliance with RIM to take over the search business on the BlackBerry smartphones. An even bolder move would be for Microsoft to convince RIM to replace its aging operating system with Windows and combine this business with Nokia's smartphones.

The lesson here? Platform leadership can be both a blessing and a curse. Gates' major mistake (back in the late 1990s) was probably to insist that Microsoft remain a Windows company, rather than become a broader platform company. Microsoft engineers tried to force Windows onto the new platforms, the Internet, and then mobile phones, rather than create optimized software from scratch and then link the new platforms back to Windows. Microsoft also cut down Windows for the Xbox, but did not retain Windows compatibility. Of course, Windows on the desktop is the modern-day equivalent of a gold mine. It is not difficult to understand why Gates and Ballmer have been so reluctant to cannibalize this business. Apple, by contrast, was never wedded to the original Mac platform, which failed as a business in the 1980s and 1990s anyway. It later replaced the first Mac OS with NeXT software, which was based on Unix. But Apple did remain wedded to its remarkable capabilities in user interface design and visionary product innovation. Those skills are the basis for Apple's business success with the iPod, iPhone, iTunes, and iPad and its remarkable transformation into a global platform leader on multiple integrated devices. But we shall need a few years to see how well Apple copes with it own platform leader's dilemma. C

Michael A. Cusumano (cusumano@mit.edu) is a professor at the MIT Sloan School of Management and School of Engineering and author of *Staying Power: Six Enduring Principles for Managing Strategy and Innovation in an Unpredictable World* (Oxford University Press, 2010).

DOI:10.1145/2001269.2001282

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

Kode Vicious
File-System Litter

Cleaning up your storage space quickly and efficiently.

Dear KV,

We recently ran out of storage space on a very large file server—one with many terabytes of space—and upon closer inspection we found that it was just one employee who had used it all up. The space was taken up almost exclusively by small files that were the result of running some data-analysis scripts. These files were completely unnecessary after they had been read once. The code that generated the files had no good way of cleaning them up once they had been created; it just went on believing that storage was infinite. Now we've had to put quotas on our file servers and, of course, deal with weekly cries for more disk space. Surely there is a better way of dealing with this problem than clamping down on everyone for fear that one of them will do the wrong thing.

Caught Between a Block and a Lack of Space

Dear Caught,

Yes, there are better ways of handling this problem. You have now discovered one of the drawbacks of cheap storage (and yes, that old adage is true): files will always expand to fill the available storage space, just as programs expand to fill all available memory and spawn more threads until all of your CPU is utilized as well.

Shared storage, such as you are dealing with, presents the thorniest problem because it is *shared*, and, it would seem—as regular readers of this column are, I'm sure, aware—people simply cannot be trusted to police themselves. In reality most people can, but it takes just one, as you found out, to "ruin it for everybody," as our teachers used to say.

The point you make about the scripts not having a way of cleaning up after themselves is a good one. When you build programs out of many small source files your tools also generate intermediate files—the objects that then get linked into a final executable. All build systems worthy of the name, however, have some form of "clean" target. Although this target was originally created so that you could start a new build from scratch, it is also a handy way of shrinking down the size of your work area when a project is either complete or on hold. Having a program that would do the same work with intermediate data files is a good start, but there are other things that can be done to improve the situation.

Littering the file system with files that have to be deleted later results in a performance problem. If you need to find all the files via recursive descent of the file system before you can delete them, then you are going to be hammering your file system. In the case of NFS (network file system)-mounted systems, you will also be hammering your network while trying to clean up after yourself. Although it might appear that the best course of action would be to delete the files immediately after use, this would prevent you from debugging problems in your data analysis. Also, if you have to rerun some part of the analysis, then the derived objects you created could come in handy in speeding up the second, or third, or—well, you know—the *n*th run before you finally get it right. Probably the best compromise position is to place all of the derived objects into their own directory or set of directories, which can be easily located and purged when it is time to free up some space on the file system.

Keeping all the files in one place means you do not have to descend the file system recursively to find all the files that can be safely deleted. That will make the process easier, faster, and therefore more likely to be used by

the people on your system. If cleaning up after yourself takes 30 *seconds*, you are pretty likely to do it; if it requires 30 *minutes*, you are going to put it off as long as you can, usually long enough for the file system to fill up again.

KV

Dear KV,

You have written in previous columns about not using `printf` to debug programs, and you recommended using a debugger, but you must admit that there are times when a `print` statement is just an easier way of debugging a program and that using a debugger is overkill.

Still Pounding on Printf

Dear Pounding,

True, I have written in previous columns about the reasons for not using `print` statements for debugging, and I have recommended that people use finer tools such as debuggers to find problems in their programs. There are two instances in which I agree that a `print` statement is a better solution.

The first instance where `print` beats a debugger is when either you have no debugger or the debugger itself is incredibly painful to use. I find this happens often with interpreted languages, probably because adding a `print` statement and rerunning your program is just so easy that no one ever bothers to write a decent debugger for the language. Compiled languages, on the other hand, usually have debuggers because the time needed to add a `print` statement and rebuild a large program is longer than it takes to fire up the debugger. An example of this problem is present in my scripting language of choice, Python. I love writing in Python, but I do not love the Python debugger. It has improved over the past few years, likely because bigger and bigger systems are being built in Python, so having a debugger makes finding the bugs easier. As debuggers go, however, the ones for Python are nothing compared with those available for compiled languages.

The second instance where `print` beats a debugger is one that perhaps most readers of this column have not had to experience: bringing up a new piece of hardware. In the not-too-distant past it was uncommon for anyone except a device-driver writer to worry about bringing up new hardware. With more people using open source operating systems, however, it has become more common to have to do some level of work with new hardware. I recently experienced this when I bought a new laptop. Of all the things that did not work when I installed my operating system of choice, it happened to be the built-in keyboard that did not work with the operating system's keyboard driver. It turned out I could plug in a USB keyboard and boot with the internal keyboard disabled, but that was not quite how I envisioned using my new, light, slick, laptop—with a USB keyboard attached.

I normally don't work on keyboard drivers, but I know the people who did, and I know there is nothing more frustrating than having a whiny user send you an email message saying, "The keyboard doesn't work." The driver itself was not long, and I knew about where the hang would happen in the code, so I just backtracked from where I thought the hang point was and used an Emacs macro I had written for just such an occasion, as shown in Figure 1.

Attaching the code shown in Figure 1 to a key sequence, I could insert a `print` statement anywhere in my code, and when it was reached, it would print out the function, filename, and line that had been reached. Using this primitive method, I was able to track down what was causing the system to hang and thus could avoid it, as well as send a much more detailed bug report to the driver maintainer. Certainly more could be done with this macro; Figure 2 shows an example that builds on the previous code to enclose the `print` statement in a debug block that can be turned on and off from the makefile or command line.

Yes, there are times when you need or want `printf`, or `print` statements, but I still say that those times are, hopefully, few and far between.

KV

Figure 1. Emacs macro example.

```
(defun dbgprintf ()
 "Insert a debug printf for kernel debugging."
 (interactive "*")
 (indent-for-tab-command)
 (insert "printf(\"reached function %s file %s line %d\\n\",\n")
 (indent-for-tab-command)
 (insert "__func__, __FILE__, __LINE__);\n"))
```

Figure 2. Emacs macro example with `print` statement enclosed in a debug block.

```
(defun debug-block ()
 "Insert a debug printf inside a C ifdef debug block."
 (interactive "*")
 (insert "#if defined(DEBUG)\n")
 (indent-for-tab-command)
 (dbgprintf)
 (insert "#endif /* DEBUG */\n")
 (indent-for-tab-command)
)
```

Related articles on queue.acm.org

A Conversation with Bruce Lindsay
http://queue.acm.org/detail.cfm?id=1036486

Photoshop Scalability: Keeping It Simple
Clem Cole, Russell Williams
http://queue.acm.org/detail.cfm?id=1858330

A Call to Arms
Jim Gray, Mark Compton
http://queue.acm.org/detail.cfm?id=1059805

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and a member of the *ACM Queue* editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.

DOI:10.1145/2001269.2001281

Carsten Schürmann

Inside Risks

Modernizing the Danish Democratic Process

Examining the socio-technological issues involved in Denmark's decision to pursue the legalization of electronic elections.

On March 3, 2009 the German Supreme Court decided that the use of electronic voting machines in parliamentary elections is unconstitutional as long as it is not possible for citizens to exercise their right to inspect and verify the essential steps of the election. The verdict did not rule voting machines unconstitutional, but the particular election law that legitimized their use during the 2005 election. Electronic elections, like any traditional election, must be under public control. And as this has not been achieved yet, the Supreme Court decision effectively outlawed the use of e-voting machines for German parliamentary elections, at least for now.

Copenhagen election workers empty a ballot box containing votes cast during the June 7, 2009 election for the European Parliament.

PHOTOGRAPHS BY JØRGEN WEINREICH MALTESEN

European E-Voting Initiatives

One might think that events as such would have slowed down the efforts of other European nations to push e-voting technology into polling stations or even homes. This, however, is not the case. Many European countries are newly invigorated, running experiments with voter registration, vote casting, and vote tallying. Switzerland, a direct democracy, legalized Internet elections in 2009. Norway used a newly developed online voting system for its parliamentary elections on September 12, 2011. In 2005, Estonians were the first permitted to vote from their homes, using their national ID cards and off-the-shelf smartcard readers connected to their computers to authenticate themselves.

But why would governments advocate this kind of technology, risking decades of democratic achievements, in what seemingly contradicts common sense? There is more to this discussion than meets the eye. Governments and administrations currently revisit former decisions on how to implement the voting process and view them in the new light of information technology. Modern mobile devices such as smartphones, for example, interact and tinker with the very assumptions that secret and free elections are built upon. Off-the-shelf scanning technology can be used to identify individual sheets of paper simply by the composition of their fibers. It is also easy to take and transmit a photo or even a live video of the vote-casting process. European nations are pushing forward with the adoption of electronic and even Internet voting architectures, because their governments feel the risks of staying with the status quo outweigh the risks associated with modernizing the democratic process. I believe these European initiatives are healthy, necessary, and natural. The evolution of the democratic process must not come to

The evolution of the democratic process must not come to a standstill just because serious challenges lie ahead.

a standstill just because serious challenges lie ahead.

Denmark

There are only very few countries in Europe that have resisted (so far) the urge to jump on the e-voting bandwagon—among those is Denmark, a small country of slightly more than 5.5 million people. Denmark has a long history of democracy in Europe and is top-ranked according to the information society index (ISI)—an index often quoted for comparing countries according to their ability to access and absorb information and information technology. E-voting is not banned by law, no trials have been conducted on the national level to date, citizens generally respect and trust their government and politicians, and there is an educated electorate with a pervasive desire for fairness, openness, and equality.

Earlier this year, the Danish Board of Technology, an advisory committee to the government, released a report recommending Denmark to take initial exploratory steps toward research in and experimentation with e-voting technology, with the goal to improve the implementation of constitutional law. One aspect of the recommendations includes requiring a secret and free vote for all—not just those who are able to see, read, write, or visit the polling station on Election Day. The report even went so far as to propose an experimental deployment of mobile networked voting vans, visiting the elderly and those with disabilities on Election Day. There is little to disagree with: information technology can make elections more inclusive, more lawful, and also more convenient for those who run them.

Denmark has a long tradition of public control in nationwide elections. On Election Day, volunteers assume the role of election observers who oversee the entire voting process from early morning to late at night. They meet, look over each others' shoulders, tally and recount every vote as required by law, and by doing so, create much of the overall trust in the Danish voting process. Unfortunately, the numbers of volunteer election observers are dwindling, which is problematic as currently the demand exceeds supply. Denmark's municipalities do not expect this trend to change soon, but hope instead for information technology to complement the smaller corps of election observers.

Therefore, Denmark will sooner or later follow its European partners and jump on the bandwagon—at least for casting votes. Since 1984, the final result of an election—the number of seats awarded to each party in Parliament—is calculated by a computer program that runs on a Unix workstation located at the Ministry of the Interior and Health. The results computed that way are legally binding. For casting the vote, on the other hand, Danish law needs to be changed, because in its current form, it is prohibitively restrictive. All details are regulated, including how to design a ballot form and how to distinguish valid from invalid votes.

Denmark has evolved its voting laws over many decades. It became a constitutional democracy in 1849, ballots were made secret in 1901, women obtained the right to vote in 1915, ballots were allowed to be cast by letter since 1920, Danes living abroad obtained the right to vote in 1970, and in 1978 the legal voting age was set to 18. In 2009, the voting law was changed to require that any visually impaired voter must be accompanied by an election observer when casting a vote. The law will soon evolve to accommodate information technology, and this will happen in the spirit of the Danish tradition of participatory design. Decision makers, administrators, and (computer, social, and political) scientists, will work together in the best interest of democracy.

In this, I believe, Denmark stands apart from many other countries that are currently introducing new voting

laws, new voting culture, and new voting technologies without listening to the voices of scientists and other specialists. Decisions are all too often made by politicians, administrators, and industry alone, without properly attempting to understand the nature of the technological challenges, their vulnerabilities, and their effects on the trust of the voters and society as a whole. Scientists carry a large socio-technological responsibility and must be heard. Given the opportunity, they will act on behalf of the public, play the role of the independent auditor, keep an eye on the innovation and improvements of the democratic process, and increase public trust.

Defining a Metric for Success

Denmark's decision to pursue the legalization of electronic elections will to a large part depend on the implementation of the recommendations outlined in the report of the Danish Board of Technology and the success of the suggested trials. A few key groups will keep a close eye on these trials: The Ministry of the Interior and Health, which is responsible for their constitutionality; municipalities, who are responsible for a lawful, smooth, and efficient implementation; suppliers, who are responsible for the quality of the voting solution; and scientists who are responsible for ensuring the deployed technology serves the best interest of the public. Therefore, decision makers will be confronted with conflicting opinions about whether or not a trial was successful. It is thus prudent that all groups get together ahead of time and define a coherent metric for measuring success.

A modernized voting system does not need to be perfect, but it should implement a process that is at least as trustworthy as the one we know today.

A Danish child submits her parent's paper ballot for the European Parliament election.

It might seem like an obvious task to tackle, but it is not at all clear how to define this metric. Abstract concepts, such as trust, belief, and perception are difficult to model logically or mathematically; however, they need to be brought together with the formal aspects of hardware, software, and engineering in a meaningful way. Elections are cyber-social systems.

The usual indicators (such as voter turnout or opinion polls) are inadequate to measure success. Voter turnout, which is consistently above 85% in Denmark, is too infrequent a measure to be useful. Opinion polls that may be conducted more frequently are usually too volatile to be indicative. In February 2011, a poll conducted by the Danish newspaper *Børsen* showed that 63% (sample size 1,053) of Danish citizens of all ages would happily vote electronically even if it meant they must authenticate using a personal digital signature.

Instead, the metric must mirror the scientific evaluation of technical and social observations collected during the trial. It must measure the functional correctness of the voting infrastructure: how well the final result matches the voters' intent, how well privacy and secrecy are secured, and to what extend Danish voting culture is preserved. This is where the real challenge lies. Even for simple election schemes, such as winner-take-all, technological solutions consist of many complex and communicating components that interface in various ways with election officials and voters. Scientists need to convince themselves that the system will always perform according to specification, even under the most obscure and unlikely circumstances including software bugs, malicious hacker attacks, or power outages. Furthermore, they must be sure the collective trust of the population is not negatively affected and that there are clear and accessible mechanisms to exercise public control. Electronic elections in controlled environments have thus a much better chance for success than elections in uncontrolled environments, such as Internet-based (remote) elections, for which there are still more open problems than solved ones. A modernized voting system does not need to be perfect, but it should implement a process that is at least as trustworthy as the one we know today.

Finally, the metric must reflect the operational aspects of carrying out the electronic election. Election observers follow new protocols, initialize new machines and read out final results, handle unfamiliar physical evidence, and respond to unknown and unforeseen problems, ranging from hardware and software failures to denial-of-service attacks. For the trials, election observers must be adequately prepared, and their reactions and experiences must be carefully documented and evaluated.

Conclusion

With the right metric in place, Denmark will be in an excellent position to begin a rigorous scientific analysis of various voting schemes, technologies, and platforms. Politicians, administrators, and even suppliers have already signaled their willingness to cooperate and have promised scientists access to all parts of the election. If we do things right, Denmark will not repeat past mistakes of other nations, but its solution may serve as a guide for how to define future democratic processes. C

Carsten Schürmann (carsten@demtech.dk) is an associate professor at the IT University of Copenhagen, where he is leading the DemTech research project that aims to modernize the Danish democratic process.

DOI:10.1145/2001269.2001280

Phillip G. Armour

The Business of Software
Testing: Failing to Succeed

Optimizing what we learn from testing.

There are two situations in software testing that scare testers: when they see "too many" defects and when they do not see "enough." Too many defects may indicate the system contains a lot of problems even though it has moved into the testing phase. Too few discovered defects might imply the product is of high quality but it usually means the testing process itself is not working.

This is symptomatic of the paradox of testing: We want to find defects in the system under test, but we do not *really* want to find them. While the business of software has evolved from actually punishing people for finding defects, we can still be pretty ambivalent about them. Experienced testers have a good feel for the balance between finding too many defects and not finding enough[2] but most organizations and most customers would still prefer to hear "we haven't found many defects" than "we've found huge numbers of defects."

Testing as Knowledge Acquisition

If we view testing as a knowledge acquisition activity rather than simply a post hoc quality assurance process we get a different view. We also get a different view when we line testing up against the Five Orders of Ignorance.[1] Zero Order Ignorance (0OI) is lack of ignorance; it is proven knowledge, knowledge that we have acquired and against which some trial has made that certifies it as "correct." Second Order Ignorance (2OI) is lack of awareness of ignorance; it occurs when we don't know something and we are unaware that we don't know it. That is, we have ignorance about our lack of knowledge. We test systems primarily for these two of the Five Orders of Ignorance and their focus is quite different.

Testing for 0OI

We test to ensure the system performs as it was specified and as it was designed to perform. This is known as clean testing and it is ensuring the "proven" part of the definition of 0OI—we are showing that what we think we know is, in fact, correct. This kind of testing is relatively straightforward to set up, run, and prove. While the 0OI test set might be large, it is bounded—there are a finite number of things we want any system to do.

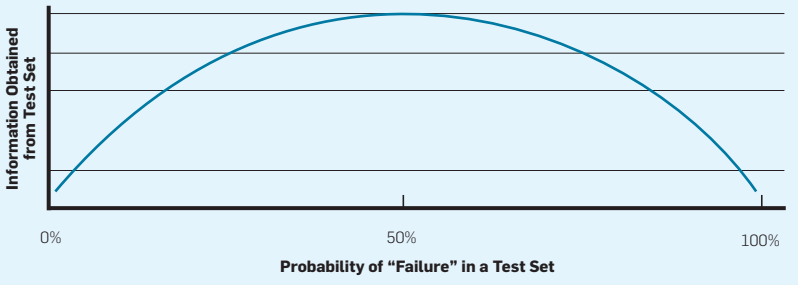
Testing for 2OI

The other kind of testing we do is to try and flush out any 2OI—we run dynamic tests on a system to see if there is anything we don't know we don't know about how the system actually runs. We can't be specific about the problems we are trying to detect (in advance of actually running the test) or presumably we would have already fixed them. However, it is true that designing a good test may structure our thinking so we see the problem before we actually run the test. In this case the test we do run will be focused on 0OI, proving that our preemptive knowledge insertion really works. Testing for 2OI is an unbounded test set—the number of things a system *might* do but should not is essentially infinite. This is also a difficult test set to create since we have to devise tests for something we are not specifically looking for. The best we can do is to apply testing heuristics: test to the boundaries of ranges, test all classes on inputs and outputs, test all combinations of multiple logic conditions, and so forth. To expose our ignorance about system usability, we might put the system in front of a naïve user and see what hap-

Figure 1. Information content equation.

$$\textit{Information Content of a Test} = Pf * \log\left(\frac{1}{Pf}\right) + Ps * \log\left(\frac{1}{Pf}\right)$$

Figure 2. Binary view of information content of a test.



pens. We might simply "monkey test" by firing all manner of random data at the system. In all these cases we do not know a priori what will happen. We are looking for something, but we do not quite know what it is.

Not Testing for 1OI

First Order Ignorance (1OI) occurs when we do not know something but we are fully aware of what we do not know. We should never test for 1OI. If we truly knew in advance what we did not know and where the limitations of our system's knowledge lie, we would resolve our ignorance first, incorporate what we have learned into the system, and then conduct a 0OI clean test to prove it.

A Successful Test

Here we see the dichotomy of testing: for 0OI a "successful test" does not expose any new knowledge, it simply proves our existing knowledge is correct. On the other hand, a 2OI test does expose new knowledge, but usually only if it "fails." The two yardsticks for success in testing: passing and failing tests are focused on these two different targets. This is partly where the tension between exposing and not exposing defects comes from. While having defects in our system is clearly a bad thing, finding them in testing (versus not finding them) is equally clearly a good thing. As long as there aren't too many.

How Much of a Good Thing?

For our 0OI testing 100% passing is the goal. Any test that "fails" indicates the presence of 2OI in the system (or the test setup or possibly sloppiness in testing, which is a different, Third Order Ignorance, process kind of failure). For 0OI testing, the ideal situation is that every bit of knowledge we baked into our system is correct and the successful tests simply prove it.

But what about the 2OI tests? Logic would suggest that a set of 2OI tests that exposed no defects at all would not be a good test run since nothing new would be learned.[a] It is possible that a test run that exposed no defects at all shows the system is very, very good and there are no lapses in the knowledge that we built into the system. But this is unusual and most testers would be very suspicious if they saw no defects at all, especially early in a testing cycle. Logic aside, emotion would suggest that a set of 2OI tests that exposed 100% errors would also not be a "good" test run. While finding such a huge number of errors might be better than not finding them, it indicates either a criminally poor system or a criminally poor test setup. In the second case the knowledge we acquire by testing relates to how we conduct tests that might be easily learned and fixed. In the poor system case our ignorance is in the system being tested and it may indicate an awful lot of rework in the requirements, design, and coding of the system. This is not an effective use of testing. Indeed, it might be that we are actually using testing as a (very inefficient) way to gather requirements, design, and code the system since the original processes clearly did not work.

So if 0% defect detection is too low, and 100% defect detection is too high, what is the right number? Well, it would be somewhere between 0% and 100%, right? To find where this sweet spot of defect detection is, we need to look back to 1928.

Transmission of Information

In 1928, Ralph Hartley, a telephone engineer at Bell Labs, identified a mechanism to calculate the theoretical information content of a signal.[3] If we think of the results of a test run as signals containing information about the system that are transmitted from the system under test to the tester, at what point is this information maximized? Hartley showed the information content of a signal is proportional to the logarithm of the probability that the event occurs. Viewing a test result as a simple binary—a test throws a defect ("failure") or a test does not throw a defect ("success")—the information content returned is given by the equation shown in Figure 1, where *Pf = probability of failure (error is detected); Ps = probability of success (error is not detected).*[4] The graph of this function is shown in Figure 2. In the simple binary view the maximum amount of information is returned when tests have a 50% probability of success (no error thrown). At that point, of course, they also have a 50% probability of failure.

Managing Test Complexity

This gives testers a metric by which to design test suites. If a set of tests does not return enough defects we should increase the test complexity until we see about half the tests throw errors. We would generally do this by increasing the number of variables tested between tests and across the test set. Contrariwise, if we see that more than 50% of the test cases expose defects, we should back off the complexity until the failure rate drops to the optimal level.

This optimization is ideal for *knowledge acquiring* (2OI) tests. For *knowledge proving* (0OI) tests, the ideal is 100% pass rate. The problem is, we do not know in advance that (what we think is) a 0OI test won't expose something we were not expecting. And sometimes what is exposed in a 0OI test is really important, especially since we weren't expecting it. Still, as we migrate testing from discovery to proof we should expect that the failure rate will switch from 50% to 100%. How this should happen is a story for another day.

I Knew That

I showed this concept to a tester friend of mine who has spent decades testing systems. His response: "I knew that." He said. "I mean, if *no errors* is bad and *all errors* is bad, of course a good answer is *some errors* in the middle and 50% is in the middle, right? I don't need an 80-year-old logarithmic formula derived from telegraphy information theory to tell me that."

Hmm, it seems the unconscious art of software testing is alive and well. C

References

1. Armour, P.G. *The Laws of Software Process.* Auerbach Publishers, Boca Raton, FL, 2004, 7–10.
2. Armour, P.G. The unconscious art of software testing. *Commun. ACM 48*, 1 (Jan. 2004).
3. Hartley, R.V.L. Transmission of information. *Bell Systems Technical Journal*, 1928.
4. Reinertsen, D.G. *The Principles of Product Development Flow.* Celeritas Publishing, Redondo Beach, CA, 2009, 93.

Phillip G. Armour (armour@corvusintl.com) is a senior consultant at Corvus International Inc., Deer Park, IL, and a consultant at QSM Inc., McLean, VA.

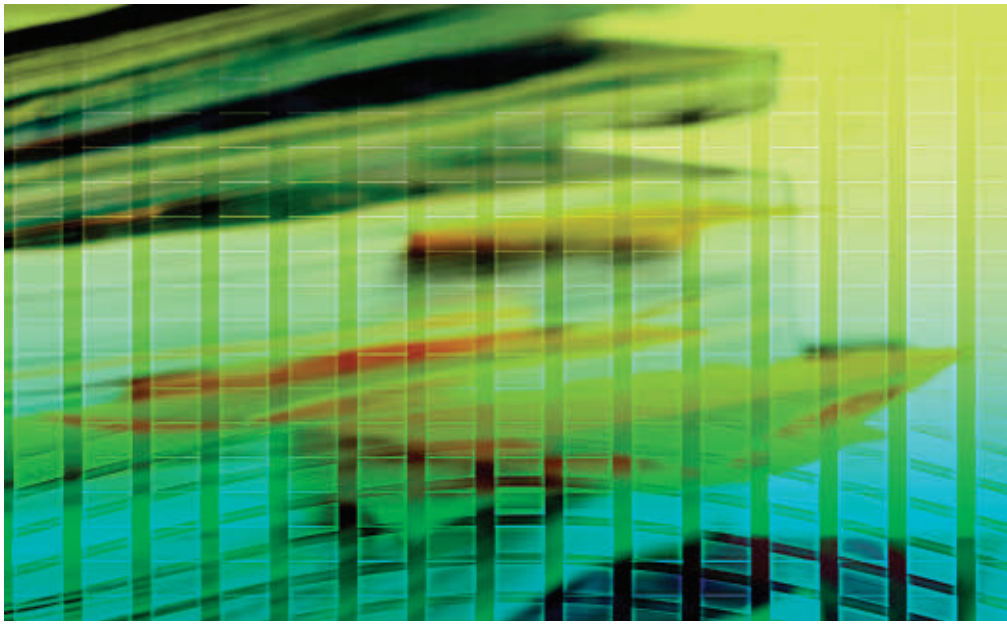
a Information theory does assert that the knowledge content of a system is increased by a 2OI test that "passes"—specifically it assures that the system will not throw an error under the specific conditions the test employed and provides some assurance for certain related tests. However, since the possible 2OI test set is functionally infinite, this assurance is not strong.

DOI:10.1145/2001269.2001283

Dan S. Wallach

Viewpoint
Rebooting the CS Publication Process

A proposal for a new cost-free open-access publication model for computer science papers.



Many computer science academics have written lately about problems with how our publication procedures have failed to scale as the field has grown (for example, see [2–4,8,9]). While others have focused on trying to shift CS from conferences back to journals, it is worthwhile to understand exactly which problems we are trying to solve.

Acceptance Rates. The top conferences, where publication can make or break a career, may publish 10% of the submitted papers.[a] Submission rates have grown in the past decade with acceptance rates either flat or dropping, despite an increasing absolute number of papers accepted. What happens to the rejects? Realistically, there are three categories. First, there are the "bubble" papers. If, for whatever reason, the conference were to double its acceptance rate, these would be published, but they were rejected either because they were seen as too narrow or uninteresting, or they were considered to have significant flaws. Next are "second tier" papers that could well be publishable at area-specific workshops or less competitive conferences. Also in the "second tier" category would be "least publishable unit" (LPU) papers, where an author advances their own work by the smallest possible amount and the program committee wants more. Finally, there are "noncompetitive" papers, where the paper would have no chance at publication in any respectable venue.

Overloaded Reviewers. As submission rates have gone up, program committees must decide between huge workloads per reviewer, or adding PC members to the point where most members are completely disconnected from most papers' discussions. This appears to increase the degree of randomness in whether a paper gets in or is rejected.

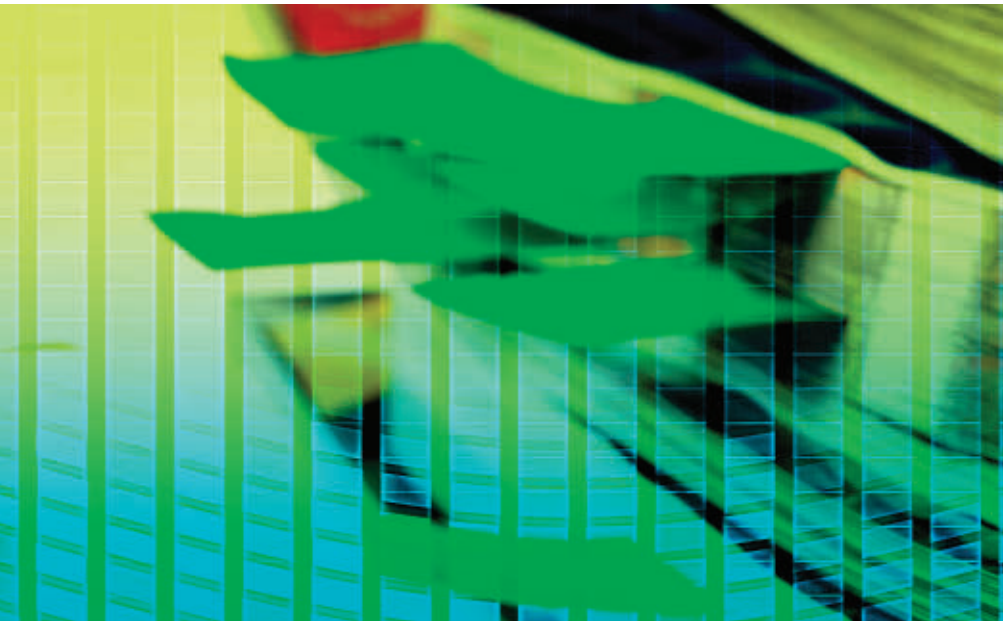
Resubmission. What happens with all these rejections? Many are inevitably resubmitted. Major conferences try to coordinate their accept/reject announcement dates with subsequent conferences' submission dates, but these can still be quite tight. (For example, there was only one week, in early 2010, between IEEE Security & Privacy's notification date and the USENIX Security Symposium's submission date.) Consequently, similar content is reviewed again and again.

Journal Latencies. In many other fields, conferences only take short papers and the "real" work is submitted to journals. Journals offer the benefit of having the same set of reviewers through each phase of a paper's life cycle. The reviewers can insist on improvements and can then agree that the

a Networking conference statistics are tracked by Kevin Almeroth (http://www.cs.ucsb.edu/_almeroth/conf/stats/), who links to statistics for other disciplines as well.

authors satisfied their requirements. In computer science, much work is never submitted to journals, and, at least in my experience, journals often receive a large volume of noncompetitive submissions, consuming reviewers' time. Furthermore, a latency of one year from submission to publication is entirely normal, and it can be far longer.

Short Incremental Work. Our current system of promotion and tenure strongly incentivizes authors to collect as many publications as possible, resulting in many different papers for any one given idea. Many current academics bemoan the good old days when you could have a group working on an involved, complex project, and have only one or a small number of groundbreaking papers. Given all these disparate concerns, can we evolve or redesign our way to a better structure for our academic processes?



A Clean-Slate Solution

Here, I describe the high-level design of a clean-slate solution, called CSPub (clean-slate publication, or perhaps, ambitiously, computer science publication). CSPub is, at its core, a mashup of conference submission and review management software with technical report archiving services like arXiv and with bibliographic management and tracking and search services like DBLP and Google Scholar.

Technical Reports on Steroids. Today, computer scientists have three different mechanisms for disseminating technical reports: their own personal or laboratory home pages on the Web, their departmental "official" technical report services, or centralized services like arXiv. Efficient publication and dissemination is the first and most fundamental service that we would have in CSPub. Ultimately, *every* paper published in our field can and should be available via this one mechanism, regardless of whether it is a "technical report," a "preprint," a "conference" paper, or a "journal" paper. Furthermore, when a paper is submitted to a conference or journal, the mechanism should be that the paper is submitted to CSPub, for all the world to see, and it should be flagged for the target conference or journal. CSPub could easily support a variety of submission mechanisms, including double-blind manuscripts for a conference linked to an optional public copy with the authors' full affiliation.

If all of academic computer science's scholarship were available in CSPub, a variety of new features would become feasible and relevant. First and most obviously, search engines could efficiently compute simple citation counts and *h*-indices as well as more sophisticated PageRank-like metrics. CSPub's ranking function(s) should be public and well documented, giving us a clear understanding of the incentives to adjust our publishing behavior. (The field of "bibliometrics" is dedicated to the design and evaluation of such systems, and those scholars would, no doubt, participate in the design of CSPub's metrics. *Nature* recently published a special issue on the topic.[1]) Alternately, CSPub could publish enough raw metadata that third-party services could apply their own metrics. *If we can agree on metrics that favor smaller numbers of better publications, CSPub can help incentivize the community to change its publishing behaviors.*

Given all these disparate concerns, can we evolve or redesign our way to a better structure for academic processes?

Publication status and awards (including "best paper" awards given by a conference or even "test of time" awards given in retrospect for papers that have had a significant impact) are metadata that could either be provided by the author or the conference steering committee. Such annotations help when somebody is searching CSPub for a paper to cite on a particular topic, and they may also contribute directly to how a paper is ranked. Professional curation would be necessary to prevent authors falsely giving themselves awards and to deal with related issues, including plagiarism, as they might arise. This could be aided by CSPub users "reporting" spam as well as automated anti-spam and anti-plagiarism systems.

With CSPub, other helpful features become easy. If Alice releases "version 2" of her paper, she could add bidirectional links, so "version 1" states that it has been superseded by "version 2", and "version 2" links to the earlier edition. This would allow ranking accumulated by preprints or tech reports to be applied to later conference and journal publications. Similarly, if Alice's paper has made a splash and she gets invited to give talks at a number of universities, those invited talks are, in

effect, additional metadata links that are endorsed by the institution that invited Alice to give a talk, and which will improve Alice's ranking. (In CSPub, a university's colloquium series could be represented much like a journal, linking to existing papers.)

Problem Solving. With CSPub, the top papers will still get in, as always. "Bubble" papers will, at the very least, get the proper priority date of their initial submission and will start getting citations. Conferences might introduce "accepted without presentation" distinctions, allowing more papers to be recognized and to avoid the need for subsequent resubmission. Today, authors of rejected papers must choose whether to edit and resubmit to a top conference, resubmit to a lower-ranked conference, or abandon a paper. With CSPub, these decisions can be delayed. If the paper turns out to be popular and starts gaining citations, then its authors will be motivated to update it and resubmit it. If the paper turns out to be poorly received, its authors can rationally abandon it and move on. *By reducing resubmission rates, conference program committees will have fewer papers to consider and can do a better job.*

When a paper is rejected, and the author receives feedback from the rejecting conference, that feedback would also be in CSPub, presumably (but not necessarily) private to the author. This creates an opportunity for the author to choose to give this feedback to a subsequent program committee, along with a statement about how the previous committees' comments were addressed. This moves the treatment of the manuscript closer to the consistent handling available through the journal process, yet with the speed of the conference process. An anonymous reviewer suggested that rejected papers might be indelibly tagged as such, in public, as a disincentive to authors submitting poor work to conferences. The idea of "negative" metadata, permanently associated with one's name and reputation, would be seen as offensive by many researchers. Certainly, CSPub could support such features if they were desired.

If all of academic computer science scholarship were available in CSPub, a variety of new features would become feasible.

CSPub can also enable new models for how conferences operate. "Unpublished" papers would be easy for program committees to discover on their own and "pull" into a conference. One-time workshops might be built purely around thematically linked, previously unpublished papers.

Journals. In CSPub, a journal is nothing more than an organization that adds metadata notations to papers in the system. As such, anybody can start their own journal for almost no cost. Some journals would have calls for papers, as conferences do, and authors would indicate a submission in their metadata when posting a manuscript. Other "journals" would be nothing more than collections of thematically related papers, perhaps put together by graduate students as part of their related work search. Of course, if a senior academic puts together a collection with a catchy title ("Alice's List of Seminal Papers in Blah-Blah Theory"), and Alice is a highly ranked professor, the collection would help increase the included manuscripts' rankings, both directly, due to Alice's strong personal ranking, and indirectly, by leading more academics to read and cite the papers on Alice's list.

CSPub offers a number of improvements to the journal latency problem. It allows early drafts to be seen and cited, while simultaneously being under review, and it completely eliminates printing latencies. Accepted papers "appear" immediately. CSPub also trivially supports new models, such as the hybrid journal/conference approach being taken by VLDB.[7]

Rollout

Without a doubt, the biggest challenge of CSPub is getting the ball rolling. Computer science scholarship is pub-

lished under a variety of professional organizations including the ACM, IEEE, AAAI, USENIX, ISOC, IACR, and many more. It is enough of a challenge to imagine any one of these organizations moving to the wholesale adoption of a new publication model, much less all of them at once.

The only feasible path is for one organization to develop CSPub for itself and start using it one conference at a time. Initially, anybody could submit a paper, as in arXiv or the Crypto ePrint server, and for the pioneering conferences, this would be the exclusive mechanism for submitting a paper to be considered for inclusion. By making this mandatory, at least for the authors at the pioneer conferences, the system will be populated by those papers and will have its initial users. CSPub's initial implementation could certainly build on the existing arXiv service, which already hosts many CS papers in its Computing Research Repository (CoRR).[6] According to Joseph Halpern, who runs CoRR, CoRR grew 35%–40% per year for several years running; he anticipates 10% growth (over 7,000 new papers) for 2011. CoRR is increasingly hosting archival papers from major journals and conferences. Elsevier also now allows authors to post accepted journal paper "pre-prints" on the Web and on CoRR. However, Elsevier does not allow authors to redistribute the final, camera-ready version of their work.

Already, many conferences and journals provide free, open access to their publications. All USENIX conference publications are available freely on USENIX's Web site. Similarly, *Logical Methods in Computer Science* (see http://www.lmcs-online.org) is a paperless journal published under the auspices of the International Federation of Computational Logic, with no cost to publishers or readers. Authors retain their copyright while agreeing to have their work distributed by the journal under a Creative Commons license. LMCS also distributes its publications through CoRR.

Authors who presently serve PDF files of their papers from personal or lab Web pages could incrementally migrate to using CSPub instead. CSPub could easily generate dynamic HTML that can be included in personal Web pages, research group pages, and so forth. By providing such convenient services, academic authors may well upload all of their papers to take advantage of CSPub's features and increase their work's visibility. Inevitably, the switch will occur one research area at a time as CSPub matures and individual communities adopt it.

By providing such convenient services, academic authors may well upload all of their papers to take advantage of CSPub's features and increase their work's visibility.

The largest concern with CSPub would be the loss of revenue from journal subscriptions and digital libraries hosted by our existing professional societies. Regardless, virtually any current manuscript can be found on one of its co-authors' home pages, free of charge for the reader. "Paywalls" between our papers and their readers will inevitably go away. CSPub, by virtue of institutionalizing this practice, would require the ACM, IEEE, and so forth to forgo this income as their authors adopt it. Consequently, conference registration fees will inevitably go up.[b] However, if we save our institutional libraries from the costs of journal subscriptions, that money could be redirected in many ways including scanning and entering old work into CSPub. Given that the bulk of U.S. computer science research is supported by the National Science Foundation, it is not unreasonable that the NSF could underwrite CSPub.

A related issue is *ownership*. New manuscripts can adopt a Creative Commons-style license where authors grant CSPub nonexclusive rights to redistribute their work. Older manuscripts often have their copyrights assigned to legacy publishers, who will certainly be reluctant to give up their lucrative franchises. We control our professional societies; we can vote for new policies. Other publishers may well put up a fight or go out of business as their authors abandon them. Ultimately, academic authors are incentivized to have their papers widely read and cited. Cost-free open-access to our manuscripts, whether through CSPub or any other mechanism, is the obvious way to accomplish this.

b Some associations, like USENIX, already provide archival papers freely online; adopting CSPub for them would not be financially disruptive.

Acknowledgments

This Viewpoint began through informal discussions with many of my peers. I would like to thank Drew Dean, Joseph Halpern, Mike Herf, Peter Honeyman, Carol Hutchins, Chris Jermaine, Dave Johnson, Chris Kelty, Eric Rescorla, Moshe Vardi, Suresh Venkatasubramanian, Ellie Young, and the anonymous *Communications* reviewers for their feedback and commentary. A longer version of this Viewpoint, with detailed citations, appears at http://www.cs.rice.edu/_dwallach/pub/reboot-2010-06-14.pdf.

References

1. Abbott, A. et al. Metrics: Do metrics matter? *Nature 465* (June 2010), 860–862; http://www.nature.com/news/2010/100616/full/465860a.html.
2. Birman, K. and Schneider, F.B. Program committee overload in systems. *Commun. ACM 52*, 5 (May 2009), 34–37; http://cacm.acm.org/magazines/2009/5/24644-program-committee-overload-in-systems/fulltext.
3. Crowcroft, J., Keshav, S., and McKeown, N. Scaling the academic publication process to Internet scale. In Workshop on Organizing Workshops, Conferences, and Symposia for Computer Science (WOWCS '08), (San Francisco, CA, Apr. 2008); http://www.usenix.org/events/wowcs08/tech/full_papers/crowcroft/crowcrofthtml/, also reprinted in *Commun. ACM 52*, 1 (Jan. 2009).
4. Fortnow, L. Time for computer science to grow up. *Commun. ACM 52*, 8 (Aug. 2009); http://cacm.acm.org/magazines/2009/8/34492-viewpoint-time-for-computer-science-to-grow-up/fulltext.
5. Halpern, J. Private communication, August 2011.
6. Halpern, J.Y. and Lagoze, C. The computing research repository: Promoting the rapid dissemination and archiving of computer science research. In *Proceedings of the Fourth ACM International Conference on Digital Libraries* (Berkeley, CA, Aug. 1999); http://arxiv.org/ftp/cs/papers/9812/9812020.pdf.
7. Jagadish, H.V. The conference reviewing crisis and a proposed solution. *SIGMOD Record 37*, 3 (2008), 40–45; http://portal.acm.org/citation.cfm?id=1462582.
8. Korth, H.F. et al. Paper and proposal reviews: Is the process flawed? *ACM SIGMOD Record 37*, 3 (2008):36–39, 2008; http://doi.acm.org/10.1145/1462571.1462581.
9. Vardi, M.Y. Revisiting the publication culture in computing research. *Commun. ACM 53*, 5 (Mar. 2010), http://cacm.acm.org/magazines/2010/3/76297-revisiting-the-publication-culture-in-computing-research/fulltext.

Dan S. Wallach (dwallach@cs.rice.edu) is an associate professor with the systems group in the department of computer science at Rice University in Houston, TX.

DOI:10.1145/2001269.2001284

Article development led by acmqueue
queue.acm.org

Applying lessons from software languages to hardware languages using Bluespec SystemVerilog.

BY RISHIYUR S. NIKHIL

Abstraction in Hardware System Design

The strain is beginning to show as hardware chip capacity has grown exponentially according to Moore's Law and we are called upon to design entire systems-on-a-chip (SoCs) of astonishing diversity and complexity.

Another important issue is that verification (testing) has so far been done using simulation, but this is decreasingly practical.[3] In modern SoCs, the hardware is large and complex, and it runs heavy software loads such as full-featured operating systems and applications. Verification involves simulating all of these together, and it is not unusual for software simulations to run for days or weeks. Simulation speeds are typically cited in the 10s to 100s of KHz, whereas what are needed are MHz speeds.

ILLUSTRATION BY DAVE BOLLINGER

THE HISTORY OF software engineering is one of continuing development of abstraction mechanisms designed to tackle ever-increasing complexity. Hardware design, however, is not as current. For example, the two most commonly used hardware description languages (HDLs)—Verilog and VHDL[9,12]—date back to the 1980s. Updates to the standards lag behind modern programming languages in structural abstractions such as types, encapsulation, and parameterization. Their behavioral semantics lag even further. They are specified in terms of event-driven simulators running on uniprocessor von Neumann machines (and this is true even for their recent descendents, SystemVerilog and SystemC[10,11]).

These HDLs all have "synthesizable subsets" that constrain them in an effort to narrow this behavioral gap, but the mismatch is never completely eliminated.

The only feasible solution to this problem is what hardware people call *emulation*, which is simulation of the hardware design on field-programmable gate arrays (FPGAs). This FPGA-based emulation cannot be left to final drafts of a design, however; it must begin at the very start of the design process, from early models onward. Doing so places further requirements on the behavioral semantics of HDLs.

Simple emulation is not enough, however. Previously it might have been acceptable to use separate high-level languages for models and test benches that run only in software simulation, such as SystemC TLM (Transaction Level Modeling)[19] and SystemVerilog VMM (Verification Methodology Manual).[2] Today, however, what is needed is an HDL that can be used for all these purposes—from early models and test benches to detailed implementation—where these different components can be mixed freely and where *all* these components can be synthesized for FPGA simulation. Thus, the HDL needs to be universal, both in the sense that it is suitable for all kinds of digital designs (for example, not just signal processing) and in the sense that it should be fully synthesizable, whether used for models, test benches, or implementations.

These requirements mean a radical reconsideration of HDLs. This article describes Bluespec SystemVerilog (BSV),[18] the design of which was motivated by just such a reconsideration while reusing features from SystemVerilog wherever possible. BSV's structural extensions (involving more expressive types, overloading, encapsulation, and parameterization) are inspired by Haskell,[20] the functional programming language. Its behavioral semantics are inspired by Term Rewriting Systems[13] (or Guarded Atomic Actions), which are best suited for describing complex, concurrent hardware. This is more than a theoretical exercise—BSV and its tools have been used in industry for more than five years (your smartphone or tablet may well contain components designed with BSV).

Why Not Use Existing Software Language for Hardware Design?

Before looking at BSV, it is useful to

consider whether a new language is necessary at all. Although from a functional point of view "it's all just computation," hardware system design is characterized by features very different from software design.

Concurrency/parallelism. Hardware systems typically have parallelism that is massive, fine-grain, heterogeneous, and reactive. (Unlike some authors, I make no distinction between the terms *concurrency* and *parallelism*). This parallelism results in increased speed, which is often the main reason to implement something in hardware. *Massive* in this context means that the number of parallel activities may number in the thousands or even millions. *Fine-grain* means that parallel activities interact frequently (measured in time scales of clock cycles) over possibly very small shared resources (such as individual registers of a few bits). *Heterogeneous* means that parallel activities involve diverse tasks—in contrast, for example, to SIMD (single instruction, multiple data) or SPMD (single process, multiple data) computation. Finally, reactive means that parallel activities are often triggered by asynchronous events, such as unpredictable data arrival, interrupts, arbitration resolution, and I/O.

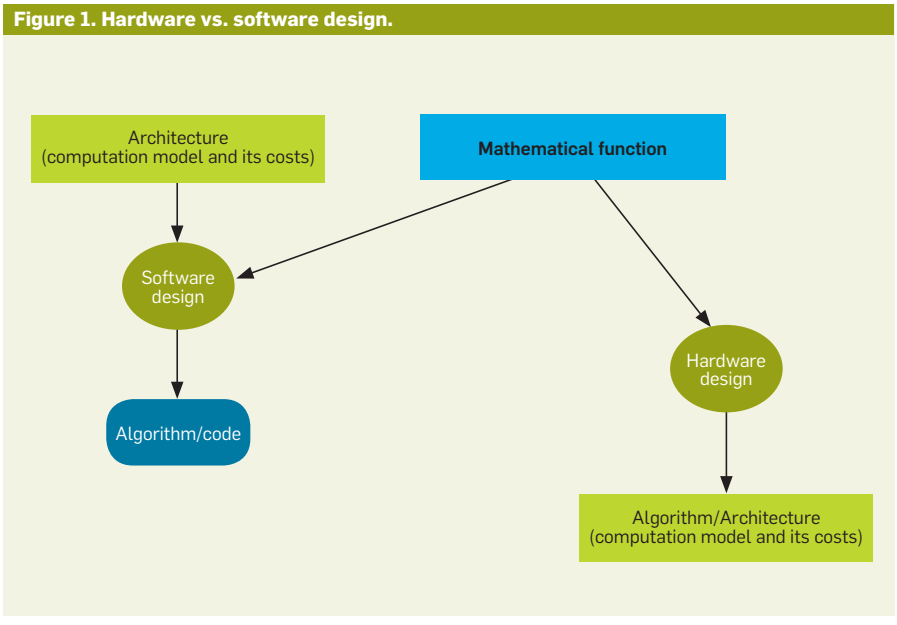
Unfortunately, most software languages are not parallel at all, but instead are entirely sequential. Some extensions for massive parallelism address only SIMD parallelism. Thread extensions can handle heterogeneity, but they are notoriously difficult for massive, fine-grain, or reactive parallelism.[1,15,22]

Architecture and algorithm. In hardware system design, good architecture (with its attendant cost model) is a central design goal and an outcome of design activity, whereas software is mostly designed for a fixed, given input architecture (typically von Neumann, perhaps with extensions such as SIMD), as illustrated in Figure 1. Thus, in hardware design, algorithm and architecture are inseparable, and it is meaningless to talk about "pure algorithm design" in the abstract, without some architectural model that gives it a concrete cost metric.

Since they are all Turing-complete, any architecture can be modeled or simulated with existing programming languages, but there are two fundamental problems with this. First, it takes extraordinary discipline (and therefore a lot of time and effort) to model a complex architecture accurately. Second, modeling an architecture often means losing orders of magnitude in execution speed, both because of the extra layer of interpretation and because the natural parallelism of the architecture being modeled is essentially completely discarded.[3] Domain-specific languages (DSLs) can address the first issue, but not the second. For most software programming languages, the "native" architectural model is the von Neumann model (one sequential process, with constant-time access to a large, flat memory), and it is only native von Neumann algorithms that execute at speed.

The term *architectural transparency* expresses the idea that the source program directly reflects the desired architecture. Abstraction mechanisms can hide detail but should not distort the resulting architecture. This property is essential for the programmer/designer, for whom abstraction is good, provided that it does not compromise predictability and control. This property is also essential for compilers (synthesis) to produce efficient implementations.

Figure 1. Hardware vs. software design.



The Inadequacy of Software Simulation

As mentioned, today's SoCs need FPGA-based emulation to achieve acceptable simulation speed, and this requires universal applicability of HDL (models, test benches, and implementations of the full gamut of SoC components) and universal synthesizability. Unfortunately, no software languages are suitable for this.

Many in the industry advocate using a combination of C++ and SystemC—the former to describe the algorithmic content of individual intellectual property (IP) blocks, and the latter to describe system-level communication, hierarchy, and integration of these IP blocks into an SoC. The C++ inside IP blocks can then be subjected to so-called high-level synthesis[5] (HLS), using tools that automatically generate parallel hardware from sequential C++, given declarative objectives such as latency, throughput, area, power, and target silicon technology. A detailed critique would require another article, but this section and Stephen A. Edwards' article on the topic[7] provide some background.

Bluespec SystemVerilog

Let's focus first on the computation model of BSV (that is, its behavioral semantics), because that is the greatest limitation of existing software languages for hardware design. Then we present an example to demonstrate the use of modern structural abstraction mechanisms.

Rules: The basic computation model. Verilog and VHDL, with their origins as simulation languages, are built on the uniprocessor von Neumann

model: sequential processes, stack-based procedure calls, traditional stack-based updatable variables, and cooperative multitasking. BSV, on the other hand, is born out of direct hardware description—state and parallel state transitions. The computer science literature has a computation model that is well suited for this, variously called Term Rewriting Systems,[13] Guarded Atomic Actions, or Rewrite Rules. It is used in many formal specification languages for complex concurrent systems, including Dijkstra's Guarded Commands,[6] Chandy and Misra's UNITY,[4] Lamport's TLA+,[14] Abrial's Event-B,[16] and many more. The following BSV excerpt illustrates a computation to sort four integers in ascending order.

```
Reg #(int) x1 <- mkRegU;
Reg #(int) x2 <- mkRegU;
Reg #(int) x3 <- mkRegU;
Reg #(int) x4 <- mkRegU;

rule swap12 (x1 > x2);
    x1 <= x2;
    x2 <= x1;
endrule

rule swap23 (x2 > x3);
    x2 <= x3;
    x3 <= x2;
endrule

rule swap34 (x3 > x4);
    x3 <= x4;
    x4 <= x3;
endrule
```

The first few lines instantiate four registers (storage elements) named `x1 ... x4`, containing values of type `int`. The `mkRegU` right-hand sides are the constructors, whose details are not important here.

The rule (or Guarded Atomic Action) `swap12` is like a reactive process (that is, it can "fire" whenever its Boolean condition `x1>x2` is true). The body of a rule is an *Action* and is semantically an *instantaneous* event. Here it is composed of two smaller Actions: one that assigns the value in `x1` into `x2`, and vice versa. The net effect is to swap the values in the two registers. Rules `swap23` and `swap34` are similar.

There is no inherent sequencing between rules—a rule can fire whenever its condition is true. Thus, in the example, it may happen that rules `swap12` and `swap34` perform their swaps in parallel. But what about rules that act on shared state, such as `swap12` and `swap23`, both of which update `x2`? Rules have the semantics of *atomic transactions*, and thus, in this case, those two rules can fire only in some order (such as, sequentially). For this example, it does not matter which of the two possible orderings is chosen; the registers will eventually be sorted anyway.

The strain is beginning to show as hardware chip capacity has grown exponentially according to Moore's Law and we are called upon to design entire systems-on-a-chip (SoCs) of astonishing diversity and complexity.

The previous fragment seems repetitive—four similar registers and three similar rules—but we wrote it that way first to explain the semantics. It would more likely be written this way:

```
Vector#(4,Reg#(int))
   x <-replicateM (mkRegU);

for (int j=1; j<4; j=j + 1)
  rule swap (x[j-1]>x[j]);
    x[j-1] <= x[j];
    x[j]   <= x[j-1];
  endrule
```

This expands ("statically elaborates," in hardware language jargon) to what is essentially the same fragment as before. Of course, the `4` here could be abstracted into a parameter. The fragment represents the classic bubble-sort algorithm of introductory programming texts—with the same $O(N^2)$ worst-case complexity—except that here the "bubbling" can happen in parallel, modulo atomicity (that is, any pair of enabled rules that do not conflict on a shared register can fire in parallel).

When atomicity requires two enabled conflicting rules to be sequenced, the order chosen can be left unspecified. This is at first alarming to hardware designers, where nondeterminism is equated with unpredictable (bad) results. It is usually welcomed for formal specifications, however, where insisting on a schedule too early is regarded as an unnecessary over-specification. In BSV various techniques are available to nail down particular scheduling choices whenever necessary.

What rules mean for hardware, and why they matter. All hardware design involves specifying parallel activities

that update state. The classic style in Verilog (all of these remarks apply to VHDL as well) is state-centric. For each state element x:

```
always @ (posedge CLK)

  if (...cond1...)
      x <= ... value1 ...

  else if (...cond2...)
      x <= ... value2 ...
  else if ...
      ...
  else
      x <= ... valueN ...
```

More generally, multiple state elements may be updated in each conditional arm, the conditional could be written as a case statement. In synthesizable code, however, each state element may be updated in only one always block and may be updated at most once in each conditional arm.

This is how the problem of parallel updates to a shared resource is solved in (synthesizable) Verilog: it is updated in only one "process" (always block) and, for every clock, exactly *one* possible new value for x is specified. It is almost a direct, explicit specification of the hardware that is generated: the conditional represents a *multiplexer* on the input of the x register; the conditions specify how one of the multiplexer inputs is selected to be clocked into the register; and the conditions may also specify whether the register is updated at all. Collectively, all this logic is referred to as *control logic*.

This organization of behavior in terms of states is, unfortunately, orthogonal to how behavior is typically conceptualized: as a collection of steps, or *transactions*. Each step of behavior may involve updates (perhaps conditionally) to multiple state elements. This "transpose" from the transaction-centric dimension to the state-centric dimension is at the heart of the problem with Verilog. The state-centric view does not scale well as the number of parallel activities increases and becomes more complex in regard to how the activities compete for shared state. Consider the problem in Figure 2, which illustrates two registers x and y updated by three parallel processes under certain conditions. Assuming that process 0 has lower priority than process 1 for the update of x, and process 1 has lower priority than process 2 for the update of y, the Verilog solution may look like this:

```
always @ (posedge CLK) begin
   if (cond2)
       y <= y - 1;
   else if (cond1) begin
       y <= y + 1;
       x <= x - 1;
   end

   if (cond0 && !cond1)
       x <= x + 1;
end
```

This Verilog code could be written in many styles, but they are all state-centric. The "transpose" from problem statement into state-centric code has intertwined and obscured the original transactions. Also, the priorities are wired into the code structure—the priority of process 2 over process 1 is expressed in the sequentiality of if ... else. The priority of process 1 over process 0 is expressed in the !cond1 term. In fact, a clever designer may recognize that when cond2 is true, process 1 cannot update x, thus opening an opportunity for process 0 to do so. Therefore, the designer might "improve" the last two lines to:

```
if (cond0 && (!cond1 || cond2))
    x <= x + 1;
```

Note the transitivity of control: Process 0's update of x is affected by the condition of the nonadjacent Process 2. Third, some of the process conditions are duplicated in multiple clauses, with the usual risks of incorrect duplication.

Imagine a change to the spec (this happens all too frequently in the real world) requesting a different priority among the three processes. Or imagine adding another process, or possibly another shared state variable, or both, with new control conditions. The whole code would need to be revised; it's hard to make local incremental changes.

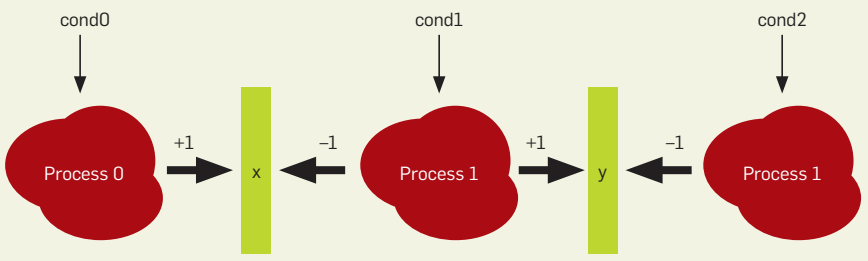
The fact that even such a small example carries such subtleties in control logic should sensitize the reader to the fact that reasoning about parallel updates in a state-centric manner on a per-clock basis can get very tricky while scaling to more parallel activities and shared state elements, and scaling the complexity of the update conditions.

The BSV code for this example looks like this:

```
rule proc0 (cond0);
   x <= x + 1;
endrule

rule  proc1 (cond1);
   y <= y + 1;
   x <= x - 1;
endrule

rule proc2 (cond2);
   y <= y - 1;
endrule

(*descending_urgency = "proc2,
proc1, proc0" *)
```

The rules are a direct expression of the process descriptions (i.e., they are transaction-centric, not state-centric), and the final line expresses the scheduling priority. Synthesis from this code produces essentially the same

Figure 2. Two registers updated by three parallel processes.



Update priority: Process 0 < Process 1 < Process 2

Verilog code shown earlier—namely, the complex control logic necessary to manage parallel update to shared state. It is easy to add more processes, or more shared state, or more complex update conditions.

Rules matter even more when scaling up to systems organized into modules; we will return to this discussion after describing modularity mechanisms.

Organizing rules into modules. In object-oriented programming, complex systems are organized into objects. The key behavioral construct is a *process* or *thread* (just one in a sequential language). A process is rooted in some module (perhaps "main"), but it may span object boundaries through method calls. Because methods may themselves call methods, a process may thus access an unbounded number of objects. Each method is semantically a fragment of a process.

Similarly, in BSV a syntactic `rule` in one module can invoke a method in another module. Methods can, in turn, invoke other methods. Each method is semantically a fragment of a Rule. Thus, a method is more than just a procedure—it is a guarded expression. A semantic Rule is composed of syntactic components: a `rule` construct and (transitively) all the methods it invokes; this semantic Rule is the unit of parallelism and atomicity.

A small module that implements a FIFO containing integers illustrates this. First its interface declaration:

```
interface FIFOint;
   method  Action   enq   (int  x);
   method  int      first ();
   method  Action   deq   ();
endinterface
```

Like a C++ virtual class, it merely describes the externally visible methods of a module. The `enq` method enqueues an item `x`. Its `Action` type indicates that it returns no value and just has an effect. The `first` method takes no arguments and returns the value of the element at the head of the queue. The `deq` method is a pure `Action`—it has no arguments or results; it just has the effect of discarding the first element in the queue.

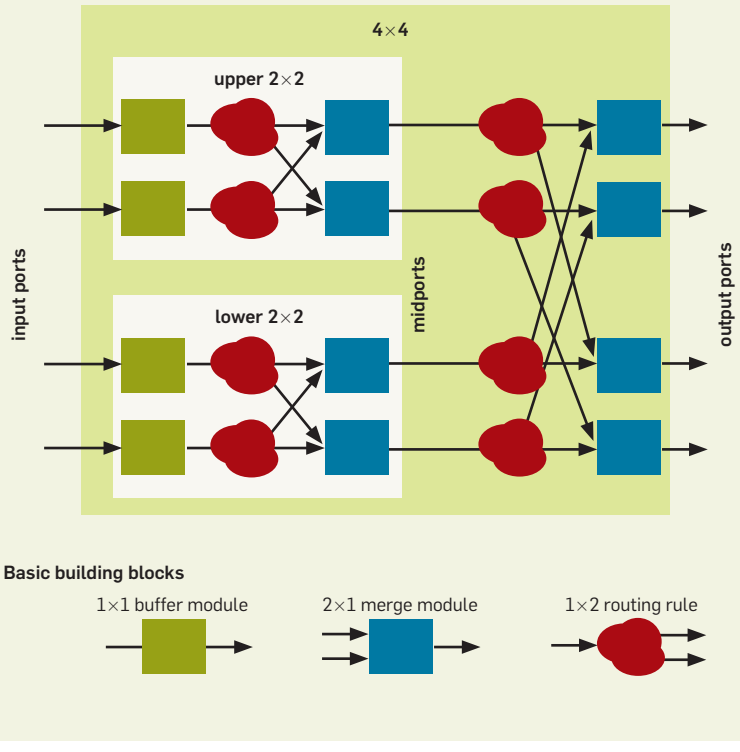
Figure 3 illustrates the code for one possible module that implements this interface. This is a module *constructor* that can be *instantiated* multiple times to create actual modules (hence the stylistic choice of `mk` in the module name `mkFIFOint`, suggesting the word *make*). `FIFOint` in the header specifies the interface. The next two lines instantiate two registers: `data`, containing an integer, initially unspecified; and `empty` containing a Boolean, initially `True`. The `enq` method is guarded by the condition `if (empty)`—any rule from which it is invoked will be enabled only if the guard is true. When invoked, it stores its argument `x` in the data register and sets `empty` to `False`. The `first` and `deq` methods are guarded by the condition `if (! empty)`—any rule from which they are invoked will be enabled only if this guard is true. The `first` method returns `data`'s value and does not change the FIFO. The `deq` method has the effect of setting `empty` to `True`.

This example is simple, but it could be generalized to an *N*-element FIFO by extending `data` to a vector of registers and maintaining the usual head

Figure 3. A FIFO module.

```
module mkFIFOint (FIFOint);
  Reg #(int)  data  <- mkRegU;
  Reg #(Bool) empty <- mkReg (True);

  method Action enq (int x) if (empty);
         data <= x; empty <= False;
  endmethod

  method int first () if (! empty);
         return data;
  endmethod

  method Action deq () if (! empty);
         empty <= True;
  endmethod
endmodule
```

Figure 4. Butterfly crossbar switch



and tail pointers. The FIFO could also be generic (polymorphic) in its content type, instead of just containing `int` values.

Many modules have FIFO-like interfaces—that is, flow-controlled ports that accept or yield values. We can define polymorphic interfaces for this:

```
interface FIFOin #(type t);
  method Action enq (t x);
endinterface

interface FIFOout #(type t);
  method t first ();
  method Action deq ();
endinterface
```

Other than in the syntactic details, these capabilities are similar to what is seen in object-oriented languages. The key difference is that while OO methods are fragments of sequential processes, BSV methods are fragments of rules and have guards.

A larger example. Figure 4 illustrates a Butterfly crossbar switch, which is a hardware device with N input ports and N output ports (here, N=4). Packets enter at input ports, and each is routed to an output port. Note that the switch has a recursive structure: the 4x4 switch shown contains, as components, two 2x2 switches, and each of those in turn contains two 1x1 switches (buffers). Two of these 4x4 switches, in turn, could be used to make an 8x8 switch, and so on. Also note that there are three basic components: FIFOs (1 in, 1 out), merge boxes (2 in, 1 out), and routing logic (1 in, 2 out). Figure 5 shows the interface type declaration.

It is generic, or polymorphic, in the type `packet_t` of packets flowing through the switch. It is nested, or hierarchical, in that it is defined in terms of two sub-interfaces: `input_ports` and `output_ports`. These, in turn, use standard `List` data structures to refer to collections of `FIFOin` and `FIFOout` interfaces. Figure 6 shows a module `mkXBar` implementing this interface.

P1-P3 are parameters: `n` requests an $n \times n$ switch; `destinationOf` is a function from packets (of type `t`) to destinations (of type `UInt #(32)`, unsigned 32b integers) that can be used to take routing decisions; and `mkMerge2×1` is a constructor for the 2-input, 1-output merge modules. In the semantics, a module constructor is just a function from parameters to modules. Further, it is a higher-order function—that is, its parameters may themselves be functions and module constructors. These features are standard in advanced programming languages such as Haskell[20] and Standard ML (SML)[17] but are very unusual in HDLs, particularly in *synthesizable* HDLs.

The bulk of the module is a recursive definition of the module. The `then` part of the big conditional is the base case (`n==1`). A 1x1 switch is implemented using `mkFIFO`. The next line uses functions `toFIFOin` and `toFIFOout` (easy; not shown) to separate the `FIFO` interface into `FIFOin` and `FIFOout` interfaces, and builds them into singleton lists `iports` and `oports`.

The `else` part is the recursive case (`n>1`). The code first recursively instantiates `mkXBar` twice, at size $n/2$, for the upper and lower 2×2 subswitches. It appends their `input_ports` and `output_ports` and holds them in the lists `iports` and `midports`, respectively. It then instantiates the vertical column of n 2×1 merge blocks at the right edge of the switch shown in Figure 4 using the library higher-order function `replicateM` on parameter `mkMerge2×1`. It uses the standard library list-processing higher-order function `map` with the `oport_of` function (not shown; simply returns the output port of a `mkMerge2×1` module) to create the `oports` list.

Figure 5. Interface type declaration.

```
interface XBar #(type packet_t);
     interface List #(FIFOin #(packet_t))    input_ports;
     interface List #(FIFOout #(packet_t))   output_ports;
endinterface
```

Figure 6. A crossbar switch module.

```
module mkXBar #(Integer n,                                    //(P1)
                Function UInt #(32) destinationOf (t x),      //(P2)
                Module #(Merge2x1 #(t)) mkMerge2x1)           //(P3)
               (XBar #(t) );
  List#(Put#(t)) iports; List#(Get#(t)) oports;

  if (n == 1) begin
      FIFO#(t) f <- mkFIFO;
      iports = cons(toFIFOin(f),Nil);
      oports = cons(toFIFOout(f),Nil);
  end
  else  begin
      XBar#(t) upper <- mkXBar(n/2, destinationOf, mkMerge2x1);
      XBar#(t) lower <- mkXBar(n/2, destinationOf, mkMerge2x1);
      iports = append (upper.input_ports, lower.input_ports);
      let  midports = append (upper.output_ports, lower.output_ports);

      List#(Merge2x1) merges <- replicateM (n, mkMerge2x1);
      oports = map (oport_of, merges);

      for (Integer j = 0; j < n; j = j + 1)
        rule route;
          let x = midports [j].first();
          let jDest = computeRoute (destinationOf (x), j, n);
          if (jDest == j) merges [j]    .iport0.enq (x);
          else            merges [jDest].iport1.enq (x);
          midports [j].deq();
        endrule

  end
  interface input_ports = iports;
  interface output_ports = oports;
endmodule: mkXBar
```

The `for` loop generates *n* instances of the rule. In any one instance, the name `x` refers to the packet at the head of the j^{th} port in `midports`. The function `computeRoute` (not shown; a simple numeric calculation) is applied to get the index of the merge module into which `x` should be forwarded, which is done in the small conditional. The rule also dequeues the packet.

Each rule expressing the switch's behavior hides much control detail:

- The rule is enabled only when a packet is available on the rule's upstream FIFO (because of guards on the `first` and `deq` methods).
- The rule is enabled only when a packet can be sent into its downstream merge block (because of the guard on `enq`). For example, it may be blocked as a result of flow control.
- This last control condition is further nuanced by the fact that the downstream merge block is dynamically selected, depending on the packet, and, therefore, only the enq guard of the selected block matters.

An even subtler control condition arises out of conflict between two rules. In the `for` loop, pairs of rules enqueue into the same merge blocks. This contention may require one of the competing rules to be stalled (that is, not enabled) to satisfy atomicity.

Why rules matter (continued). In Verilog, all of the above control considerations manifest themselves as explicit, visible control wires on the physical input and output ports of the merge modules, together with a protocol on how to use those wires. These are specified explicitly by the designer, and require effort on a per-module basis. They are ad hoc in that these control interfaces and protocols vary from module to module, from designer to designer, and even from day to day for the same designer. The protocol (semantics) is usually conveyed in accompanying text and timing diagrams and, as you might imagine, is often incomplete, ambiguous, and sometimes wrong. When working with rules, the compiler handles this control complexity naturally, automatically, and formally.

This automation is especially important in accommodating change (for fixing a bug, reacting to changing specs, and so on). The `mkMerge2×1` module is a formal parameter, a black box. One actual parameter may permit both `enq` ports to be activated simultaneously, whereas another may not, perhaps because one has separate internal buffering, while the other shares internal buffers. This affects whether competing rules in `mkXBar` can fire simultaneously or not (this is another example of the transitivity of control). With ad hoc control logic (as in register-transfer level, or RTL), it may not be able to accommodate such a change without redesigning `mkXBar`, whereas with rules, the compiler handles it.

Rules are part of a good DSL for hardware description because they are like state machine transitions, familiar and intuitive to hardware engineers. Rules, however, are more profoundly powerful because they are parallel and atomic.

Consider this analogy: in compiling software, imagine doing register allocation by hand, and designing and documenting an ad hoc calling convention for every function you write. It is rather difficult and painful doing it even once, but it's much worse if the source code changes because register allocation has a transitive effect across function boundaries, such as control logic across hardware module boundaries. Just adding an argument to a function may require redoing register allocation for both the callers and the callees, and this may have a ripple effect across multiple functions. Fortunately, compilers automate this, and adding an argument to a function is trivial for the programmer. Similarly, Rules simplify writing source code and modifying it; the compiler (synthesis) figures out the required changes in control logic.

Rules are part of a good DSL for hardware description because they are like state machine transitions, familiar and intuitive to hardware engineers. Rules, however, are more profoundly powerful because they are parallel and atomic.

Synthesis into Hardware

Historically, rules in programming and specification languages were concerned just with functionality and not performance; there is only an abstract notion of time in the sense that one rule may fire before another. In hardware design, performance is usually a central requirement, not merely an implementation detail; so the computation model must make this visible to the designer.

Space does not permit a full de-

scription here, but BSV indeed defines such a concrete mapping of rules into clocked hardware. Clocks and Resets are abstract data types, and rules and methods are mapped into clock and reset domains. Strong static type checking verifies the isolation of these domains. In summary, BSV has easy-to-use and semantically rigorous facilities for the robust creation of designs with multiple clock domains.

Another very important consideration today is power consumption. At a fine granularity, BSV clocks can be gated, with gating conditions integrated organically into rule conditions. At the coarser granularity of IP blocks, BSV has power-management facilities to switch off or scale power and clocks to entire modules or subsystems in a disciplined manner.

One intriguing thought is that BSV could also be compiled into asynchronous logic (which has many potential advantages relating to circuit timing and power consumption), but how to do this, and how to reason about system performance in this regime, are still open research questions.

Synthesis quality for ASICs. Technology to compile Rules into efficient hardware was first developed more than a decade ago.[8] It has been improved continuously since then and has been used on hundreds of designs. In some there was an existing RTL design for apples-to-apples comparisons, and in general, quality has been comparable (as measured in silicon area and performance), typically within 5%.

In a few cases, BSV-generated RTL has significantly better quality (15%–25%) than hand-coded RTL. This is surprising at first because in software you typically pay a performance penalty for higher abstraction. As illustrated in Figure 1, however, higher abstraction in hardware design can yield significant *algorithmic* advantages (that is, more efficient architectures).

Synthesis for FPGA-based simulation. The universal applicability and synthesizability of BSV has opened the door for many to use FPGAs as their standard simulation engine, from the earliest stages of design, and with a "design-by-refinement" methodology.

Bluespec Inc., by "eating its own dog food," has used BSV to create highly parameterized communication, debugging, and IP libraries for FPGA boards. These form a kind of operating system for FPGAs, so that the user has high-level facilities for communicating with a host and debugging a model or design instead of the customary painful wrestling with the raw FPGA board hardware.

Where users previously might have written test generators or analyzers or models in C++, they can now write them in BSV without loss of abstraction but with the ability now also to synthesize them and run them on the FPGA adjacent to their designs.

As a result, given that boards with significant FPGA capacity can now be purchased for less than $2,000 (such as Xilinx ML605), with comfortable high-level abstractions for communications, debugging, and libraries in BSV, designers can now treat FPGAs as another computation weapon in their arsenal, just like GPGPUs (general-purpose computing on graphics processing units).[21]

Conclusion

Historically there has been an almost total separation between software and hardware designers, much of it attributable to the wide cultural (semantic) gap between the languages they use to design their respective systems. Modern systems demand a reduction or elimination of this specialization. All interesting hardware systems today must run sophisticated software, and many complex software computations must move to hardware to meet performance and power consumption goals.[21]

BSV is an attempt to address this problem. Rather than trying to force fit solutions for von Neumann machines (such as C++ and threads) into hardware description, BSV has taken excellent ideas from software that are naturals for hardware description. It describes behavior using Rules (from Term Rewriting Systems), which are excellent for massive, fine-grain, heterogeneous, reactive concurrency (hardware!). It describes structure and structural abstraction using types, overloading, higher-order functions, parameterization, and even monads from Haskell. These ideas have been tested in the field for well over five years and used in finished products, one of which might be in your pocket right now. C

Related articles on queue.acm.org

SoC: Software, Hardware, Nightmare, Bliss
George Neville-Neil, Telle Whitney
http://queue.acm.org/detail.cfm?id=644265

The Reincarnation of Virtual Machines
Mendel Rosenblum
http://queue.acm.org/detail.cfm?id=1017000

Blurring Lines Between Hardware and Software
Homayoun Shahri
http://queue.acm.org/detail.cfm?id=644267

References

1. Adve, S. Data races are evil with no exceptions. *Commun. ACM 53*, 11 (Nov. 2010), 84.
2. Bergeron, J., Cerny, E., Hunter, A. and Nightingale, A. *Verification Methodology Manual for SystemVerilog.* Springer, 2006.
3. Burger, D., Emer, J., Hoe, J. C., Chiou, D., Sendag, R. and Yi, J. J. The future of architectural simulation. *IEEE Micro* (May/June 2010), 8–18.
4. Chandy, K. and Misra, J. *Parallel Program Design: A Foundation.* Addison Wesley, Reading, MA, 1988.
5. Coussy, P. and A. Morawiec, eds. *High-Level Synthesis: from Algorithm to Digital Circuit.* Springer, 2008.
6. Dijkstra, E.W. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM 18*, 8 (Aug. 1975), 453–457.
7. Edwards, S.A. The challenge of synthesizing hardware from C-like languages. *IEEE Design and Test of Computers 23*, 5 (2006).
8. Hoe, J. C. Operation-centric hardware description and synthesis. Ph.D. thesis, MIT, 2000.
9. IEEE. 2002. IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-1993.
10. IEEE. 2005. IEEE Standard for System Verilog—Unified Hardware Design, Specification and Verification Language. IEEE Std 1800-2005.
11. IEEE. 2005. IEEE Standard SystemC Language Reference Manual. IEEE Std 1666-2005.
12. IEEE. 2005. IEEE Standard Verilog Hardware Description Language. IEEE Std 1364-2005.
13. Klop, J. *Term Rewriting Systems*, vol. 2. Oxford University Press, 1992, 1–116.
14. Lamport, L. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers.* Addison-Wesley Professional (Pearson Education), 2002.
15. Lee, E. A. 2006. The problem with threads. *IEEE Computer 39, 5*, 33–42.
16. Metayer, C., Abrial, J.-R. and Voisin, L. May 31, 2005. The Event-B Language;. http://rodin.cs.ncl.ac.uk/deliverables.htm.
17. Milner, R., Tofte, M., Harper, R. and MacQueen, D. *The Definition of Standard ML* (revised). MIT Press, Cambridge, MA, 1997.
18. Nikhil, R.S., Czeck, K.R. *BSV by Example.* CreateSpace 2010 (Amazon.com).
19. OSCI. 2009. OSCI TLM-2.0 Language Reference Manual; www.systemc.org.
20. Peyton Jones, S., ed. *Haskell 98 Language and Libraries: The Revised Report.* Cambridge University Press, 2003; www.haskell.org.
21. Singh. S. Computing without processors. *Commun. ACM 54*, 8 (Aug. 2011) 46–54.
22. Williams, R. 2010. Photoshop scalability: Keeping it simple. *Commun. ACM 53*, 10 (Oct. 2010), 36.

Rishiyur Nikhil (nikhil@bluespec.com) worked on Haskell and functional programming languages and compilers for parallelism, and dataflow and multithreaded architectures for about 20 years. Since 2000 he has been applying those ideas to hardware design.

DOI:10.1145/2001269.2001285

Article development led by acmqueue
queue.acm.org

Big data is about more than size, and LINQ is more than up to the task.

BY ERIK MEIJER

The World According to LINQ

PROGRAMMERS BUILDING WEB- and cloud-based applications wire together data from many different sources such as sensors, social networks, user interfaces, spreadsheets, and stock tickers. Most of this data does not fit in the closed and clean world of traditional relational databases. It is too big, unstructured, denormalized, and streaming in real time. Presenting a unified programming model across

all these disparate data models and query languages seems impossible at first. By focusing on the commonalities instead of the differences, however, most data sources will accept some form of computation to filter and transform collections of data.

Mathematicians long ago observed similarities between seemingly different mathematical structures and formalized this insight via category theory, specifically the notion of monads as a generalization of collections. Languages such as Haskell, Scala, Python, and even future versions of JavaScript have incorporated list and monad comprehensions to deal with side effects and computations over collections. The .NET languages of Visual Basic and C# adopted monads in the form of LINQ (Language-integrated Query) as a way to bridge the gap between the worlds of objects and data. This article describes monads and LINQ as a generalization of the relational algebra and SQL used with arbitrary collections of arbitrary types, and explains why this makes LINQ a compelling basis for big data.

LINQ was introduced in C# 3.0 and Visual Basic 9 as a set of APIs and accompanying language extensions that bridge the gap between the world of programming languages and the world of databases. Despite the continuing excitement about LINQ in the external

developer community, the full potential of the technology has not yet been reached. Thanks to the foundational nature of LINQ, there is still enormous potential for its mapping scenarios outside object-relational (O/R), especially in the area of big data.

The advent of big data makes it more important than ever for programmers to have a single abstraction that allows them to process, transform, compose, query, analyze, and compute across at least three different dimensions: *volume*, big or small, ranging from billions of items to a handful of results; *variety* in models, structured or unstructured, flat or nested; and *velocity*, streaming or persisted, push or pull. As a result, we see a mind-blowing number of new data models, query languages, and execution fabrics. LINQ can virtualize all these aspects behind a single abstraction.

Take, for example, Apache's Hadoop ecosystem. It comes with at least eight *external* DSLs (domain-specific languages) or APIs: a set of low-level Java interfaces for MapReduce computations; *Cascading*, a "data-processing definition language, implemented as a simple Java API;" *Flume*, a "simple and flexible architecture based on streaming data flows;" *Pig* a "high-level language for expressing data analysis programs;" *HiveQL*, an "SQL-like language for easy data summarization, ad hoc queries, and the analysis of large data sets;" *CQL*, a "proposed language for data management in Cassandra;" *Oozie*, an XML-based "coordinator engine specialized in running workflows based on time and data triggers;" and *Avro*, a schema language for data serialization.

To create an end-to-end application, programmers need to use several of these external DSLs in addition to a general-purpose programming language such as Java to glue everything together. If data comes from an external RDBMS (relational database management system) or push-based source, then even more DSLs such as SQL or StreamBase are required. Using LINQ and C# or Visual Basic on the other hand, programmers can use *internal* DSLs to program against any shape or form of data inside a general-purpose OO (object-oriented) language that comes with tooling (Visual Studio or cross-platform solutions from Xamarin such as MonoDevelop, Mono Touch for iPhone, or Mono for Android) and an extensive collection of standard libraries (.NET Framework).

Figure 1. Example pie chart.

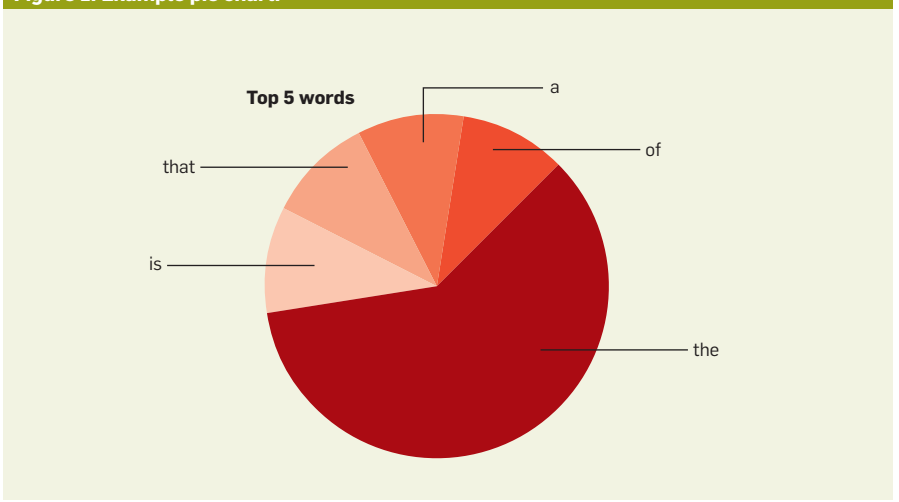
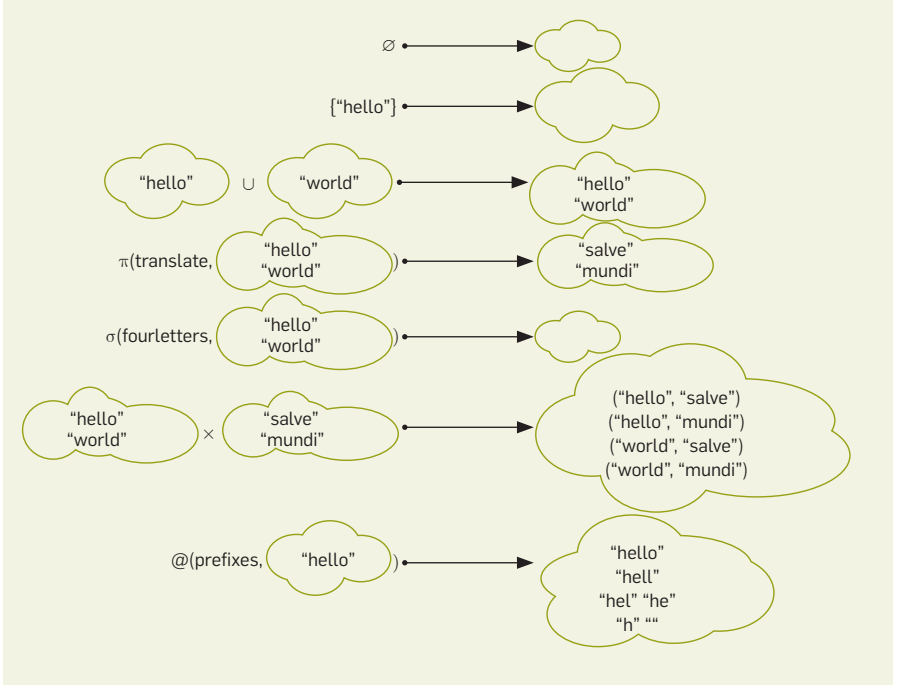


Figure 2. Relational algebra operators.



Standard Query Operators and LINQ

Assume that given a file of text—say, `words.txt`—you need to count the number of distinct words in that file, find the five most common ones, and visualize the result in a pie chart. If you think about this for a minute, it becomes clear that this is really an exercise in transforming collections. This is exactly the kind of task for which LINQ was designed. To keep things simple, we have implemented this example using LINQ to Objects to process the data in memory; however, with minimal modification the same code runs on LINQ to HPC (high-performance computing) over terabytes of data stored in commodity clusters.

The standard `File.ReadAllText` method provides the content of the file as a single giant string. You first need to chop up this string into individual words by breaking it at delimiter characters such as space, comma, period, etc. Once you have a list of words, you need to clean it up, removing all empty words. Finally, normalize all words to lowercase.

Using the LINQ sequence operators, you can transliterate the description from the previous paragraph directly into code:

```
var file = System.IO.File.
ReadAllText("words.txt");
var words = file.Split(delimiters)
          .Where(w⇒!w.IsNullOr-
          WhiteSpace())
          .Select(w⇒w.ToLower());
```

Instead of using the sequence operators directly, LINQ also provides a more "declarative" query comprehension syntax. Using comprehensions, you can rewrite the code as follows:

```
var words =
  from w in file.Split(delimiters)
  where !w.IsNullOrWhiteSpace()
  select w.ToLower();
```

Once you have converted the file into a sequence of individual words, you can find the number of occurrences of each word by first grouping the collection by each word and then counting the number of elements in each group (which contains all occurrences of that word):

```
var wordcount =
  from w in words
  group by w into group
  select new{ Word = group.Key,
              Count = group.Count() };
```

Without using query comprehension syntax, the code would look like this:

```
var wordcount = words.GroupBy(w⇒w).
                Select(group⇒
                    new{ Word =
                    group.Key, Count =
                    group.Count() };
```

To find the top five most frequent words, you can order each record by `Count` and take the first five elements:

```
var top5 = wc.OrderByDescending(p⇒
             p.Count).Take(5);
```

Now that you have a collection of the top five words in the file, you can visualize them in a pie chart, as in Figure 1. A pie chart is really nothing more than a collection of slices, where each slice consists of a number that represents the proportion of the total pie and a legend that describes what the slice represents. This means that by defining the charting API to be LINQ friendly, you can create charts by writing a query over the Google image charts API:

To truly understand the power of LINQ, let's take a step back and investigate its origins and mathematical foundations. Don't worry, you need knowledge of only high school-level mathematics.

```
var chart = new Pie(from w in top5
select new Slice(w.Count){ Legend = r.Word })
{Title = "Top 5 words" };
var image = await chart;
```

The `await` keyword is used in an unorthodox way to make the expensive coercion from `Google.Linq.Charts.Pie` into an image that requires a network round-trip explicit.

This example just scratches the surface of LINQ. It provides a library of sequence operators such as `Select`, `Where`, `GroupBy`,... to transform collections, and it provides syntactic sugar in the form of query comprehensions that allows programmers to write transformations over collections at a higher level of abstraction.

To truly understand the power of LINQ, let's take a step back and investigate its origins and mathematical foundations. Don't worry, you need knowledge of only high school-level mathematics.

Datacentric Interpretation

Relational algebra, which forms the formal basis for SQL, defines a number of constants and constructors for sets of values $\{\Sigma\}$, such as the *empty set* $\varnothing\in\{\Sigma\}$; *injection* of a value into a singleton collection $\{\_\}\in\Sigma\rightarrow\{\Sigma\}$; and the *union* of two sets into a new combined set $\cup\in\{\Sigma\}x\{\Sigma\}\rightarrow\{\Sigma\}$. There are also a number of relational operators such as *projection*, which applies a transformation to each element in a set $\pi\in(\Sigma\rightarrow\Lambda)x\{\Sigma\}\rightarrow\{\Lambda\}$; *selection*, which selects only those elements in a set that satisfy a given property $\sigma\in(\Sigma\rightarrow\mathbb{B})x\{\Sigma\}\rightarrow\{\Sigma\}$; Cartesian product, which pairs up all the elements of a pair of sets $X_{\in}\{\Sigma\}x\{\Lambda\}\rightarrow\{\Sigma x\Lambda\}$; and *cross-apply*, which generates a secondary set of values for each element in a first set $@\in(\Sigma\rightarrow\{\Lambda\})x\{\Sigma\}\rightarrow\{\Lambda\}$.

Figure 2 depicts the relational algebra operators using clouds to denote sets of values. An SQL compiler translates queries expressed in the familiar `SELECT-FROM-WHERE` syntax into relational algebra expressions; to optimize the query it applies algebraic

laws such as distribution of selection: σ(p,σ(q,xs)) = σ(x⇒p(x)∧q(x),xs); and then translates these logical expressions into a physical query plan that is executed by the RDBMS.

For example, the SQL query `SELECT Name FROM Friend WHERE Likes(Friend, Sushi)` is translated into the relational algebra expression π(f⇒f.Name,(σ(f⇒Likes(f,Sushi),Friend). To speed up the execution of the query, the RDBMS may use an index to quickly look up friends who like `Sushi` instead of doing a linear scan over the whole collection.

The cross-apply operator @ is particularly powerful since it allows for correlated subqueries where you generate a second collection for each value from a first collection and flatten the results into a single collection @(f,{a,...,z})=f(a)∪...∪f(z). All other relational operators can be defined in terms of the cross-apply operator:

```
xs X ys  = @(x⇒π(y⇒(x,y),ys),xs)
π(f,xs)  = @(x⇒{f(x)},xs)
σ(p,xs)  = @(x⇒p(x)?{x}:∅,xs)i
```

As a programmer you can easily imagine writing up a simple implementation of cross-apply: you would just iterate over the items in the input set, apply the given function, and accumulate the results into a result set. Such an implementation, however, wouldn't need its argument to be as *set* {Σ}; anything that we can iterate over such as a list, array, or hash table would suffice. Similarly, there is no reason at all that relational algebra operations should be restricted to sets of values {Σ}. They can be implemented based on other types of collections as well.

Perhaps surprisingly, there is also no reason that the operations passed into π, σ, and @ should be restricted to concrete *functions* Σ→Λ. In fact, you can use any representation of a function from which to determine which computation to perform. For example, in a language such as JavaScript you could simply pass a string and then use eval to turn it into executable code.

What you are searching for is the underlying *interface* that relational algebra implements. As long as there is a type constructor for collections M<Σ> that provides the operations that satisfy similar set-like algebraic properties as {Σ}, and a type constructor for computations Σ→Λ that satisfies similar function-like properties as Σ→Λ, you can generalize relational algebra to the following set of operators and still be able to write SQL queries over these collections by desugaring query syntax:

```
∅ ∈ M<Σ>
{ _ } ∈ Σ→M<Σ>
∪ ∈ M<Σ>xM<Σ>→M<Σ>
@ ∈ (Σ→M<Λ>)xM<Σ>→M<Λ>
```

For programmers this is just separating interface from implementation; mathematicians call the resulting structure *monads*, and instead of queries they speak of *comprehensions*.

An OO language such as C# uses the canonical interface for collections `IEnumerable<T>` as a specific instance of the abstract collection type M<T> and uses delegates Func<Σ,Λ> to represent computations Σ→Λ. By doing this, you recognize the operators from relational algebra as the LINQ standard query operators as defined in the `Linq.Enumerable` class, as shown in Figure 3.

Alternatively, you can use the `IQueryable<T>` interface to represent collections `M<T>` and expression tree `Expression<Func<Σ,Λ>>` to represent computations Σ→Λ. In that case you recognize the relational algebra operators as the LINQ standard query operators as defined in the `Linq.Queryable` class. The ability to treat code as data using morphisms—or in the C# case using the `Expression` type and lambda expressions for code literals—is a fundamental capability that allows the program itself to manipulate, optimize, and translate queries at runtime.

Instead of SQL syntax, the C# language defines XQuery-like comprehensions of the form `from-where-select`. The previous SQL query example looks like this:

```
from friend in friends where friend.
Likes(Sushi) select friend.Name
```

Just as in SQL, comprehensions are translated by the compiler into the underlying LINQ query algebra:

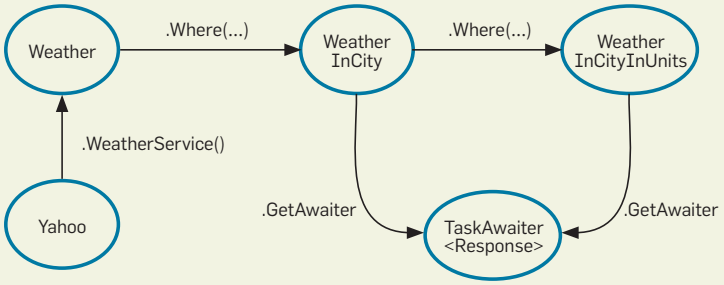
```
friends.Where(friend⇒friend.
Likes(Sushi)).Select(friend⇒friend.Name)
```

Depending on the overloads of `Where` and `Select`, the lambda expressions will be interpreted as code or data. A simplified implementation of `IQueryable` is discussed later.

Figure 3. LINQ standard query operators and relational algebra.

```
//projection π
  IEnumerable<T> Select<S,T>(IEnumerable<S> source,
                             Func<S,T> selector)
  //CROSS-APPLY @
  IEnumerable<T> SelectMany<S,T>(IEnumerable<S> source,
                                 Func<S,IEnumerable<T>> selector)
  //selection σ
  IEnumerable<T> Where<T>(IEnumerable<T> source,
                          Func<T,bool> predicate)
```

Figure 4. Yahoo weather state machine.



As already shown, monads and their incarnation in practical programming languages such as LINQ are simply a generalization of relational algebra by imagining the interface that relational algebra implements. The concepts and ideas behind LINQ should therefore be deeply familiar to both database people and programmers.

Theory into Practice

Unlike Haskell, which has incorporated monads and monad comprehensions in a principled way, the C# type system is not expressive enough for the mathematical signatures of the monad operators. Instead, the translation of query comprehensions is defined in a purely pattern-based way. In a first pass, the compiler blindly desugars comprehensions, using a set of fixed rules, into regular C# method calls and then relies on standard type-based overload resolution to bind query operators to their actual implementations.

For example, the method `Foo Select(Bar source, Func<Baz, Qux> selector)`, which does not involve any collection types, will be bound as the result of translating the comprehension

```
var foo = from baz in bar select qux
```

into the desugared expression

```
var foo = bar.Select(baz⇒qux)
```

This technique is used extensively in the example presented next.

Another difference between LINQ and its monadic basis is a much larger class of query operators including grouping and aggregation, which is more SQL-like. Interestingly, the inclusion of comprehensions in C#, which was inspired by monad and list comprehensions in Haskell, has recursively inspired Haskell to add support for grouping and aggregation to its comprehensions.

Custom Query Providers

The Yahoo weather service (http://developer.yahoo.com/weather/) allows weather forecast queries for a given location, using either metric or imperial units for the temperature. This simple service is a good way to illustrate a nonstandard implementation of the LINQ query operators that is completely specialized for this particular target and that will allow only strongly typed queries of the form

```
var request = Yahoo.WeatherService().
Where(forecast⇒forecast.City == city).
Where(forecast⇒forecast.
Temperature.In.units);
var response = await request;
```

or equivalently using query comprehensions

```
var request =
from forecast in Yahoo.WeatherService()
where forecast.City == city
where forecastTemperature.In.units
select forecast;
var response = await request;
```

The implementation of the operators extracts the city and temperature unit from the query and uses them to create a REST call (http://weather.yahooapis.com/forecastrss?w=*woeid*&u=*unit*) to the Yahoo service as a result of using the `await` keyword to explicitly coerce the request into a response.

The technical trick in this style of custom LINQ provider is to project the capabilities of the target query language—in this case the Yahoo weather service that requires (a) a city and (b) a unit—into a type-level state machine that guides users in "fluent" style (and supported by IntelliSense) through the possible choices they can make (Figure 4).

At each transition in the state machine we collect the various parts of interest of the query—in this case, the particular city and the temperature unit. In principle, the city doesn't really need to come first, but it might be more natural for the graph to allow either type of `where` clause to be specified first, but with the restriction that both `where` clauses are required. I leave the lifting of this restriction in the state machine as an exercise for the reader.

Note that none of the types `Weather`, `WeatherInCity`, or `WeatherInCityInUnits` implements any of the standard collection interfaces. Instead they represent the stages in the computation of a request that will be submitted to the Yahoo Web service, for which you do not need to define an explicit container type. What also surprises many people is that neither of the two `Where` methods actually computes a Boolean predicate. Even stranger is that each of the three occurrences of the range variable `forecast` in the query has a different type.

The `Weather` class defines a single method that picks the city specified in the query and passes it on to `WeatherInCity`, which is the next state in the type-based state machine:

```
WeatherInCity Where(Weather source,
Func<CityPicker,string> city)
{
  return new WeatherInCity{ City =
  city(new CityPicker()) };
}
```

The "predicate" in the `Where` method is a function that takes a value of type `CityPicker`, which has a single property that returns the phantom class `City` that exists only to facilitate IntelliSense and whose equality check immediately returns the string passed to the equality operator:

```
class CityPicker { City City; }
class City
{
  static string operator == (City c,
  string s) { return s; }
}
```

As a result of this, calling `Yahoo.Weather().Where(forecast⇒forecast=="Seattle")` really is just a convoluted way of creating a new `WeatherInCity{ City = "Seattle" }` instance using a `Where` method that does not take a Boolean predicate and an equality operator that returns a string.

You can use the same trickery in `WeatherInCityInUnits Where(Func<UnitPicker,Unit> predicate)`, so that calling `Where(forecast⇒forecast.Temperature.In.Celsius)` on the result of the previous filter creates an instance of new `WeatherInCityInUnits{ City = "Seattle", Unit = Unit.Celsius }`. The techniques used here are not only useful for defining custom implementations of the LINQ operators, but also can be leveraged for building fluent interfaces in general.

Since the Yahoo service requires

Figure 5. The LINQ implementation of Task<T>.

```
class YahooWeatherInCityInUnits
{
     string City; string Units; string AppID;

     TaskAwaiter<ForeCast> GetAwaiter()
     {
          var www = new WebClient();
          var response =
               from xml in www.DownloadStringTaskAsync(…City, AppID…)
               let woeid = …fish WOEID from result…
               from rss in www.DownloadStringTaskAsync(…woeid…)
               let forecast = …deserialize forecast from rss…
               select forecast;
          return response.GetAwaiter();
     }
}
```

the city as a WOEID (where on earth ID), we need to make two service calls under the hood in order to retrieve the weather forecast. The first service call retrieves the WOEID of a requested city via http://where.yahooapis.com/v1/places.q(city)?appid=XXXX. If that successfully returns, then a second call is made to retrieve the weather forecast for that location. The calls to the Web server are performed asynchronously and both return a `Task<T>` (in Java you would use `java.util.concurrent.Future<T>` to represent the result of an asynchronous operation). Since we can consider a `Task<T>` as a kind of collection that contains (at most) one element, it also supports the LINQ query operators, and we have turtles all the way down; the LINQ implementation for `Weather` is defined using the LINQ implementation of `Task<T>` (see Figure 5).

Though this is an extremely small and limited example, it clearly illustrates many of the techniques used to create real-world LINQ providers such as LINQ to Objects, LINQ to SharePoint, LINQ to Active Directory, LINQ to Twitter, LINQ to Netflix, and many more.

Generic Query Providers

The weather service query provider example is structured as an internal DSL. While this provides a great user experience with maximum static typing, it allows little room for reusing the actual implementation of the provider. It is custom built for the particular target top-to-bottom. At the other end of the spectrum we can create a completely generic query provider that records a complete query "as is," using a little bit of meta-programming magic.

In C# a lambda expression such as `x⇒x>4711` can be converted into either a delegate—say, of type `Func<int,int>`—or into an expression tree of type `Expression<Func<int,int>>`, which treats the code of a lambda expression as data. In Lisp or Scheme one would use *syntactic quoting* to treat code as data. In C# lambda expressions in combination with the type expected by the context provide a *type-based quoting* mechanism.

The class `Queryable` implements LINQ standard query operators that take expression trees as arguments and return an `Expression` representation of their selves, very much like a macro recorder as shown in Figure 6.

For example, given a value `xs` of type `Queryable<int>`, the call `xs.Select(`**`x⇒x>4711`**`)` causes the lambda expression to be converted into an expression tree (shown in bold), and then returns an expression tree that represents the call itself `xs.Select(`**`x⇒x>4711`**`)`. Now it is up to the specific query provider (such as LINQ to SQL, Entity Framework, LINQ to HPC) to translate the resulting expression tree and compile it into the target query language.

The `IQueryable`-based implementation that ships with the .NET Framework uses the same scheme as the simplified example code just shown, except that it is interface based, and it therefore relies on a second interface `IQueryProvider` to supply a factory for creating instances of `IQueryable`.

The advantage of a generic query provider is that you can offer general services such as query optimization, which implement rewrite rules such as `xs.Union(ys).Where(p) =xs.Where(p).Union(ys.Where(p))` that can be reused across many LINQ providers.

LINQ-Friendly APIs

All examples so far have dealt with implementing particular LINQ providers. An orthogonal aspect of LINQ is APIs that leverage particular LINQ implementations, often LINQ to Objects. For example, LINQ to XML is an API for manipulating XML documents that has been designed specifically with LINQ in mind, which eliminates the need for a DSL such as XQuery or XPath to query and transform XML.

The Google Chart API is a Web service that lets you dynamically create attractive-looking charts, using a simple URI (Uniform Resource identifier) scheme. The URI syntax for Google charts, however, is not very sequence friendly. For example, the URI for the earlier sample pie chart looks like this:

```
http://chart.apis.google.com/chart
    ?cht=p3&chtt=Top+5+words&chs=
    500x200
    &chd=t:21,12,7,7,6
    &chl=the|of|a|that|is
```

The problem is that the specification for the labels `(chl=the|of|a|that|is)` and the specification for the data set `(chd=t:21,12,7,7,6)` of the chart are given in two separate collections. On the other hand, to generate a pie chart using a query, you want a single collection of pairs that specify both the value and the label for each slice as in `from w in top5 select new Slice(w.Count){ Legend = r.Word }`.

In other words, to make the Google Chart API sequence friendly, you must transpose a collection of pairs `M<SxT>` into a pair of collections `M<S>xM<T>`. Functional programmers immediately recognize this as an instance of the function `Unzip∈(R→S x R→T x M<R>)→M<S>xM<T>`. `Unzip` can convert a chart that contains a sequence of slices into the URI format required by the Google Chart API by formatting the various collections using the separators prescribed by the chart service as shown in Figure 7.

Conclusion

Big data is not just about size. It is also about diversity of data, both in terms of data model (primary key/foreign key versus key/value), as well as consumption pattern (pull versus push), among many other dimensions. This article argues that LINQ is a promising basis for big data. LINQ is both a generalization of relational algebra and has deep roots in category theory—in particular, monads.

With LINQ, queries expressed in C#, Visual Basic, or JavaScript can be captured either as code or expression trees. Either representation can then be rewritten and optimized and subsequently compiled at runtime. We have also shown how to implement custom LINQ providers that can run in memory and over SQL and CoSQL databases, and we have presented LINQ-friendly APIs over Web services. It is also possible to expose streaming data so as to implement the LINQ standard query operators, resulting in a single abstraction that allows developers to query over all three dimensions of big data.

Figure 6. Class `Queryable` implements LINQ standard query operations.

```
class Queryable<T>
  {
      Expression This { get; set; }

      Queryable(){ This = Expression.Constant(this); }

      Queryable<S> Select<S>(Expression<Func<T,S>> f)
      {
        return new Q<S>
        {
           This = Expression.Call(This,"Select",new[]{typeof(S)}, f)
        };
      }
  }
```

Figure 7. Making the Google chart API sequence friendly.

```
  string CompileToUri()
     {
          var tt = Title.UrlEncode();
          var p = Slices.Unzip(
                  slices⇒slices.Select(slice⇒slice.Legend).Separated-
                  By("|"),
                  slices⇒slices.Select(slice⇒slice.Value).Separated-
                  By(","));

          return string.Format(@"http://chart.apis.google.com/chart
                  ?cht=p3&chtt={0}&chl=t:{1}&chd={2}", tt, p.First,
                  p.Second);
     }
```

Here are some convenience constructors for the types Pie and Slice, and a GetAwaiter method on Pie that triggers the compilation of the sequence to a URI and makes a Web request to Google:

```
class Pie
{
     IEnumerable<Slice> Slices; string Title;
     Chart(IEnumerable<Slice> slices){ Slices = slices; }
     TaskAwaiter<Image> GetAwaiter(){ …CompileToUri()… }
}

class Slice
{
     int Value; string Legend;
     Slice(int Value){ Value = value;}
}
```

Now you can create pie charts (and all other Google chart types) by writing LINQ queries to generate the data set in a natural way, and then await the image of the chart to come back from the call to http://chart.apis.google.com/chart:

```
var slices = from w in top5
             select new Slice(w.Count){ Legend = r.Word };
var image = await new Pie(slices){ Title = "Top 5 words" };
```

Acknowledgments

Many thanks to the Cloud Programmability team members Savas Parastatidis, Gert Drapers, Aaron Lahman, Bart de Smet, and Wes Dyer for all the hard work in building the infrastructure and prototypes for all flavors of LINQ and coSQL; to Rene Bouw, Brian Beckman, and Terry Coatta for helping to improve the readability of this article; and to Dave Campbell and Satya Nadella for providing the necessary push to actually write it.

Related articles on queue.acm.org

The Pain of Implementing LINQ Providers
Oren Eini
http://queue.acm.org/detail.cfm?id=2001564

A Conversation with Erik Meijer and José Blakeley
January 02, 2009
http://queue.acm.org/detail.cfm?id=1394137

A co-Relational Model of Data for Large Shared Data Banks
Erik Meijer, Gavin Bierman
http://queue.acm.org/detail.cfm?id=1961297

Related Reading

1. C# Query Expression Translation Cheat Sheet; http://bartdesmet.net/blogs/bart/archive/2008/08/30/c-3-0-query-expression-translation-cheat-sheet.aspx
2. Comprehensions with Order by and Group by; http://research.microsoft.com/en-us/um/people/simonpj/papers/list-comp/index.htm
3. Expressions; http://www.cs.cmu.edu/Groups/AI/html/r4rs/r4rs_6.html#SEC28
4. Google Chart API; http://code.google.com/apis/chart/image/
5. Hadoop; http://hadoop.apache.org/
6. JavaScript; https://developer.mozilla.org/en/JavaScript/Guide/Predefined_Core_Objects#Array_comprehensions
7. LINQ; http://msdn.microsoft.com/en-us/netframework/aa904594
8. LINQ to HPC; http://blogs.technet.com/b/windowshpc/archive/2011/07/07/announcing-linq-to-hpc-beta-2.aspx
9. Monads; http://en.wikipedia.org/wiki/Monad_%28functional_programming%29
10. Parallel Programming with .NET; http://blogs.msdn.com/b/pfxteam/archive/2010/04/04/9990343.aspx
11. Python; http://www.python.org/dev/peps/pep-0289/
12. Rx (Reactive Extensions); http://msdn.microsoft.com/en-us/data/gg577609
13. Scala; http://www.scala-lang.org/node/111
14. Xamarin; http://xamarin.com/

Erik Meijer (emeijer@microsoft.com) has been working on "Democratizing the Cloud" for the past 15 years. He is perhaps best known for his work on the Haskell language and his contributions to LINQ and Rx (Reactive Framework).

DOI:10.1145/2001269.2001286

Article development led by acmqueue
queue.acm.org

Avionics software safety certification is achieved through objective-based standards.

BY B. SCOTT ANDERSEN AND GEORGE ROMANSKI

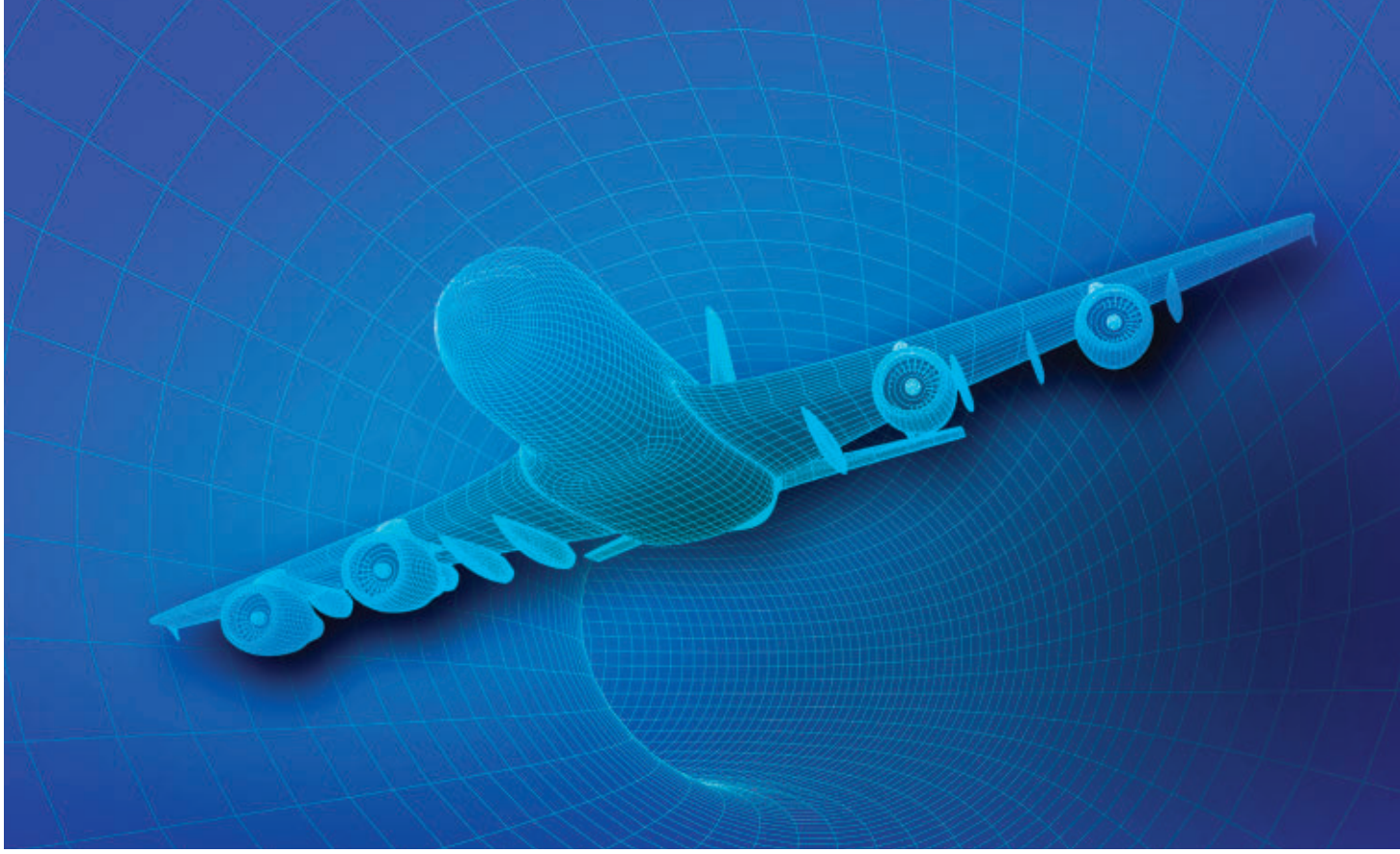
Verification of Safety-Critical Software

AVIONICS SOFTWARE HAS become a keystone in today's aircraft design. Advances in avionics systems have reduced aircraft weight thereby reducing fuel consumption, enabled precision navigation, improved engine performance, and provided a host of other benefits. These advances have turned modern aircraft into flying data centers with computers controlling or monitoring many of the critical systems onboard. The software that runs these aircraft systems must be as safe as we can make it.

The Federal Aviation Administration (FAA) and its European counterparts, along with the major airframe, engine, and avionics manufacturers worked together to produce guidance for avionics software developers culminating in the document *Software Considerations in Airborne Systems and Equipment Certification*[6] published in the United States by the nonprofit organization RTCA as DO-178B and in Europe by EUROCAE as ED-12B. The guidance in DO-178B is in the form of objectives and activities that must be met or performed to earn certification for the software product.

A safety assessment and hazard analysis helps determine the Design Assurance Level (DAL) for the software by characterizing the effects of its failure on the aircraft, crew, and passengers. There are five DALs (quoted directly from DO-178B and FAA Advisory Circular AC 25.1309-1A):[1]

- **Catastrophic**: Failure conditions that would prevent continued safe flight and landing.
- **Hazardous/Severe-Major**: Failure conditions that would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be: (1) a large reduction in safety margins or functional capabilities, (2) physical distress or higher workload such that the flight crew could not be relied on to perform their tasks accurately or completely, or (3) adverse effects on occupants including serious or potentially fatal injuries to a small number of those occupants.
- **Major**: Failure conditions that would reduce the capability of the aircraft or the ability of the crew to cope with adverse operating conditions to the extent that there would be, for example, a significant reduction in safety margins or functional capabilities, a significant increase in crew workload or in conditions impairing crew efficiency, or discomfort to occupants, possibly including injuries.
- **Minor**: Failure conditions that would not significantly reduce aircraft safety, and that would involve crew actions that are well within their capabilities. Minor failure conditions may include, for example, a slight reduction in safety margins or functional capabilities, a slight increase in crew workload, such as routine flight plan changes, or some inconvenience to occupants.
- **No Effect**: Failure conditions that do not affect the operational capability of the aircraft or increase crew workload.



DO-178B identifies a set of Software Levels Definitions A through E that roughly correspond to the DALs, with level A software the highest criticality and level E the lowest. The objectives to be met depend upon the software level assigned to the project. This article discusses activities and objectives associated with level A.

Software Engineering 101

In the preface of Nancy Leveson's *Safeware: System Safety and Computers,* the author cites "One obvious lesson is that most accidents are not the result of unknown scientific principles but rather of a failure to apply well-known, standard engineering practices. A second lesson is that accidents will not be prevented by technological fixes alone, but will require control of all aspects of the development and operation of the system."[4] Definition and control of the software development process is the first key to creating safety-critical systems.

The objectives and activities identified in DO-178B should be familiar to anyone who was attentive during software engineering courses in college. The key points are that activities and work-products are defined and repeatable. A software life cycle model must be identified, and transition criteria between processes must be documented. In short, things are written down. There are plans and standards used by software development and verification teams, and those activities are audited to ensure compliance with those plans and standards. These plans include:

- **Plan for Software Aspects of Certification (PSAC).** The primary means of communicating the overall project plan to the certification authority.
- **Software Development Plan (SDP).** Defines the software life cycle and development environment.
- **Software Verification Plan (SVP).** Describes how the software verification process objectives will be satisfied.
- **Software Configuration Management Plan (SCMP).** Describes how artifacts will be managed.
- **Software Quality Assurance Plan SQAP).** Describes how the SQA (Software Quality Assurance) function will ensure plans are followed and standards are met.

DO-178B demands three standards be present: requirements, design, and source code.

A software configuration management (SCM) system consistent with the SCMP must be provided. Additionally, an issue-tracking or bug-tracking system must be provided to document issues with either the software or compliance to standards and processes. All of these activities, plans, and standards (with the exception of the PSAC) are things that a well-run organization at SEI[3] level 2 or 3 would have in place.

No Surprises

What does it mean to be *safe*? If complete safety were defined as the complete absence of hazards, then it would be difficult to imagine any practical and useful real-world system ever being completely safe. The safeness of a particular system is, therefore, a relative notion defined by the identified potential hazards within the system, the likelihood of each hazard's occurrence, and the effectiveness of the mitigations put in place for each of those hazards.

In order to inventory and understand the hazards for a system, you must first understand what the system is supposed to do. Otherwise, how can you tell if it is doing it wrong (or what might result)? A strong set of requirements is necessary for the foundation of any safety-critical system. Further, the requirements will appear at various levels of abstraction beginning as high as the requirements for the airframe and descending into systems, line-replaceable units, CSCIs (computer software configuration items) for a particular subsystem, high-level requirements for that subsystem, and low-level requirements. The particular levels of abstraction present will depend on the particulars

of the aircraft and the project, but the presence of multiple levels of requirements and abstraction is common.

Returning for a moment to the question of what it means to be *safe*, a portion of that answer could be summed up as "no surprises." The software should perform such that all of the requirements are fulfilled. Further, there should be an *absence of unintended function*. That is, the software should do only what is specified in the requirements: no more, no less. You do not want to be learning about a hidden, extra feature of the software during an emergency at 30,000 feet.

Traceability

To say that "the system should have an absence of unintended function," prompts the question: unintended by whom? As requirements are developed at lower levels, the requirements statement must include more details and specifics. Lower-level requirements should and must provide additional value to the implementers, or they would just be restatements of their parent requirements. How can you differentiate between something that has been added to a lower-level requirement and something that is just a decomposition or elaboration of a higher-level requirement?

Managing the relationships between levels of requirements (and other project artifacts) is done through a system called *traceability*. Traceability is a mapping between requirements or other project artifacts that provides a navigable relationship between two or more items. For example, a high-level software requirement can be traced to one or more low-level requirements. Where such a mapping shows a decomposition and elaboration of the higher-level item (and no new behavior), no additional safety analysis is necessary. If a lower-level item cannot be directly traced to a higher-level item, then the possibility exists that this lower-level item has introduced an unintended function. In such a case, the lower-level (unmapped) item should be subjected to a safety assessment.

Traceability provides a means of following the mappings from the highest level of requirements down through each level of abstraction to the lowest level. From the lowest levels of software requirements traceability continues to software design artifacts, source code, verification methods and data, and other related artifacts. You should be able to start with a system-level requirement and follow each related requirement in turn through the traceability mapping down to the related code and verification data.

Reviews and Independence

Each requirement and other project artifact must be subject to review, and the review should be based on criteria identified in the project planning documents. Additionally, depending on the criticality of the software as defined by its Software Level Definition, the review of a specific artifact might need to be done with *independence*, defined in DO-178B as the "separation of responsibilities, which ensures the accomplishment of objective evaluation. For software verification process activities, independence is achieved when the verification activity is performed by a person(s) other than the developer of the item being verified, and a tool(s) may be used to achieve an equivalence to the human verification activity. For the software quality assurance process, independence also includes the authority to ensure corrective action."

The DO-178B guidance does not specify how reviews must be performed. Some organizations hold meetings with several (or many) reviewers, collect minutes, and sign the review as a group. Anyone who has witnessed or participated in a group review will likely attest that the efficacy of such a review is highly dependent upon the acumen of those doing the review and the culture of the organization holding the review. In short, the danger here is that if everybody is responsible, then nobody is responsible.

Verocel, a company that provides software verification services, takes a different approach to organizing reviews. Instead of assigning groups of engineers to a given review, it assigns a single engineer who is solely responsible for the completeness and correctness of the artifact and its review. It is not unusual for the assigned engineer to solicit assistance from other members of the group or even the originator of the artifact to answer questions, provide further analysis or insights, or obtain clarifications. In the end, however, only one signature appears on the review checklist, and responsibility for the quality of the artifact and its review rests with the one person who signed.

Mechanics of Artifact Organization and Traceability

There are practical considerations in the organization of project artifacts and the traceability between them. Often, projects have hundreds or even thousands of requirements. How should these items be managed? How can the traceability between them be maintained? Again, DO-178B provides a set of objectives but is silent on how these objectives should be met. It is up to the development team to provide details for how each objective will be met in the project planning documents, including artifact organization and traceability. The development team and the certification authority (or its representative) must agree on these plans.

Several commercial offerings are available for requirements management. Some organizations use word processors and spreadsheets (along with some specific procedures) to meet these objectives. Verocel found all of these approaches wanting and developed its own requirements and artifact management system with a relational database as its backing store. This system, called VeroTrace, manages requirements text directly and holds references to other artifacts such as design components, source files, test procedures, and test results, which are maintained in the CM (configuration management) system. VeroTrace fulfills the navigable traceability objective. Once again, DO-178B is not prescriptive on these matters. A valid organizational and traceability system could conceivably be created with little more than a stack of paper index cards. As a practical matter, large software development projects require automation of some type to manage the project artifacts, their review state, and their associated traceability.

A Good Requirement

The DO-178B objectives for a good requirement are related to: (a) compliance with system requirements, (b) accuracy and consistency, (c) compatibility with the target computer, (d) verifiability, (e) conformance to standards, (f) traceabil-

ity, and (g) algorithm aspects. DO-178B provides specifics for each of these topics. The "conformance to standards" objective suggests that there are standards with which to conform. Indeed, project planning documents will likely include standards and development guidelines for requirements, design components, coding standards, test development standards, and other aspects of software development. Having these guidelines and standards documented provides a means for SQA to assess whether the applicable processes are being followed and to demand corrective action if they are not.

The requirement development standards contain additional criteria. For example, a requirement should have a unique identifier so that it can be unambiguously referenced both in its review and by other requirements or artifacts through traceability. A requirement should have some version identifier so that a change can be recognized and impacted relationships can be assessed. The requirement author must be identified or independence of review cannot be guaranteed. The review of a requirement must identify the reviewer for the same reason. A DO-178B objective often generates a secondary group of implied activities and objectives that must be met to fulfill the original objective.

This might sound like an excessive amount of work, and it certainly is a lot of work to do correctly. The intent of this system is to provide a set of solid requirements with traceability between those of the same level of abstraction (where necessary) and between levels of abstraction to ensure the absence of unintended function. Only then can you begin to assert the software is safe.

Software should do only what is specified in the requirements: no more, no less. You do not want to be learning about a hidden, extra feature of the software during an emergency at 30,000 feet.

Waterfall

A software development process known commonly as "the waterfall model" generally follows these steps: identify all the requirements, complete the architecture and design documents, then write the code. Virtually no system of significant size can be built this way. The guidance in DO-178B acknowledges this and states the following in Section 3: "The guidelines of this document do not prescribe a preferred software life cycle, but describe the separate processes that comprise most life cycles and the interaction between them. The separation of the processes is not intended to imply a structure for the organization(s) that perform them. For each software product, the software life cycle(s) is constructed that includes these processes." Further, DO-178B Section 12.1.4d states, "Reverse engineering may be used to regenerate software life cycle data that is inadequate or missing in satisfying the objectives of this document." In short, it is important to meet all the objectives specified by DO-178B, but the order in which those objectives are met is not dictated.

That said, the FAA had certain concerns about the practice of reverse engineering and commissioned George Romanski (Verocel, Inc.) and Mike DeWalt (then of Certification Services Inc.) to research the problem. (DeWalt has since returned to the FAA as chief scientific and technical advisor for aircraft computer software.) The study, *Reverse Engineering Software and Digital Systems*, determined that 68% of those surveyed had used some sort of reverse engineering on a project. The report uncovered a number of other interesting facts, but the upshot for this discussion is that the software development process need not be "waterfall" to be successful.[2]

Validation and Verification

Validation: "Are we building the right product?" *Verification*: "Are we building the product right?"[5] The FAA commissioned the reverse engineering study because of the obvious concern that a reverse engineering effort might simply restate what the code was doing rather than determine what the code *should* be doing. Validation—determining if we are building the right product—is still an important component of the safety process, and there are no shortcuts to achieving this goal.

The reverse engineering study used data provided by Verocel from 13 projects with a total of 250,000 e-LOC (effective lines of code). (For C programs, for example, an e-LOC excludes blank lines, comment lines, lines with only a single bracket, else, or other keyword.) Problems reported within these projects were apportioned among the following categories: design error, comment error, documentation error, error-handling problems, test errors, structural coverage problems, modified functionality, requirements errors, and

code errors. Details on *how* these problems were found were also included: review, analysis (manual inspection used to reverse engineer artifacts), observation (manual inspection not related to the specific artifact being worked on), beta test/functional test, structural coverage analysis, or system test.

Of all the problems found, 77% were discovered by engineers using engineering judgment, including 54% found by analysis, 8% by observation, and 15% by review. That is, three-quarters of problems with the software were found by engineers looking hard to determine if the software is doing what it *should* do. A successful verification effort can be done only on a set of project artifacts that has been through a successful validation effort.

Software Verification

Software verification can be accomplished by any of several means or a combination. The most common means is by test. Requirements-based testing uses a set of requirements as the basis for the test criteria and produces results that conclusively report whether the software under test fulfills those requirements.

A second common verification method is *analysis*. Certain requirements are difficult or impossible to verify by test. For example, if a requirement demands that interrupts be locked and a critical section formed during a particular operation, then interrupt locking may be impossible to detect from the outside. In this case, it is appropriate to produce a small analysis document that provides evidence that this requirement is fulfilled.

Test and analysis are by no means the only verification options available. For example, *formal methods* can be used to prove correctness of an algorithm or fidelity to the software requirements. For small systems such as a pressure sensor, *exhaustive input* testing can be used to fully cover the input space for the software. Even product service history can be used in certain limited situations.

Structural Coverage Analysis

DO-178B says the following about SCA (structural coverage analysis): "The objective of this analysis is to determine which code structure was not exercised by the requirements-based test procedures...The structural coverage analysis may be performed on the Source Code, unless the software level is A and the compiler generates object code that is not directly traceable to Source Code statements. Then, additional verification should be performed on the object code to establish the correctness of such generated code sequences. A compiler-generated array-bound check in the object code is an example of object code that is not directly traceable to the Source Code."

The idea behind SCA is simple: if testing is complete and the software fulfills the requirements completely, then one would expect 100% of the code would be executed during the test. There are reasons why this might not be true, however. For example, robustness checks in the software might have execution paths that are unreachable because it is impossible under normal testing conditions to create the exceptions they guard against.

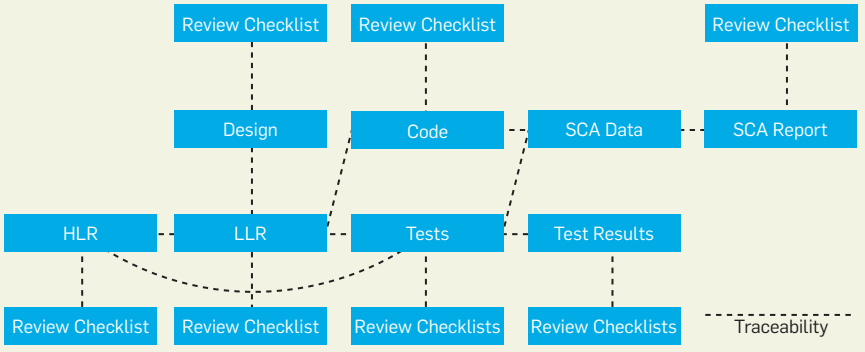
Gaps in the SCA might also indicate other problems: shortcomings in the requirements-based test cases or procedures, inadequacies in the software requirements, or *dead code,* which is code that cannot be traced to any requirements. Dead code should be removed from the system or the requirements should be updated if that function is necessary for the software.

Deactivated code is not intended to be executed during flight, although it has corresponding requirements and should be present in the software. (Perhaps this code is used only for ground maintenance activities.) It is necessary to show that deactivated code cannot be inadvertently executed in a mode where the code is not intended to run.

In addition to the activities showing coverage of source or object code in the software under test, further analysis must be performed to confirm the data coupling and control coupling between the code components. The output of the compiler is not trusted. Testing and structural coverage analysis is used to confirm that source code and object code are fully covered and any discrepancies are remedied or documented. Similarly, the linker (or any other reference fix-up mechanism) is also not trusted. Proper coupling and fix-up must be shown to be correct. SCA data along with control coupling data are reviewable artifacts.

The accompanying figure shows the web of traceability from HLR (high-level requirement) to SCA report and its review checklist. Dashed lines indicate traceability. Note that this diagram is only representative (and not complete). For example, documents used for verification by analysis are not shown. Even so, this gives a good general overview of a typical web of traceability.

Sample traceability.



Development and Verification Tools

As previously mentioned, neither the compiler nor the linker is trusted fully. These are considered *development tools* because they create artifacts that fly. It is possible to create *qualified development* tools whose output need not be checked. The guidance for the creation of such a tool (and its verification) is the same as the guidance for the creation and verification for software that flies. At times, however, the extraordinary burden of creating that full set of

verification artifacts is worth it. The output of a qualified development tool does not need to be subject to further verification efforts.

A less intense effort is required for the creation of a *qualified verification* tool. Such a tool can be substituted for a human reviewer on the targeted artifacts. Qualified verification tools require only requirements, requirements-based tests, and appropriate documentation as outlined by DO-178B. As an example, Verocel has a qualified verification tool called VerO-Link that helps with the control coupling analysis.

The Role of the DER

In the U.S. the certification authority is the FAA, but the FAA does not supply engineering resources to each project developing software for aircraft. Instead a system of DERs (designated engineering representatives) has been developed that allows the FAA to delegate responsibility for project oversight to specially trained and properly certified engineers who work within the industry, either for an independent firm or even for the manufacturers themselves. The FAA always does the final sign-off for a project, but the DERs do the lion's share of the work following the progress of the project and evaluating the materials produced by the project.

DERs assigned to the project will hold a series of four stages of involvement (SOI) meetings at various stages of the project. The first of the meetings, SOI-1, covers the plan for certification and reviews materials that are key to ensure that the project will proceed properly. This involves verifying that the processes and procedures necessary for a successful certification effort are in place, that there is a good and solid plan for meeting each of the identified DO-178B objectives, and that there is good agreement between the DER and the software developer on all important aspects of the project. The point of the first meeting is to ensure that a plan for certification is in place and that everyone agrees that if the plan is followed and the artifacts are produced, then there should be no impediments for achieving certification.

A second meeting, SOI-2, reviews any open issues from the first meeting, assesses the impact of any variations from the plan, and explores the quality of the materials developed thus far in the project. The minimum project state for having a successful SOI-2 is 50% of the requirements developed and reviewed, 50% of the design artifacts completed and reviewed, 50% of the source code completed and reviewed, and traceability between all of these artifacts. The purpose of this meeting is not to assess or review the completed set of requirements or other artifacts but to identify any problems early in the process so that corrective action can be taken while changes are relatively inexpensive and minimally disruptive. The volume of materials required to be ready for such a meeting may be more or less than the 50% identified here, as it is completely between the DER and the development team to set goals and milestones.

A third meeting, SOI-3, takes place when development is complete and the verification effort is approximately 50% complete. Verification can be done by any number of mechanisms including test, analysis, formal methods, or any other approach described and approved within the project plans for aspects of certification.

The final meeting, SOI-4, happens when all certification evidence has been produced for the completed project, and the review is primarily to assess the readiness of the package for final certification assessment.

A DER is in an adversarial role. He or she is also in a position to be an asset, assisting the project team in identifying flaws in their plans, defects in their processes, or even as a facilitator who can help the project get out of a bind by making suggestions or identifying options that the team has not identified themselves. A DER should not be placed in a position of developing process plans; the DER's job is to evaluate and not to develop materials that he or she would later be evaluating. A good DER is tough as nails and helps ensure the project produces software of high integrity.

Wrapping Up

DO-178B is not prescriptive, but it is comprehensive in its criteria and objectives. Not all of what DO-178B demands can be captured in a relatively short article, but we have highlighted the main points about how DO-178B provides guidance to help produce safe software. Good requirements, vetted to ensure they match the intent of what the software should do, must be developed. These requirements and all software development artifacts (including designs, code, tests, test results, and structural coverage data) are connected through a web of navigable traceability that helps ensure each of the requirements is fulfilled by the software and that there is no unintended function. On the other end, structural coverage analysis identifies code not covered by requirements-based tests, and dead code (code not associated with any requirements) can be flagged for removal.

All of this is presented with a backdrop of solid and documented engineering practices and a mature software development environment. With proper procedures and controls in place, it is possible to create software upon which one would trust their safety and the safety of others.

Related articles on queue.acm.org

Interview: A Conversation with Jim Ready
http://queue.acm.org/detail.cfm?id=644261

There's Still Some Life Left in Ada
Alexander Wolfe
http://queue.acm.org/detail.cfm?id=1035608

Best Practice (BPM)
Derek Miers
http://queue.acm.org/detail.cfm?id=1122688

References

1. Federal Aviation Administration. *System Design and Analysis*, Advisory Circular AC 25.1309-1A (June 21, 1998).
2. Federal Aviation Administration. *Reverse Engineering Software and Digital Systems*, April 2011 [Draft Report]. To be published by the FAA William J. Hughes Technical Center.
3. Humphrey, W.S. *Managing the Software Process*. SEI Series in Software Engineering. Addison-Wesley Publishing Co., Reading, MA, 1990.
4. Leveson, N. *Safeware: System Safety and Computers*. Addison-Wesley Publishing, Reading, MA, 1995.
5. Rakitin, S.R. *Software Verification and Validation for Practitioners and Managers*, second ed. Artech House, Norwood, MA, 2001.
6. RTCA. *Software Considerations in Airborne Systems and Equipment Certification* (DO-178B), 1992.

B. Scott Andersen is a principal software engineer for Verocel, Inc. He has more than 30 years of experience in the software industry, most recently as a member of the Jini development team at Sun Microsystems helping to build a distributed computing model for Java. .

George Romanski is a co-founder and president of Verocel, Inc. He has specialized in the production of software development environments for the past 40 years. His work has focused on compilers, cross-compilers, runtime systems, and tools for embedded realtime applications.

DOI:10.1145/2001269.2001287

Expect more cyberwarfare on the conventional battlefield and less against civilian infrastructure ...assuming containment is possible.

BY JOHN ARQUILLA

From Blitzkrieg to Bitskrieg: The Military Encounter with Computers

WARFARE IS NOT just a matter of hurling mass and energy at one's enemies; it is also about gaining an "information edge." In ancient times the emergence of writing allowed for battle orders that could guide subordinates at a distance, making possible greater operational complexity and enhancing the importance of skillful generalship. In the Middle Ages the Mongol Arrow Riders, a Pony-Express-like messenger system, coordinated movement of armies across vast distances and contributed to the startling victories that created a "nomad empire" from East Asia to Central Europe. In the Napoleonic era, 1790–1815, the British Navy's Popham signaling system allowed transmission and receipt of more, and far more complex, information than opposing fleets could muster. During the period from the American Civil War through the German wars of unification and on to World War I, telegraphy supported the deployment choreography of masses of rail-mobile troops.

A generation later in World War II, maturing radio capabilities played a key role in coordinating the German armored blitzkrieg on land and the U-boat wolf packs at sea, the latter nearly starving Britain into submission. In that conflict, fast-moving panzer divisions and far-flung submarine squadrons, each guided to their objectives and commanded in battle from great distances, completely revolutionized fighting doctrine. Radio reports by spotter planes also played a key role in empowering aircraft-carrier operations, allowing some naval battles to be conducted without opposing ships ever coming into visual range of each other.

Early computers also debuted during World War II, helping enable breakthroughs in many areas, including ballistics, but made their most important contribution in codebreaking.

» key insights

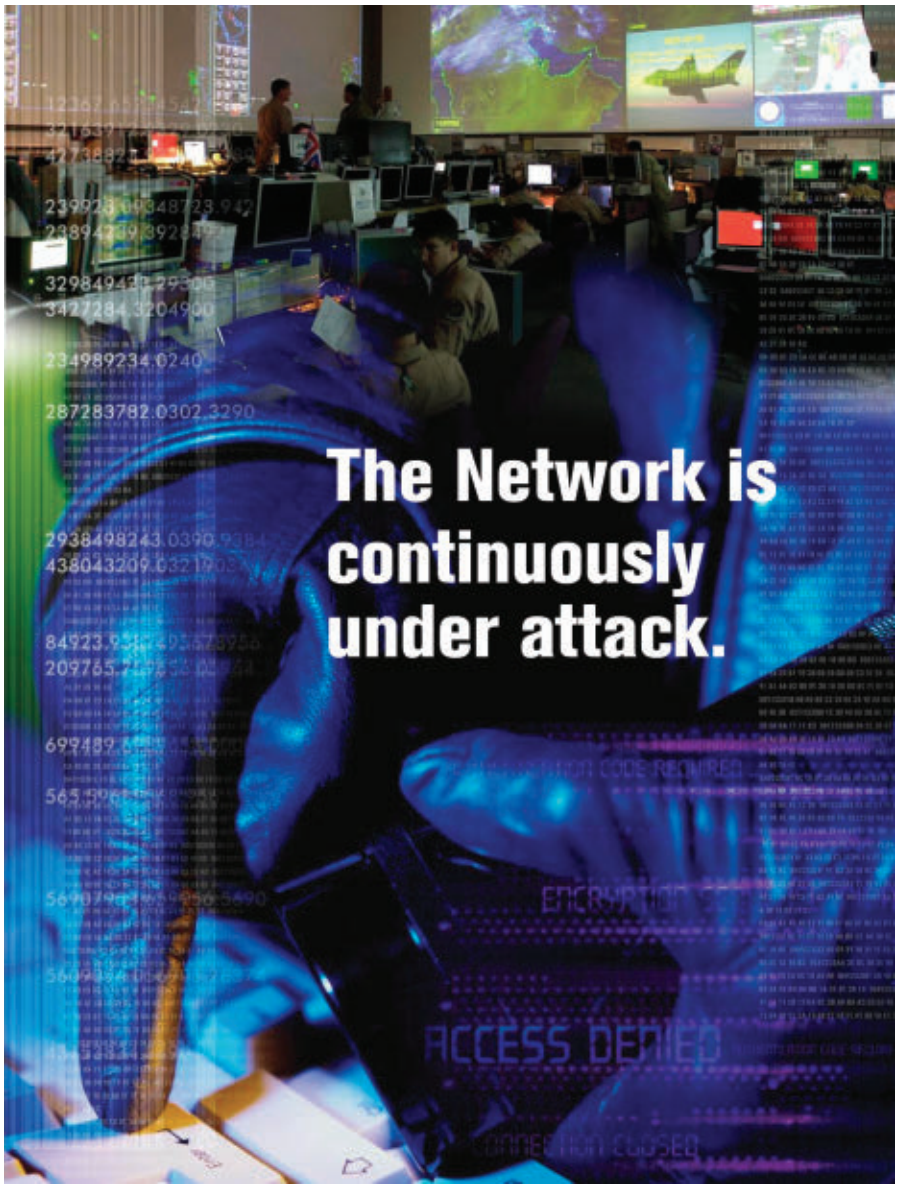
- **Militaries have been greatly empowered by a range of advanced information systems but are so dependent on them their disruption would have crippling effects.**
- **The rise of cyberwarfare should impel a reexamination of classical just-war ethics, given its relative ease of use as a "first resort" and the tempting prospect of being able to achieve national aims with less-bloody use of force.**
- **To head off a virtual arms race, behavior-based cyber arms control should be explored, including multilateral agreements to refrain from targeting civil infrastructures and pledging "no first use" of such tactics.**

Indeed, the ability of Britain's "Ultra" and its American counterpart "Magic" to decipher, respectively, German and Japanese codes made all the difference in the first few years of the war when the Allies often had to engage Axis forces from a position of material inferiority. Later, when the tide had turned, codebreaking enabled Allied victories with fewer casualties due to foreknowledge of enemy intentions.

The age of computers in battle that has unfolded over the past 70 years has proved similar to earlier eras in military history, with these new informational tools pointing to new practices. Today, computers serve not only to guide weapons and break codes but also to winnow vast amounts of battle-related information in the search for insight while facilitating lateral communications, or contact with fellow field units, not just with distant commanders. It is this super-empowerment of those who actually conduct the fighting that most distinguishes our era of informational advances from earlier ones.

An example is the triumph enabled by some 200 American Special Forces soldiers in Operation Enduring Freedom they and local allies conducted in Afghanistan in late 2001. The Green Berets rode and fought in the immediate company of a few thousand friendly Afghans, part of a larger nominal force of perhaps as many as 40,000 fighters—until then, on the losing end of a civil war in which 95% of the country had been ceded to the Taliban.[37] The Special Forces and those accompanying them were opposed by upward of 70,000 Taliban and Al Qaeda fighters who had already shown resilience in the face of a month of American-led aerial bombing.[7] But "the 200" had a secret weapon: the tactical Web page.

Originally designed with the idea of simply allowing these soldiers to order supplies, the Web page quickly became their preferred means of communicating timely, targetable information among themselves. The resulting effect was that the teams could act more swiftly and knowledgeably and give attack aircraft supporting them far greater potency, due to what has been described as their "faster, unfiltered flow of data."[9] In just a few weeks the enemy was driven from power and out of the country by an omnidirectional assault best described as a "swarm." That this success was eventually squandered, allowing the Taliban to mount an insurgency of its own, is more a function of the return to traditional command arrangements and concepts than of any fundamental flaw in network-style operations.



Poster created by U.S. Department of Defense Cyber Strategy.

Several years after the initial takedown of the Taliban, another kind of military network emerged, this time in Iraq, where vicious insurgent action was under way. While senior American generals and Pentagon officials were having difficulty mastering this challenge, junior officers doing most of the actual fighting crafted a way to share their "lessons learned" and best practices. Through a Web site called companycommand.com, initially open only to company commanders, good ideas were quickly diffused throughout the force, sharply improving counterinsurgent practices.[5] Sadly, out of ostensible security concerns, the Web site was soon subjected to high-level oversight that had a chilling effect on the willingness of junior officers to freely share their thoughts. Still, another aspect of the power of IT-enabled networking had been demonstrated.

Every day it grows clearer that networks, best described by philosopher-technologist David Weinberger's el-

IMAGE COURTESY OF U.S. DEPARTMENT OF DEFENSE

egant phrase, "small pieces, loosely joined," comprise the organizational form most empowered by computers, the Internet, and the Web. Their strength comes from their great lateral connectivity and the kind of "collective intelligence" that arises when many share their thoughts and build on one another's ideas. Over the past two decades the world has seen networks rise, with civil-society movements toppling authoritarian regimes in a series of social revolutions, most recently in the "Arab Spring" of 2011. But terrorists and transnational criminals, the principal "uncivil society" actors of our time, have benefited from networking, too, demonstrating that the new tools may serve the darkest of purposes, as Al Qaeda has shown the world. Which of networking's Janus-like faces will prevail in the future?

The Path Since the 1970s

Given the wide-ranging effects of the computer revolution on the larger issues of society and security, it is not surprising that military affairs are also profoundly affected by computerization and networking. But this reshaping has emerged only in fits and starts, with halting progress. Difficulties in Afghanistan following the initial triumph enhanced by a tactical Web page and the fate of companycommand.com are dramatic examples of the problem. But so, too, was the notion that electrons transmitted through information systems no longer simply communicate, but were becoming actual weapons. This idea, introduced by Thomas Rona, a science advisor to the Defense Department, in his seminal 1976 think piece, "Weapons Systems and Information War,"[31] was a breakthrough concept. Yet for the next 20 years, Rona's notion was simply folded into existing strands of strategic thought, leading to information warfare being equated with either strategic air power or nuclear war.

At this time, defense researchers, still steeped in Cold War military doctrines, became deeply attracted to the idea of mounting crippling attacks on adversaries without first having to engage and defeat their sea, air, and land forces. To the extent there was debate, it was about whether "strategic information warfare," as it would come to be called,[25] would look more like the sustained aerial-bombardment campaigns of World War II and later Korea and Vietnam or be thought of in terms of the massive effects that would accompany nuclear exchanges or widespread use of chemical and biological weapons, as international security expert Walter Laqueur argued in the 1990s.[22] That such "mass disruption" could be achieved without significant loss of life made strategic information warfare irresistible.

Around the same time (early-1990s) my RAND Corporation colleague David Ronfeldt and I introduced our concept of "cyberwar," which was far less about using computer viruses to attack other societies than about gaining an information edge over adversary military forces in battle.[2] Our view was based on the belief that information warfare as a form of strategic attack would have as poor a record of success as aerial bombardment, which has seldom achieved its hoped-for goals.[28]

Instead, advanced information systems had simultaneously empowered and imperiled modern militaries, opening new operational possibilities but at the same time making armies, fleets, and air forces vulnerable to disruption. Small but better-informed forces could thus defeat larger, less-well-informed, enemies, much as the heavily outnumbered U.S. naval forces outfought the Imperial Japanese fleet at Midway in 1942 by knowing more about their adversary's dispositions and intentions. Fanatical courage, luck, and timing were important factors in this battle, but the ambush of the Imperial Japanese Fleet could not have happened without an initial information edge. So a key defense research agenda was identified, one aimed at understanding the material effects of "knowing more."

U.S. sailors assigned to Navy Cyber Defense Operations Command at Joint Expeditionary Base, Little Creek-Fort Story, VA, responsible for monitoring, analyzing, detecting, and responding to unauthorized activity within U.S. Navy information systems and computer networks.

U.S. NAVY PHOTO BY MASS COMMUNICATION SPECIALIST 2ND CLASS JOSHUA J. WAHL/RELEASED

Doctrinal Debate

Conflict between the two major competing concepts—strategic information warfare as launching "bolts from the blue" and cyberwar as doing better in battle—was inevitable. Since the 1990s, a kind of "war of ideas about the idea of cyberwar" has been waged, with each side landing telling blows. The military proponents of what journalist James Adams once called the "strategic attack paradigm"[1] have been dominant in the discourse, skillfully using the threat of this kind of assault—in the hands of hostile nations and/or networks—to drive national-security debates in countries around the world and generate huge budgetary support for protection of their "critical information infrastructures." Much as the still-frightening specter of nuclear

strikes sparked the rise and persistence of myriad well-funded ballistic-missile defense initiatives, the notion of a "digital Pearl Harbor," mounted perhaps by cyberterrorists, has ensured the future of an entire industry devoted to thwarting a massively disruptive virtual attack.

The alternative emphasis, on exploring the implications of the information revolution for battle, has gained less traction. In part this is the result of a military mind-set still steeped in the Powell Doctrine of "overwhelming force" and its aerial homunculus, "shock and awe" bombing. Commanders much prefer to flatten enemy forces with brute force if at all possible, fearing that subtler approaches enabled by advanced information systems will either have less effect or be too difficult to implement.

The same commanders express concern that growing dependence on such systems could have crippling effects should they be disrupted, whether by logic bombs or well-placed physical bombs. This last worry—about kinetic weapons—is a subtle echo of early reservations, largely dispelled, as to whether computers would ever actually be rugged enough to function in the field.[6,17]

Despite such old habits of mind, attempts have been made to articulate innovative ideas that would improve military effectiveness. The best known is the concept championed by the late Vice Admiral Arthur Cebrowski, "network-centric warfare." Cebrowski, a Vietnam-era fighter pilot with an advanced degree in computer science from the Naval Postgraduate School, envisioned an interconnected, lattice-like set of "sensor and shooter grids" that would share information swiftly and widely with the largest possible number of combat elements. Introduced in 1998, this notion has sparked much discussion but has not been implemented widely or systematically.[11] Another senior naval officer, Admiral William Owens, propounded his own views of a highly networked "system of systems" intended to function in similar fashion.[27]

In the late-1990s, while Cebrowski and Owens focused on organizational redesign along networked lines, Ronfeldt and I, interacting with them regularly at the time, shifted our own focus to developing the kind of battle doctrine implied by a force that would have better access to information than ever before. We came up with the notion of "swarming," or simultaneous assault from many directions, as the most effective means by which a well-informed network comprised of many small units could strike at its foes, whether large and traditional or small and irregular.[3]

As with Cebrowski's network-centric warfare and Owens's system of systems, swarming has been embraced only fitfully. Today it languishes in a virtual purgatory alongside "autonomous" combat systems, or weapons wielded by artificial intelligence, since swarming has been branded as best-suited to application by silicon-based intelligence.[34] As for autonomous systems, the notion of unleashing robotic weapons has been seriously studied since the 1980s,[4] a period of major technical advances in the field. Nevertheless, there remains a high barrier to change here, posed largely by human self-interest (such as pilots' fear of and resistance to replacement by robots) that slows their progress. Other concerns have to do with the possibility that robots would unwittingly inflict serious collateral damage, making it more difficult to fulfill the ethical imperative to always do one's best to wage war "justly." But now, with a swarming doctrine to guide field operations, military commanders have at least a vision of how a skillfully blended future force of humans and intelligent machines could operate effectively and ethically. All that may be necessary to make this leap would be to overcome tradition-bound organizational inertia. Given the great strains on U.S. service members from repeated deployments over the past decade, robots may soon be more welcome in the force.

The Offense-Defense Balance

In a period in which battlefield-based cyberwar has made only isolated gains, the bureaucratic triumph of the strategic attack paradigm has spawned what can only be called a "cyber defense initiative," an information-age counterpart to the Strategic Defense Initiative missile-interceptor program introduced by President Ronald Reagan in 1983. While the mix of institutional actors involved is more diverse than in the nuclear realm, the U.S. military still plays a central role in thinking through the problems associated with ensuring the continued functions of its forces in the field and the infrastructures upon which its citizens depend. The same is true with regard to infrastructure protection in many other developed countries, where several military cyber corps have sprung up.

A key problem that has plagued cyber defense is that there was, and continues to be, far too little debate over alternative paradigms. The dominant view was, and still is, tethered to a kind of preclusive security based on firewalls capable of distinguishing friendly "self" from hostile "other." The problem with this Maginot-Line-like approach is that firewalls are, for the most part, capable of recognizing only things they already know, whether hostile or friendly. They are not as good at dealing with new wrinkles.

There have been repeated serious intrusions into sensitive defense information systems in the years immediately before, as well as since, the 9/11 attacks on America, little of which can be discussed openly. These events, known to the public under such names as "Moonlight Maze," which may be linked to Russia, and "Titan Rain," which may involve China,[39,40] provide stark proof of the limitations of the Maginot Line mind-set. This is not only true of the military "infosphere"; commercial firms tend to follow the military model of preclusive security. Here, too, the news is troubling, if even more difficult to obtain in detail. But the reality is that leading corporations around the world are hemorrhaging intellectual property, as hackers tap their creative veins and bleed them of their precious information resources.

The alternative to a primarily firewall-dependent information-security model is to accept that intruders will almost always access the system, no matter how nominally secure, but by strongly encrypting the data within, a defender can deny the attacker/exploiter the advantage of having gotten inside. Dorothy Denning, a leading computer scientist and professor of defense analysis at the Naval Postgraduate School, has summed up the case for defense dominance via encryption: "If the

key length is sufficiently long, it is not feasible to test each and every key. In practice, the strength of a system needs only to be commensurate with the risk and consequence of breakage."[15]

Those who believe "data at rest is data at risk" envision the additional security option of breaking encrypted data into pieces and sending them out into "the cloud," or into cyberspace beyond one's own system, ready to be called back and reassembled at a keystroke. Strong crypto and the cloud are gaining attention, but the firewall-based model remains dominant, especially with military- and national-security-related information systems.

Thus the fear of a crippling "bolt from the blue" cyberattack is great, and the U.S. military's frenetic efforts to cope with such a possibility have sparked a return in some military circles to the classic question of whether offense or defense is "dominant." In every period of major technological change there has been sharp debate about the properties of the new tools of war, and the conclusions drawn have quite often been wrong.[29] For example, before World War I, most Western generals believed machine guns and high-explosive artillery would favor the offense. They were tragically wrong. Some millions of soldiers marched shoulder to shoulder to slaughter in that war.[35]

A generation later, at the outset of World War II, the prevailing belief, except within small circles of military mavericks, was that defense was dominant. This mind-set led to such initiatives as the massive investment in the French Maginot Line. Wrong again. Aided by mechanization, the Germans simply went around the wall and scored one of history's signal military victories in the spring of 1940. It seems that figuring out the state of the offense-defense balance, in light of the latest technological changes, has generally proved quite difficult. Today is no exception.

Security System, Attack Tool

Assessing the balance of power in battle is just as difficult to parse in the virtual realm as it has been in the physical realm. To date, the school of thought associated with notions of offense dominance in cyberwar has been ascendant, feeding the frenzy to craft defenses.[12] But articulate dissenters have also been heard from, in particular the RAND Corporation's Martin Libicki, who believes it will be difficult for cyberspace-based offensives to achieve strategic effects. As he sees it, cyberwarfare "is still largely theoretical. People have seen the detritus left behind by small-scale hacker attacks, but no one has ever seen it work at the scale often claimed for it."[23]

Even so, a theoretically superior defensive concept can still lead to wrongheaded implementation, engendering great vulnerabilities. A case in point is the Navy Marine Corps Intranet (NMCI), a classic preclusive security attempt—in the form of the world's largest intranet—to make intrusions into the sea services' information systems virtually impossible. From the outset, NMCI proved vulnerable to a range of threats, none openly acknowledged, beyond admission that one particular computer virus, a variant of MS/Blast, made its way into and throughout much of the system.[30]

Viral attacks, based on malicious software that attaches to programs or documents, have grown in sophistication and stealth, as have "worms," self-replicating programs that can even cause disruptive effects in the physical world. A recent and very troubling example of the latter is the Stuxnet worm, malicious software specifically designed, it appears, to exploit vulnerabilities in Siemens industrial control systems components in Iranian high-tech (possibly nuclear-weapons proliferation-related) equipment.[10,32]

Stuxnet is especially interesting in that it has apparently succeeded in disrupting systems not connected to the Internet, suggesting insertion of the worm may have occurred via any of a range of components, possibly through something as simple as an infected USB drive. If so, the "reach" of cyberweaponry may have to be reckoned as far greater than previously thought. The implication is that a vast range of technical components—many of them "off-the-shelf" imports—should be seen as potential conduits for attackers. Awareness of this threat has grown and, in the American case, led the military to develop a significant capacity for ensuring "supply chain security."[21]

This discussion—from Libicki's analysis to the Stuxnet example—suggests the offense-defense balance in this era may be characterized by an action-reaction cycle in which one or the other mode of war becomes temporarily ascendant. It may be much like technical and tactical developments in traditional military affairs, often favoring the attacker or defender when introduced, but which are eventually countered. For example, the World War II German U-boat wolf-pack offensive was ultimately defeated by a mix of skillful codebreaking and improved direction-finding equipment, unmasking the attackers' positions and giving the edge to the defense.

Likewise, viruses, worms, and new forms of "semantic attack" on information systems will likely be subject to technical countermeasures that will diminish, if not dispel, the threats they pose, particularly if firewall-oriented "Maginot Line mind-sets" give way to greater emphasis on strong cryptography and data being moved around much more, not just deposited for long periods in fixed locations.

Recent Cyberwars

In April and May 2007 a series of widespread cyberattacks was mounted anonymously against Estonia, sparked by removal of a World War II monument to Soviet soldiery (commemorating the Russian military campaign and its casualties suffered driving the Nazis from the country) from a prominent place in the capital, Tallinn. Outrage among Russians at this action was followed by massive cyberattacks thought to have been perpetrated, or at the least encouraged, by Russian leaders against the Estonian government and civil society. Huge disruptions ensued for a short period, with the attackers using simple tools in distributed denial-of-service attacks.[8] It was a clear example of the "strategic attack paradigm"; a scaled-up version of this sort of campaign launched against, say, the U.S. or other developed country would have inflicted enormous economic losses.

In August 2008 the Russian military launched an invasion of the trans-Caucasian Republic of Georgia, a U.S. ally whose security forces had been nurtured, trained, and equipped along

lines amenable to the Pentagon. Unlike the difficulties they had experienced fighting in nearby Chechnya, Russian troops this time sliced through Georgian defenses. (The Russians were joined in the field by Ossetian irregulars, though they did not form the advance shock troops in this particular war.) Among the factors contributing to Russian success were skillful cyberattacks mounted in conjunction with field operations, making this a battle-oriented cyberwar rather than a standalone virtual strategic offensive against infrastructure. The degree of disruption to Georgian command-and-control systems achieved by hackers (use of cyberattacks has still not been acknowledged by Moscow) was startling. Again, were similar effects scaled up against a U.S-size military, they would likely achieve catastrophic levels of disruption.

The Estonian and Georgian cyberwars both seem to support the notion that we are entering an era of offense dominance. Whether the intent is to use computers and cyberspace for mounting strategic attacks on other societies or to provide "virtual supporting fire" in force-on-force battles in the field, preventing such assaults is likely to prove problematic. They may also prove difficult to contain, at least for a while. One implication is these events could herald a period of constant cyber conflict in which cyberwars are always under way somewhere; another is that the ease of mounting such attacks will be offset by retaliatory threats or mutual agreements to refrain from doing so. Indeed, both notions of "controlling cyberwar" have been considered in recent years.

Strong crypto and the cloud are gaining attention, but the firewall-based model remains dominant, especially with military- and national-security-related information systems.

Deterrence and Arms Control

It is interesting, and somewhat ironic, that the Russians appear to be on the cutting edge of cyberwarfare, as both a form of strategic attack and mode of battle. The irony comes from the fact that, at least since the mid-1990s, Russia has been trying to make the world less permissive of this kind of conflict by bringing older concepts of deterrence and arms control into the information age. For example, an early, and very blunt, Russian attempt at deterrence came in 1995 in the form of an alarming statement from information warfare expert V.I. Tsymbal, as reported by military analyst Tim Thomas: "Moscow's only retaliatory capability [to cyberattacks] at this time is the nuclear response."[38]

Tsymbal's formulation spoke to what is called the "punitive" dimension of deterrence, or the belief that, even when defenses are poor, a capacity for devastating retaliation can prevent attacks from being mounted in the first place. An early nuclear strategy, the Eisenhower-era U.S. doctrine of "massive retaliation" with atomic weapons against any form of aggression, even on a small scale, is a classic example of the punitive approach. However, the exceedingly disproportionate nature of the threat undermined its credibility from the outset, causing the policy, in the phrasing of strategic analyst and Nobel laureate Thomas Schelling, to be "in decline almost from its enunciation in 1954." However, its successor concept, advanced in the 1960s, "mutual assured destruction" (MAD)—all-out retaliation in the event of a nuclear attack—has fared better and remains, even today, the foundation of American strategic-deterrent thought.

In the cyber realm, it seems that some informal variant of MAD may already be in place to deter strategic attacks on infrastructure, the twist being that the concept is now "mutual assured disruption," not destruction. Where many advanced militaries are hardly likely to be deterred from waging cyberwar in the field against their adversaries, developed countries are clearly aware that their information systems are and will remain vulnerable to attack, so considerable circumspection is the apparent norm when it comes to the strategic-attack paradigm. To be sure, many cyber-spying intrusions occur worldwide on a daily basis but are not attacks per se and do little or no damage to operating systems.

Key problems for cyber deterrence are that attacking nations may keep their identities secret, and not all attacks emanate from other nations. On the latter point, nations may be attacked by networks of non-state actors (such as terrorists and transnational criminal syndicates) or even super-empowered individuals. When it comes to cyberwarfare of this strategic sort,

a network could easily slip the bonds of mutual deterrence simply because it proves difficult, if not impossible, to figure out against whom to strike a retaliatory blow. The same is true if networks ever get their hands on nuclear weapons, though for now the notion of a network having its own "nuclear Napoleon" remains just a far-off possibility. It is the imminent threat posed by hacker networks (and perhaps terrorist groups) that already possess or are developing capacities for mounting mass disruptive cyberattacks that demands attention. Against them, the only measure likely to work will be based on the notion of what experts call "denial deterrence," the ability to convince malefactors that they are wasting their time with such attacks, as the likelihood of success is low.[26]

Given the great edge conveyed by being able to launch cyberattacks from behind a veil of anonymity, denial-based deterrence is to be preferred when confronting the threat from networked actors. But when it comes to other nations, the hope remains that they can be identified when they perpetrate attacks, and that punitive retaliatory threats can work to stop them.[24] However, the ambiguity as to the perpetrator(s) of the Stuxnet attack suggests the veil of anonymity may remain difficult to pierce.[19] Thus, uncertainty abounds, leading to efforts to cultivate another Cold War concept: arms control. Though clear that almost all information technologies can be "weaponized" for cyberwarfare, and trying to control their spread is futile, there is some hope of fostering a behavior-based norm of restraint by "emphasizing dialogue with like-minded nations," as former CIA director General Michael V. Hayden explained earlier this year.[20]

This notion, called "operational arms control," reflects success in both the biological and chemical weapons conventions, formally adopted in 1975 and 1997, respectively, that have done much to limit development and use of these mass-destructive weapons. The question today is whether such behavior-based controls can make a similarly beneficial contribution, inducing those nations, perhaps even some networks, to refrain from using them, at least against civilian infrastructures.

The Estonian and Georgian cyberwars both seem to support the notion that we are entering an era of offense dominance.

However, as with deterrence, it is difficult to see how such controls would be imposed on the battlefield, given that cyberwar applied in a military-on-military fight would likely convey an advantage to the better practitioner, bringing about a swifter, less-bloody end to the fighting. The answer to the question about cyber arms control and cyberwarfare is perhaps to be found in the ethical domain.

Just, Unjust Cyberwars

Classical ethical formulations about conflict address both going to war justly (acting in self-defense or fighting only as a last resort) and waging war morally (using force only in proportionate ways and refraining from inflicting harm on noncombatants).[41] Cyberwarfare puts considerable strain on both aspects of just-war ethics. The very ease of offensive action encourages redefining "defense" in preemptive or even preventive terms[a] and makes going to war in this way attractive as an early option rather than as a last resort. In terms of fighting justly, cyberwarfare, with its disruptive rather than destructive effects, hardly seems likely to qualify as "disproportionate," either as a form of strategic attack or on the battlefield.

However, no one is able to predict effects across cyberspace, and the prospect of inflicting collateral damage is likely to be high. Think of Stuxnet, which may have targeted a specific Iranian nuclear-proliferation program but which also apparently "escaped" and spread. Thus it has inflicted damage on information systems in many other countries, and, now that it is out in the world, may be reengineered by others for their own, potentially unjust, uses.

In the realm of cyberwarfare on the battlefield, as opposed to its application as a form of strategic attack, a curious new ethical nuance emerges: acting early and aggressively might cripple an opponent in ways that sharply reduce physical casualties and overall

a Preemption refers to attacking when under imminent threat of attack, with Israeli actions at the outset of the 1967 Six Day War often referred to as a clear example. Prevention means striking before the enemy poses a serious threat, the rationale some policymakers used for the U.S.-led invasion of Iraq in 2003.

war costs. In essence, devoting more attention to disrupting enemy forces, even, or especially, with early surprise attacks, might be ethically acceptable due to the reduced destruction that would ensue.

The paradox is that the rise of cyberwarfare techniques in battle may make the traditionally unethical notion of starting a war attractive, yet at the same time cyberweaponry could improve the efficiency, and thus the morality, of the war-waging process itself. The logical terminus of a world replete with cyberwarfare would be an era with more uses of force, though they would be shorter and less destructive than traditional conflicts.

A key problem with this line of reasoning is that it does not deal adequately with escalation. A nation whose military is quickly debilitated on the battlefield due to cyber strikes may, instead of standing down or surrendering, respond with whatever weapons of mass destruction it has. Indeed, some nations might respond to their own perceived vulnerability to cyber disruption by seeking to acquire nuclear, biological, or chemical arms. The threat to use them might be subject to deterrent counterthreats by the war initiator; but those who have been attacked first generally hold the somewhat higher moral ground when it comes to retaliatory escalation. The best example of this is the decades-old NATO policy of reserving the right to use nuclear weapons in response to a conventional Russian invasion of Western Europe. A similar policy might well emerge in future efforts to cool the ardor of cyberwar enthusiasts, as in the new U.S. policy of threatening to respond to cyberattacks with conventional military means.[18]

Conclusion

The military encounter with computerization is playing out against a backdrop that includes many traditional concepts—strategic attack, battlefield close support, deterrence, arms control, and "just war" ethics—exposed now to troubling new wrinkles. But for all the complexities that have emerged, there are still reasonable paths forward. One could lead to more cyberwarfare on battlefields but few, if any, direct cyberattacks on societal infrastructures. This runs counter to the current emphasis among military leaders from nations around the world on the "strategic attack paradigm" but is a shift that might have profound practical (and ethical) benefits. It is also gaining traction among leading scholars, two of whom, Peter Sommer and Ian Brown, of the London School of Economics and the Oxford Internet Institute, respectively, took a clear position against the strategic-attack paradigm but acknowledged the importance of cyber operations on the battlefield: "Pure cyberwar... is highly unlikely. [But] in nearly all future wars... policymakers must expect the use of cyberweaponry...in conjunction with more conventional kinetic weaponry."[36]

However, carrying out this battle-oriented vision places huge demand on the cultivation of cyber-adept military service members of the highest caliber.

National governments and their military leaders might also have to choose between deterrence and arms control. During the decades of the Cold War and for some time after the dissolution of the Soviet Union, these concepts were viewed as moving hand-in-hand. But the information revolution and the rise of cyberwarfare may have gravely undermined deterrence, placing ever-greater weight on the need to emphasize behavior-based arms control, as by, say, entering into mutual agreements to refrain from being the first to mount cyberattacks against another country's civilian infrastructure. Against non-state networks, defense-oriented "denial deterrence" would likely prove of greater value than reliance on punitive threats.

Civilian and military decision makers thus have two preferred pathways: limiting cyberwar to battlefield use and embracing behavior-based arms control. If pursued, the paths could enable the information age to unfold in a more peaceful manner. Indeed, they offer the world community its best chance to use advanced information technology to spread prosperity and continue the intellectual improvement of all humanity. C

References

1. Adams, J. *The Next World War.* Simon & Schuster, New York, 1998.
2. Arquilla, J. et al. Cyberwar is coming! *Comparative Strategy 12*, 2 (Apr.–June 1993), 141–165.
3. Arquilla, J. et al. *Swarming & the Future of Conflict.* RAND Corp. Santa Monica, CA, 2000.
4. Barnaby, F. *The Automated Battlefield.* Free Press, New York, 1986
5. Baum, D. Battle lessons. *The New Yorker* (Jan. 2005).
6. Bellin, D. *Computers in Battle.* Harcourt, 1987.
7. Biddle, S. *Afghanistan & the Future of Warfare.* Strategic Studies Institute, Carlisle, PA, 2002.
8. Blank, S. Web War I. *Comparative Strategy 27*, 3 (Apr.–June 2008).
9. Briscoe, C.H. et al. *Weapon of Choice.* Combat Studies Institute, Fort Leavenworth, KS, 2004.
10. Broad, W. et al. Israel tests called crucial in Iran nuclear setback. *The New York Times* (Jan. 16, 2011).
11. Cebrowski, A. et al. Network-centric warfare. *Naval Proceedings* (Jan. 1998).
12. Clarke, R. *Cyber War.* HarperCollins, New York, 2010.
13. Clayton, M. Computer worm as guided missile? *The Christian Science Monitor* (Oct. 4, 2010).
14. de Landa, M. *War in the Age of Intelligent Machines.* MIT Press, Cambridge, MA, 1991.
15. Denning, D. *Information Warfare and Security.* Addison-Wesley, Boston, 1999.
16. Dixon, N. et al. *Company Command.* West Point, NY, 2005.
17. Dunnigan, J. *Digital Soldiers.* St. Martin's Press, New York, 1996.
18. Gorman, S. et al. Cyber combat: Act of war. *The Wall Street Journal* (May 31, 2011).
19. Gross, M. A declaration of cyberwar. *Vanity Fair* (Apr. 2011).
20. Hayden, M.V. The future of things 'cyber.' *Strategic Studies Quarterly 5*, 1 (Spring 2011).
21. Hoover, J.N. Air Force to tackle supply-chain security. *InformationWeek* (Aug. 20, 2010).
22. Laqueur, W. Postmodern terrorism. *Foreign Affairs 75*, 5 (Sept.–Oct. 1996).
23. Libicki, M.C. *Conquest in Cyberspace.* Cambridge University Press, New York, 2007.
24. Libicki, M.C. *Cyberdeterrence and Cyberwar.* RAND Corp., Santa Monica, CA, 2009.
25. Molander, R. et al. *Strategic Information Warfare.* RAND Corp., Santa Monica, CA, 1996.
26. Morgan, P. *Deterrence.* Sage Publications, Thousand Oaks, CA, 1977.
27. Owens, W. *Lifting the Fog of War.* Farrar, Straus and Giroux, New York, 2000.
28. Pape, R. *Bombing to Win.* Cornell University Press, Ithaca, NY, 1996.
29. Quester, G. *Offense and Defense in the International System.* Wiley, New York, 1977.
30. Robinson, B. Love/hate relationship with NMCI. *Federal Computer Week* (Sept. 10, 2007).
31. Rona, T. *Weapons Systems and Information War.* Boeing, Seattle, 1976.
32. Sanger, D. et al. Iran fights malware attacking computers. *The New York Times* (Sept. 26, 2010).
33. Schelling, T. *Arms and Influence.* Yale University Press, New Haven, 1966.
34. Singer, P.W. *Wired for War.* Penguin, New York, 2009.
35. Snyder, J. *The Ideology of the Offensive.* Cornell University Press, Ithaca, NY, 1989.
36. Sommer, P. and Brown, I. *Reducing Systemic Cybersecurity Risk.* Organization for Economic Cooperation and Development, Paris, France, 2011.
37. Stanton, D. *Horse Soldiers.* Scribner, New York, 2009.
38. Thomas, T. The threat of information operations: A Russian perspective. In *War in the Information Age*, R. Pfaltzgraff and R. Shultz, Eds. Brassey's, London, 1997.
39. Thornburgh, N. The invasion of the Chinese cyber spies. *Time* (Aug. 29, 2005).
40. Vistica, G. We're in the middle of a cyberwar. *Newsweek* (Sept. 20, 1999).
41. Walzer, M. *Just and Unjust Wars.* Basic Books, New York, 1977.

Acknowledgments

I would like to thank the reviewers and editors for their insightful critiques of earlier drafts of this article and for their thoughtful suggestions regarding revisions to it.

John Arquilla (jarquilla@nps.edu) is a professor of defense analysis at the U.S. Naval Postgraduate School, Monterey, CA; he was, 2005–2010, director of the Department of Defense Information Operations Center for Excellence, also located in Monterey, CA.

DOI:10.1145/2001269.2001288

How computer scientists can empower journalists, democracy's watchdogs, in the production of news in the public interest.

BY SARAH COHEN, JAMES T. HAMILTON, AND FRED TURNER

Computational Journalism

THANKS IN NO small part to the modern computer's ability to gather and disseminate seemingly limitless amounts and types of data, the institutions on which the public depends for information about government are melting away. To some this shift may look like a good deal: Why not trade a few newspapers for what appears to be infinite access to information? But as news staffs decline, so too does the public's ability to monitor power.

If there's a silver lining in this situation, it is the ability of computer scientists to strengthen the hands of the remaining professional reporters and engage new players in the watchdog process. Advances in analytic techniques, computing power, and the volume of digitally stored documents have prompted improvements in making sense of unstructured data. Much of the work to date has focused on the consumer arena: Web searches, blog discussions, tweets, and text messages that generate terabytes of information. Marketers, social scientists, information professionals, and governments have all invested heavily in innovative algorithms to analyze these sources.[12]

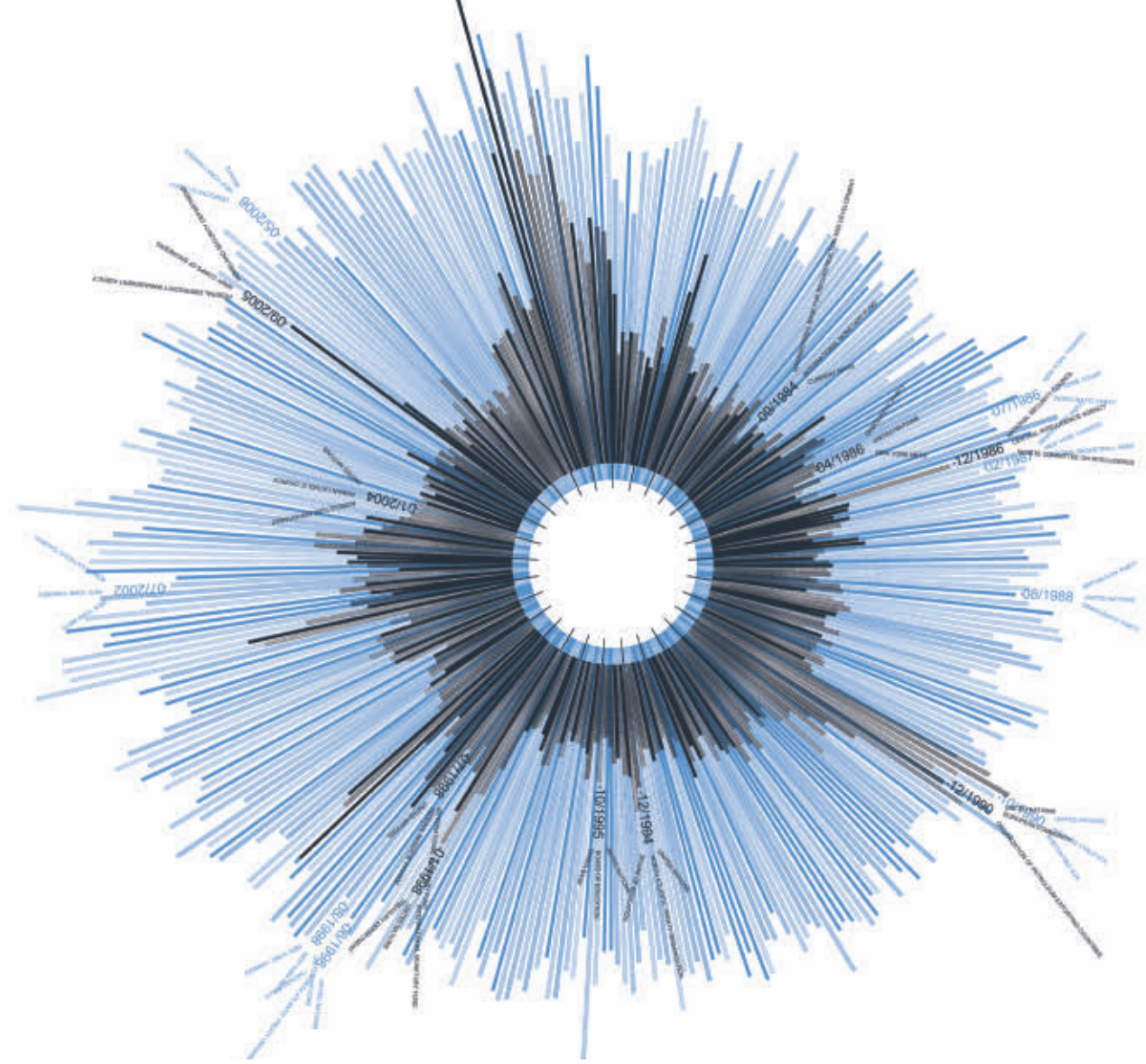
A similar push is also under way in investigative and public-affairs reporting. Researchers and journalists are exploring new methods, sources, and ways of linking communities to the information they need to govern themselves. A new field is emerging to promote the process: computational journalism. Broadly defined, it can involve changing how stories are discovered, presented, aggregated, monetized, and archived. Computation can advance journalism by drawing on innovations in topic detection, video analysis, personalization, aggregation, visualization, and sensemaking.[6–8,13]

Here, we focus on an aspect of computational journalism with a particularly powerful potential impact on the public good: tools to support accountability reporting. Building on the experience of an earlier generation of computer-assisted reporting, journalists and computer scientists are developing new ways to reduce the cost and difficulty of in-depth public-affairs reporting.

This aspect of computational journalism faces technical challenges ranging from how to transform paper-based documents into searchable repositories to how to transcribe collections of public video records. It also faces difficulty applying existing technology through user interfaces that accommodate the specific needs of journalists. Finally, it faces cultural challenges, as computer scientists trained in the ways of information meet journalists immersed in the production of news. If it is able to

» key insights

- **The public-interest journalism on which democracy depends is under enormous financial and technological pressure.**
- **Computer scientists help journalists cope with these pressures by developing new interfaces, indexing algorithms, and data-extraction techniques.**
- **For public-interest journalism to thrive, computer scientists and journalists must work together, with each learning elements of the other's trade.**



Visualization of the frequency of the words "hope" (blue) and "crisis" (graphite) published in the *New York Times*, 1981–2010.

overcome these hurdles, the field may sustain both public-interest reporting and government accountability.

IT and Watchdog Journalism

In the popular view, investigative reporting looks like the movie version of Watergate: secret meetings with sources, interviews, leaks, and cloak-and-dagger work. Due in large part to IT and statistical methods in the newsroom, it has long been more mundane and systematic.

A half-century ago, photocopying machines quietly revolutionized accountability journalism. The ability to copy documents worked in tandem with new freedom-of-information laws to make possible more sophisticated investigations. The machines let whistleblowers share agency records (such as correspondence and memoranda, inspection forms, and audits). Previously, investigations would often depend on undercover reporting, an ethically dicey practice. But the copy machine turned reporters' attention to documents and with them, to a new level of ethical clarity and accuracy.

In the late 1960s, a few reporters, led by Philip Meyer at Knight Newspapers, started using such research methods as sampling and correlation analysis to find and document stories.[14] Social-science tools have since helped establish notable patterns among variables, as in Bill Dedman's Pulitzer Prize-winning series in the *Atlanta Journal-Constitution* in 1988 on racial discrimination in mortgage lending in Atlanta.[5] Journalists have also used them to scout for outliers as news; for instance, reporters now make scatter plots to see the relationship between school test scores and student income, sometimes uncovering cheating by administrators and teachers in the odd, stray points.[9]

In the 1970s, reporters began to deploy the relatively novel methods of relational databases in their investigations. Using a portable nine-track tape reader and his newspaper's mainframe computer, Elliot Jaspin of the *Providence Journal* matched databases he acquired through government-in-sunshine laws. He found convicted drug dealers driving public school buses and local officials giving themselves discounts on their property-tax bills. By the late 1980s, he had shown that relational database technology and the Structured Query Language could be used to cross-reference systems to find news. He began traveling the U.S. showing reporters how to use these tools and founded an organization that later became the National Institute for Computer-Assisted Reporting (http://data.nicar.org/), an arm of the 4,500-member association Investigative Reporters and Editors (http://www.ire.org/).

VISUALIZAITON BY JER THORP

Today, matching data sets never intended to be matched is standard fare in newsrooms; stories highlighting child-care providers with felony records and voting rolls populated with the names of the dead are examples of the genre. The technique requires painstaking data cleansing and verification to deal with ambiguous identities and errors. Public records almost never include Social Security numbers, dates of birth, or other markers that would provide more accurate joins. But traditional news organizations have been willing to devote their time because they view documenting the failure of government regulation, unintended consequences of programs, and influence-peddling as core elements of their public-service mission.

However, such public-affairs reporting is increasingly at risk due to the decline in revenue and reporting staff in traditional news organizations—and is where the field of computational journalism can help the most. By developing techniques, methods, and user interfaces for exploring the new landscape of information, computer scientists can help discover, verify, and even publish new public-interest stories at lower cost. Some of this work requires developing brand-new technology, much of it involving work on new user interfaces for existing methods and some on simple repurposing. Technologies and algorithms already developed for informatics, medicine, law, security, and intelligence operations, the social and physical sciences, and the digital humanities all promise to be exceptionally useful in public-affairs and investigative reporting. At the same time, coupling the promised increased availability of government information with easy-to-use interfaces can aid nonprofessional citizen-journalists, non-governmental organizations, and public-interest groups in their own news gathering.

Understanding News Data

For computationalists and journalists to work together to create a new generation of reporting methods, each needs an understanding of how the other views "data." Like intelligence and law-enforcement analysts, reporters focus on administrative records and collections of far-flung original documents rather than anonymous or aggregated organized data sets. Structured databases of public records (such as campaign contributions, farm-subsidy payments, and housing inspections) generate leads and provide context, sometimes documenting wrongdoing or unintended consequences of government regulation or programs. But most news stories depend as much or more on collections of public and internal agency documents, audio and video recordings of government proceedings, handwritten forms, recorded interviews, and reporters' notes collected piece-by-piece from widely disparate sources. Some (such as press releases and published reports) are born digital; others are censored and scanned to images before being released to the public.

In many academic and commercial environments, researchers analyze comprehensive data sets of structured or unstructured records to tease out statistical trends and patterns that might lead to new policy recommendations or new marketing approaches. Journalists, however, look for the unusual handful of individual items that might point toward a news story or an emerging narrative thread. Journalists often collect records to address a specific question, which, when answered, marks the end of the analysis and the beginning of the story. This suggests a strict limit on the time and money invested in any document or data; it must be more effective or newsworthy than the alternative path of asking whistle-blowers or partisan insiders for the material.

On the flip side, investigative reporters have gigabytes of data on their hard drives and reams of documents in their file cabinets and are often willing to share them with researchers after a story is published. They are not bound by rules regarding human-subject testing or the research standards of peer-reviewed journals. Investigative reporters also expect plenty of false starts, tips that can't stand up to scrutiny, and stories that rarely hew to the route expected at the outset. In short, they write stories, not studies.

These characteristics suggest several areas of opportunity for collaboration among journalists, social scientists, humanists, and experts in computing. Over the past two years, we have conducted scores of interviews with reporters, editors, computer scientists, information experts, and other domain researchers to identify collaborations and projects that could help reduce the cost and difficulty of in-depth public-affairs reporting. In July 2009, we also brought together leaders in the field for a one-week workshop at Stanford University's Center for Advanced Study in the Behavioral Sciences (http://www.casbs.org).[10] Our conversations identified five areas of opportunity:

Combining information from varied digital sources. In our interviews, one reporter asked if it is possible to routinely combine all press releases from the 535 members of the U.S. Congress and their committees into a single searchable collection. Another wanted to search all 93 U.S. Attorney Web sites each week for news of indictments or settlements not likely to be shown in routine reviews of court records. A third dreamed of combing documents from the Web sites of the U.S. Securities and Exchange Commission, the Pentagon, and defense contractors to identify military officials who had moved into industry as consultants, board members, or executives.

What ties these ideas together is the ability to put into one repository material not easily recovered or searched through existing search engines. Each project features a limited number of discrete sources chosen by the reporter. Little of the material is available in RSS feeds. Each source has independent control of the form and format of its holdings; for instance, about one-quarter of the members of Congress keep press releases in databases accessed by Cold Fusion applications, while another significant portion post links to Adobe Acrobat files or organize HTML pages in subject-specific sections of their Web sites surrounded by member-specific templates. Each member's site is organized at least slightly differently.

This problem also arises for others, including public officials who want to monitor blogs, news sites, and neighborhood email lists, many not available in Google News alerts; corporations that want to monitor the public communications and regulatory filings of

their competitors and clients; and citizens who want to know everything their elected officials have done in the past week. Search and retrieval technology may be up this task, but the existing free and open source user interfaces to the technology remain crude and fail to address the variety of sources where finding answers to queries is less important than exploring what's new.

Information extraction. Most information collected by journalists arrives as unstructured text, but most of their work involves reporting on people and places. A beat reporter might cover one or more counties, a subject, an industry, or a group of agencies.

Most of the documents they obtain would benefit from entity extraction. Thomson Reuters allows the public to use its OpenCalais service (http://www.opencalais.com/), and at least a half-dozen open source and academic entity-extraction tools have been available for several years. The intelligence community and corporations depend on this basic but relatively new technique. But effective use of these tools requires computational knowledge beyond that of most reporters, documents already organized, recognized, and formatted, or an investment in commercial tools typically beyond the reach of news outlets in non-mission-critical functions.

Being able to analyze and visualize interactions among entities within and even outside a document collection—whether from online sources or boxes of scanned paper—would give stories more depth, reduce the cost of reporting, and expand the potential for new stories and new leads.

Document exploration and redundancy. There are two areas—finding what's new and mining accumulated documents—in which the ability to group documents in interesting ways would immediately reduce the time and effort of reporting.

Audiences, editors, and producers expect reporters to know what has been published on their beats in real time. Reporters need to notice information that is not commonly known but that could lead to news in interviews, documents, and other published sources. The recent explosion in blogs, aggregated news sites, and special-interest-group compilations of information makes distinguishing new stories time consuming and difficult. Collections of RSS feeds might comprise hundreds of stories with the same information.

Journalists look for the unusual handful of individual items that might point toward a news story or an emerging narrative thread.

In our interviews with journalists, we were told this challenge is more difficult than it seems for reporters lacking technical knowledge. But solving it would immediately reduce the amount of time spent distinguishing "commodity news," or news widely known and therefore uninteresting, from news their audience might not know or items that could prompt further reporting.

Another scenario arises in the collections of documents and data accumulated in a long investigative project. In some cases, existing search tools are not robust enough to find the patterns journalists might seek. For example, in 2006, reporters at the *New York Times* used more than 500 different queries to find earmarks for religious groups in federal legislation.[11]

In other cases, simply exploring a collection of documents might suggest further work if grouping them would help identify patterns. For example, in June 2010, the William J. Clinton Presidential Library released more then 75,000 pages of memoranda, email messages, and other documents related to Supreme Court nominee Elena Kagan. Grouping them in various ways might help better identify her interests, political leanings, and areas where she disagreed with others in the White House and suggest stories that could be missed simply by reading and searching the collection.

Combining these projects—content aggregation, entity extraction, and clustering of documents—could provide breakthrough innovation in investigative reporting. Together, they would directly address the key problem faced by most news consumers, as well as by producers: too much material too difficult to obtain containing too little information. These advances might allow for efficient, effective monitoring of powerful institutions and people and reduce the mind-numbing repetition and search in-depth reporting often requires.

Audio and video indexing. Public records increasingly consist of audio and video recordings, often presented as archived Webcasts, including government proceedings, testimony, hear-

ings, and civil- and criminal-court trials. Unless a third party has already transcribed, closed-captioned, or applied speech-recognition techniques on the record, most reporters have no way to search even a rough transcript. In addition, many reporters record many of their interviews digitally but rarely have useful speech-recognition software to index them. Basic consumer software products (such as Dragonspeech from Nuance) work on simple, short recordings or trained voices. Other promising projects (such as Google's Audio Indexing, or GAUDi) are not publicly available. GPS and voice recognition on mobile phones and voice mail could make reporters think solving their problem is simple.

Reporters could make near-daily use of technology and a user interface that would provide approximate indexing of a variety of voices and conditions, leading them to the portions they most want to review. They do not require the accuracy of, say, e-discovery by lawyers or official government records. Instead, they want a quick way to move to the portion of a recording that contains what may be of interest, then carefully review and transcribe it. Existing technology is probably adequate for reporters' immediate needs, but we are unable to find reasonably simple user interfaces to the technology that would allow unsophisticated users to test the technology on their own recordings.

Extracting data from forms and reports. Much of the information collected by reporters arrives in two genres: original forms submitted to or created by government agencies, often handwritten, and reports generated from larger systems, sometimes electronically and sometimes on paper. Examples include financial disclosure statements of elected officials, death certificates, safety inspections, sign-in sheets at government checkpoints and police incident reports. Journalists have few choices today: retype key documents into a database; attempt to search recognized images; or simply read them and take notes. An in-house programmer can occasionally find the pattern of digital reports intended for printing that can be leveraged to reverse them back into a structured database, but this time-consuming job requires skill well beyond nearly all reporters.

Extracting meaningful information from forms is among the most expensive and time-consuming large news investigations. Its cost sometimes results in abandoning promising stories. Reducing that cost could encourage substantially more important reporting on government, particularly at state and local levels where special-interest groups and NGOs are less likely to step in to help.

New Tools, New Organizations

A handful of new services have emerged to help address journalism's data challenges. Usually free for small-scale or non-commercial use, they facilitate analysis, visualization, and presentation of structured data: Google Refine promises to let reporters scrap their spreadsheets for filtering, viewing, and cleaning basic data sets; ManyEyes from IBM lets news organizations visualize and share data on their Web sites; Tableau Public from Tableau Software, Google Earth, and other such products are routinely used by news organizations to generate and publish visualizations. New tools (such as TimeFlow developed at Duke University as an investigative tool for temporal analysis) are being created to address some longstanding needs of reporters.[4]

Another set of tools created for other purposes, often experimental or academic, shows promise for the fast-paced, ad hoc nature of reporting challenges. Several political-science scholars have created tools for clustering legislation and other public documents; homeland-security developers have created tools (such as Georgia Tech's Jigsaw[15]) for visualizing the connections among documents; and the CMU Sphinx project[3] has created reasonably accurate open source speech-recognition technology. Applications developed for intelligence, law enforcement, and fraud investigations by such companies as Palantir Technologies and I2 are expensive and finely tuned to specific industries, though they address similar challenges on a different scale and with different requirements for speed and accuracy.

DocumentCloud (http://www.documentcloud.org), a nonprofit founded in 2009 by journalists at the *New York Times* and ProPublica, hopes to address one of the most vexing issues in documents reporting: scanned images files. With it, reporters can annotate their documents as they find interesting or questionable sections and see which entities appear in multiple documents. At this writing, most news organizations have used it most effectively to publish government documents. But the project, which includes information extraction as a standard feature, shows great promise helping address some of the problems of digesting large document collections.

If such new methods and tools could be more widely adopted in journalism, they could perhaps do for investigative reporting what the photocopier and relational database did in decades past.

Despite these possibilities, challenges persist in working with unstructured data for watchdog reporting. News organizations increasingly look toward their audiences to fill some of the gaps. Thanks to methods of collaboration pioneered in computer science, amateurs and professionals now find themselves reporting side-by-side.

In 2005, Josh Marshall of the online news site Talking Points Memo (http://talkingpointsmemo.com/) encouraged his readers to contribute local stories of politicization of the Justice Department under the George W. Bush administration. His work, and the work of his audience, won a prestigious George Polk award for investigative reporting in the first known use of crowdsourcing to uncover an important investigative story. The *Guardian* in London has enlisted its readers to help review payment records of members of Parliament on deadline.[1] American Public Media (http://americanpublicmedia.publicradio.org/) created possibly the largest crowdsourcing network, with more than 60,000 members in its Public Insight Network (http://www.publicinsightnetwork.org/). It now needs to find ways to better understand and mobilize them while encouraging local National Public Radio affiliates and other partners to use the network effectively. Leaders in social media and collaboration, including ProPublica, are helping reporters learn to motivate their audiences and organize ad hoc communities around projects. These models all

show promise enlisting more eyes and ears in accountability reporting.

Next Steps

For many aspects of accountability reporting, including beat and investigative work, a key question for the future is whether journalists' research problems are scientifically interesting, challenging, or even new enough. Some could be solved with new user interfaces that accommodate journalism's quirks. Others are bothersome impediments to more interesting work. For example, possibly the most intractable problem in investigative reporting remains the form in which the material arrives, often on paper, with large sections blacked out by censors, or in large files combining images of thousands of individual records (such as email messages, memos, forms, and handwritten notes). The investment required to address the problem is unlikely to be made in the news industry and may not interest software developers or scientists. Philanthropists, academic institutions, and spin-offs from government-funded research may ultimately provide the solution, not computer scientists.

Journalism and computer science schools have begun to address the questions at the intersections of their fields. Two top journalism programs, the Columbia University Graduate School of Journalism and Northwestern University's Medill School of Journalism, have initiated interdisciplinary programs with computer science or engineering departments. At the Georgia Institute of Technology, professor Irfan Essa teaches an influential course in computation and journalism, conducting research in the field, while videogame scholar Ian Bogost explores new ways to use games in journalism.[2,7]

Fortunately, funders have also stepped into the financial vacuum surrounding watchdog journalism. The largest is the Knight Foundation, which funds the annual $5 million Knight News Challenge contest and other grants and programs, along with university centers, startups, and nonprofit investigative news sites. Knight has funded digital innovators EveryBlock, DocumentCloud, and others. Projects funded through government programs (such as scanning and digitization projects at the National Archives and the Library of Congress) might further help address the challenges.

The Association for Computing Machinery's special interest groups, most notably Information Retrieval (http://www.sigir.org/) and Knowledge Discovery in Data (http://www.kdd.org/), could foster sessions at their own meetings and with journalism organizations, including Investigative Reporters and Editors. Annual research competitions (such as the Text Retrieval Competition, the IEEE Visual Analytics Symposium's Challenge, and the ACM's Data Mining and Knowledge Discovery competition) could each include as research topics the kind of data challenges facing journalists today.

Conclusion

How might the worlds of politics, governance, and social discourse change when computational journalism fulfills its promise? Not surprisingly, part of the answer is journalistic: Stories will emerge from stacks of financial disclosure forms, court records, legislative hearings, officials' calendars or meeting notes, and regulators' email messages that no one today has time or money to mine. With a suite of reporting tools, a journalist will be able to scan, transcribe, analyze, and visualize the patterns in these documents. Adaptation of algorithms and technology, rolled into free and open source tools, will level the playing field between powerful interests and the public by helping uncover leads and evidence that can trigger investigations by reporters. These same tools can also be used by public-interest groups and concerned citizens.

Much more of the answer, though, involves democracy itself. How can citizens govern themselves if they are unable to hold their governments accountable? This ancient question is often phrased as "Who guards the guardians?" A hundred years ago, or even 20, the answer might have been "full-time journalists." But today they can be only part of the answer. Journalists need to partner with computer scientists, application developers, and hardware engineers. For decades, the computing community has empowered individuals to seek information, improving their lives in the process. Few fields have done more to give citizens the tools they need to govern themselves. Few fields today need computer scientists more than public-interest journalism.

Acknowledgments

We would like to thank James Bettinger, Glenn Frankel, Ann Grimes, Jeffrey Heer, Michael Schudson, John Stasko, Fernanda Viegas, and Martin Wattenberg for generously reviewing and improving this article. C

References

1. Anderson, M. Four crowdsourcing lessons from the Guardian's (spectacular) expenses-scandal experiment. *Nieman Journalism Lab* (June 23, 2009); http://www.niemanlab.org/2009/06/four-crowdsourcing-lessons-from-the-guardians-spectacular-expenses-scandal-experiment/
2. Bogost, I., Ferrari, S., and Schweizer, B. *Newsgames: Journalism at Play.* MIT Press, Cambridge, MA, 2011.
3. Carnegie Mellon University Sphinx; http://cmusphinx.sourceforge.net/
4. Cohen, S., Viegas, F., and Wattenberg, M. TimeFlow Github repository http://wiki.github.com/FlowingMedia/TimeFlow/
5. Dedman, B. *The Color of Money: Text of the Pulitzer-Winning Articles*; http://powerreporting.com/color/
6. Diakopoulas, N. Research and projects; http://www.nickdiakopoulos.com/research-and-projects/
7. Essa, I. Computation and Journalism class at Georgia Tech; http://compjournalism.wordpress.com/
8. GVU Center at Georgia Tech. *Journalism 3G: The Future of Technology in the Field. A Symposium on Computation and Journalism* (Atlanta, Feb. 22–23, 2008); http://www.computational-journalism.com/symposium/index.php
9. Hacker, H. and Benton, J. Faking the grade. *Dallas Morning News* (June 3–5, 2007).
10. Hamilton, J.T. and Turner, F. *Accountability Through Algorithm: Developing the Field of Computational Journalism. Report from Developing the Field of Computational Journalism.* Center for Advanced Study in the Behavioral Sciences Summer Workshop (Stanford, CA, July 27–31, 2009); http://dewitt.sanford.duke.edu/images/uploads/About_3_Research_B_cj_1_finalreport.pdf; and workshop bibliography: http://dewitt.sanford.duke.edu/images/uploads/About_3_Research_B_cj_2_ReadingLinks.pdf
11. Henriques, D.B. and Lehren, A.W. In God's name. *The New York Times* continuing series (Oct. 2006–Nov. 2007); http://www.nytimes.com/ref/business/churchstate.html
12. Hey, T., Tansley, S., and Tolle, K., Eds. *The Fourth Paradigm: Data-intensive scientific discovery.* Microsoft Research, Redmond, WA, 2009; http://research.microsoft.com/en-us/collaboration/fourthparadigm/
13. Mecklin, J. Deep throat meets data mining. *Miller-McCune* (Jan./Feb. 2009); http://www.miller-mccune.com/politics/deep-throat-meets-data-mining-4015/
14. Meyer, P. *Precision Journalism: A Reporter's Introduction to Social Science Methods.* Rowman & Littlefield Publishers, Inc., New York, 2002.
15. Stasko, J., Görg, C., and Liu, Z. Jigsaw: Supporting investigative analysis through interactive visualization. *Information Visualization* 7, 2 (Summer 2008), 118–132; http://www.cc.gatech.edu/gvu/ii/jigsaw/

Sarah Cohen (sarah.cohen@duke.edu) is Knight Professor of the Practice of Journalism and Public Policy at the Sanford School of Public Policy, Duke University, Durham, NC.

James T. Hamilton (jayth@duke.edu) is Director of the DeWitt Wallace Center for Media and Democracy and Charles Sydnor Professor of Public Policy at the Sanford School of Public Policy, Duke University, Durham, NC.

Fred Turner (fturner@stanford.edu) is Associate Professor of Communication and Director of the Program in Science, Technology and Society, Stanford University, Stanford, CA.

DOI:10.1145/2001269.2001289

Exploring the connection of biology with reactive systems to better understand living systems.

BY JASMIN FISHER, DAVID HAREL, AND THOMAS A. HENZINGER

Biology as Reactivity

BIOLOGY IS NOT an exact science. Biological systems are messy and noisy, and our understanding of many biological scenarios remains extremely vague and incomplete. We cannot assign precise rules to the way cells behave and interact with one another, and we often cannot quantify the exact amounts of molecules, such as genes and proteins, in the resolution of a single cell. To make matters worse (so to speak), the combinatorial complexity observed in biological networks (for example, metabolic and signaling pathways) is staggering, which renders the comprehension and analysis of such systems a major challenge.

One way to explain a certain class of complex dynamical systems is to view them as highly concurrent *reactive systems*.[42] We argue that this perspective is a natural fit for many biological systems.[40] A reactive system is characterized by the way it responds to its inputs, as they arrive over time, sequentially, or concurrently. The system's behavior and outputs are not just a function of the input values but also of the order in which the inputs arrive, their arrival times, speeds, and locations, and so forth.

A living cell, we claim, is not only reactive in nature, but is the ultimate example of a reactive system, and so are collections thereof. As explained in Cohen and Harel 2007,[16] a cell succeeds by being robust and resilient. It reacts to inputs and perturbations and continues to survive thanks to its reactive dynamics. The cell is a reactive system that expresses a dynamic narrative, in which the DNA code is one of many formative inputs. Structural proteins, enzymes, carbohydrates, lipids, hormones, and other molecules also play key roles in forming and informing the system.

Biological systems are also highly adaptive, to both internal and external changes; they use signals coming in from receptors and sensors, as well as emergent properties to fine-tune their functioning. This adaptivity is another facet of the reactivity of such systems.

We are well aware of the fact that there are many aspects of biology that are not reactive, or that reactivity is not the best way to view them. These include, for example, structural aspects of chemicals. While we point to the importance of combining other modeling methods with reactive models, the main thrust of this article is to explain the connection of biology with reactive systems, and the benefits that can be gained from adopting such a view.

Viewing biology as reactivity is not just an illustrative analogy, but has

» key insights

- **Living cells, and collections thereof, can be viewed as reactive systems.**
- **Reactive models emphasize important aspects of biological systems, such as executability, concurrency and interaction, multiple scales, and combinatorial complexity.**
- **Concepts, languages, and tools for the description and analysis of reactive systems can help in the process of biological discovery, ultimately by providing biologists with virtual experimentation environments.**
- **Biological experimentation needs to obtain quantitative data across different levels, and reactive modeling needs to focus on incorporating and linking such data.**

far-reaching consequences. Complex reactive behavior is difficult to comprehend because it is neither predominantly computation-rich nor primarily data-intensive, and does not yield well to techniques based on algorithmics, mathematics, and data management. Rather, reactive systems "live" (if we may use a pun in an article on biology) in order to react.

Within computer science, several approaches have been offered for dealing with reactivity. One of the richest and most fruitful is that of *finite automata*, or *finite state machines*, which are used widely in many stages of the design of hardware and software systems. It suffices to observe the central role of state machines in standards for the engineering of embedded and real-time software and systems, such as the UML.

The original theory of finite automata, which was conceived of in order to model hardware circuits, has brought in its wake many extensions and variants, such as automata on infinite words and trees; communicating, hierarchical, and timed state machines; Statecharts, and many others.[1,75] In addition, and related to these, special logics, algebras, and calculi have been devised for reactivity, first and foremost among which are the varieties of *temporal logic*[63] and *process algebra*.[57] Scenario-based languages have been used to model biology too, such as *live sequence charts* (LSC).[19] All of this, in turn, has resulted not only in numerous languages and tools for specifying and programming reactive behavior, but has also given rise to powerful means for analysis, such as model checking.[15]

However, when reviewing the great efforts made in understanding and building complex reactive systems, it is not only the final results—languages and tools—that stand out. An even more important contribution can be found in the fundamental insights, concepts, and ways of thinking that have resulted from this research. Hand-in-hand with the central notion of reactivity go the discrete event-based *execution and simulation* of dynamical systems, which

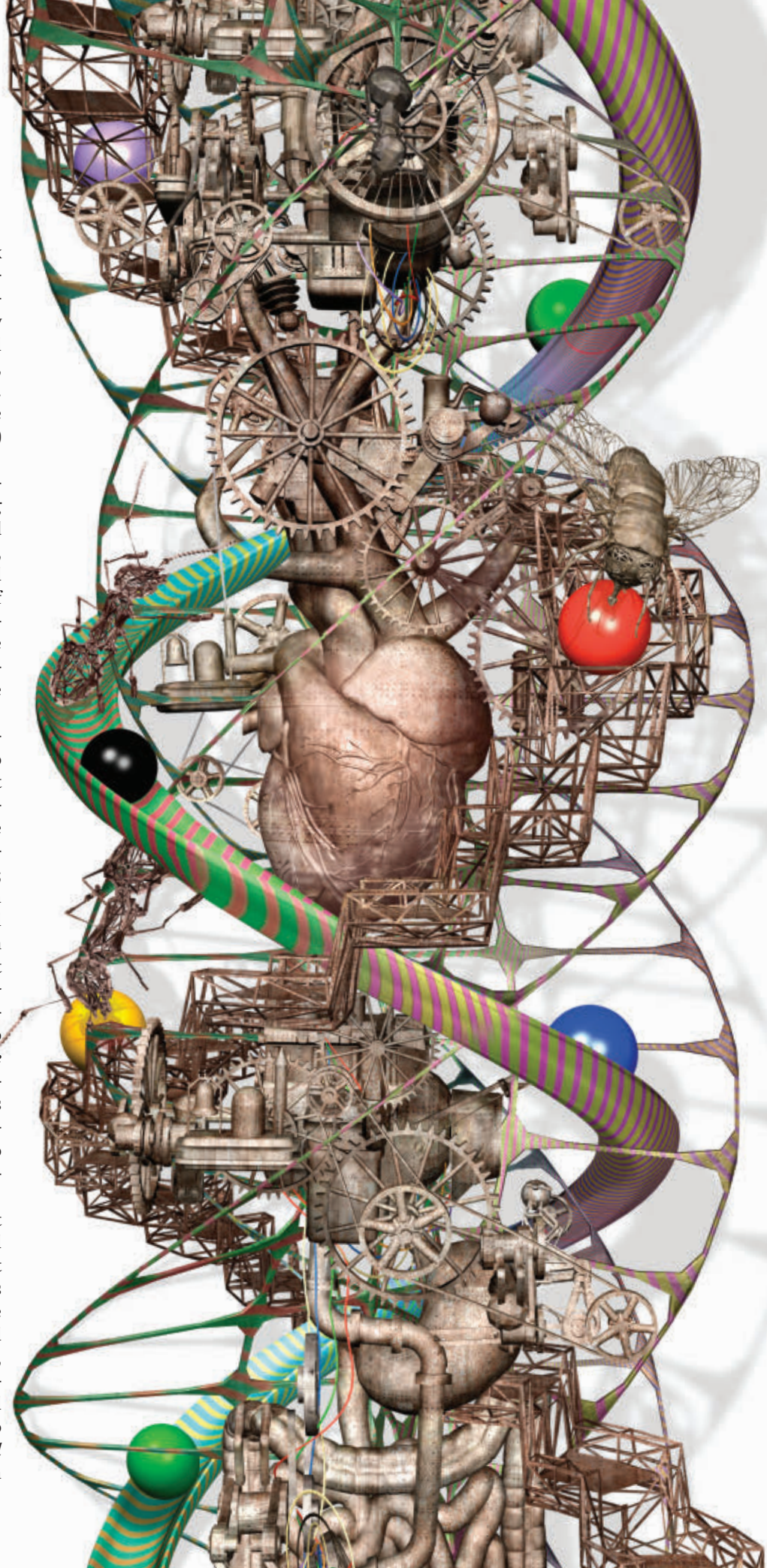


ILLUSTRATION BY J.F. PODEVIN

requires a fundamental understanding of *parallelism*, *interaction*, and *causality*; the design of complex systems from building blocks, requiring means for *composition* and *encapsulation*; and the description of systems at different levels of granularity, requiring methods for *abstraction* and *refinement*.

Of course, many of these concepts are not exclusive to reactive systems, but they are critical for understanding them, and they are among the main ideas that render possible the reliable development of truly complex reactivity. The claim made in this article is that these ideas can be of great benefit in the modeling and analysis of biology.

Reactivity and Biology

The process of modeling a piece of biology is very different from that of modeling a human-made system. The motivation is different, the goals are very different, the people involved are also different, the scales are different, and so on. Still the underlying maxim of this article is that a fighter jet and a fruit fly, for example, have many things in common, and that there is much to be gained from using ideas from engineering the former to reverse-engineering the latter, even though the fly is in a way far more complex than the airplane.

Biological systems are ultimately molecular and cellular machines. Using a fixed set of building blocks (molecules) with some fixed functionalities (for example, cleavage, connection, ionization, phosphorylation), cells emerge, and these multilevel machines then utilize various combination techniques to give rise to the life we see around us. The "execution" of a biological system is distributed to a level that is still beyond comprehension. In a sense, a biological system can be viewed, almost on a philosophical level, as a collection of cells (each containing a multitude of molecules) that work in concert and independently to achieve a mutual goal without a central controller that coordinates them all. The cells in such a collection communicate and pass information to each other in order to help achieve their mutual goal. In doing this, they can also affect limited control on their environment.

Let us now concentrate on the programs (to put it simply, the DNA molecules) that drive these enormously distributed processes. In the case of the fruit fly *Drosophila Melanogaster*, for example, a 165 million base-pair program defines approximately 13K functions (proteins). Such programs deal with two entangled and inseparable tasks: how to evolve the machine (cell collection) from a single cell, and how to run the machine so that it eventually creates other such machines. At all stages, both the development and maintenance are determined by which genes are expressed and which are not. Effectively, in each cell, the expressed genes constitute a *local control state* of the program, which tells the cell which functions it should execute and which are irrelevant to its (current) role. The state is local because it may be different in different cells. At all points, communication with other cells and reaction to the environment are the keys to determine what happens next.

More precisely, the expressed genes tell the cell and its machinery which proteins to produce and how many of them, and which proteins not to produce. Gene expression and the corresponding protein production are the basic mechanisms that govern all actions of a cell. Proteins serve various purposes in the cell itself and also act as mediators of communication between cells. The protein concentration levels within a cell can be considered to be the *local data state* of the machine. As in computer programming, the data state helps prescribe the control flow, in that the proteins determine which genes are expressed and which are not.

Consider the first part of the program, the one that tells the system how to evolve. All multicellular organisms evolve from a single cell. The main engine driving this part is called *differentiation*, which is the process by which a cell that divides can give rise to what will ultimately become very different types of cells. There are mainly two mechanisms for cell differentiation. One, which is mostly active in the early stages of development of the machine, is by *asymmetric segregation*—asymmetric accumulation of substances inside a single cell that divides into two different kinds of cells. The second, which becomes active after a few initial divisions have taken place, is governed by communication between cells, by *morphogen gradient* or *inductive signaling*. The former breaks the symmetry between a group of equivalent cells, by one cell, external to the group, secreting a gradient of molecules (for example, according to distance from the secreting cell). The latter involves direct communication between cells, leading them to choose different fates through the passage of information. All communication is carried out by transferring and manipulating molecules. Such a process may start, for example, by one cell sending a molecule to a neighboring cell, which will then trigger the creation of another molecule in that neighboring cell (function activation).

The differentiation phase of cells leads to a global system state where different cells have specialized to perform certain roles that are necessary for the mutual fulfillment of their common goal (reproduction). The second part of the program guides the system's behavior after having evolved from an embryo, although major pieces are already in use throughout development because the cells are alive and need to maintain themselves during the process. Initially, this part of the program includes little coordination and communication between cells. However, as the cells specialize, they start to communicate and to manipulate their environment in order to perform basic actions such as transferring nutrients from one place to the other, transferring oxygen, and so on.

Perhaps the most magical part of this program controls movement (actuation). This process is slightly different, as it involves a new form of communication through electrical signals that are transmitted via nerve cells, and which are then translated into local molecular mechanisms. But notice that when a muscle contracts, the nerve reaches only some of the muscle cells, and responsibility for the remainder of the signal's effect must be taken over by other mechanisms. In the reverse direction, sensors such as eyes, ears, and taste buds pass information from chemical to electrical signals. Moreover, as mentioned earlier, the ability of biological systems to adapt, as a result of incoming signals, is also a highly relevant feature of their reactivity.

These are the kinds of processes that we aim to understand better by creating executable reactive models. The ul-

timate challenge is to understand how the behavior of an organism, or a part thereof, emerges from the individual behaviors of the cells in the collection. We believe this is best done by focusing on the reactivity of cells and the communication between them.

Concurrency and interaction. The essence of a reactive system is that different parts of the system are simultaneously active ("alive") and interact with each other. In the case of computerized systems, these can be software processes, hardware components, intelligent agents, or the environment, and in the case of biology they can be molecules and molecular processes, cells, tissues, bacteria, or again, the environment.

The operation that combines several simultaneously active parts of a system is called *parallel composition* because the individual activities happen concurrently, in "parallel," rather than sequentially (one after the other). However, in computer science there are several different interpretations of what "in parallel" actually means. At one extreme is the *clock synchronous* interpretation, where execution proceeds in lockstep with a global clock, as a sequence of discrete steps, with each building block (component) of a system contributing one action to each time unit. This happens, for example, in a clocked hardware circuit built from individual logic gates without combinational loops, where each gate performs one operation per clock tick. At the other extreme is the *interleaved asynchronous* interpretation, where each action represents the contribution of a single component, but different actions, possibly carried out at different time rates, may represent different components. This happens, for example, in distributed software systems where the individual processes proceed at independent speeds. Implicit to the asynchronous interpretation is a scheduler, which chooses for each step the component that will contribute to that step. The scheduler is usually assumed to be "fair," the simplest interpretation of which is that it cannot neglect to choose any component forever.

Both kinds of concurrency occur in biology, for example, in molecular processes "running" in parallel, and techniques for capturing both have found their way into biological models. Still, sticking religiously to these two interpretations may be inadequate for representing the parallelism that arises in a large collection of individual cells. Cells, for example, often proceed "roughly" in lockstep, but not completely so: some reactions may be a bit faster here but a bit slower there, but biological reactions do not proceed at completely independent rates either. An example of how to adapt useful computer science techniques to biology can be found in the recent compromise between these two called *bounded asynchrony*.[33] It allows the components of a system (for example, cells) to proceed independently while bounding the differences in their rates. Bounded asynchrony has been used to accurately model some of the experimental observations made about certain cell-cell interactions: it captures the variability observed in cells that, although equally potent, assume distinct fates, and thus can be used to provide mechanistic explanations for some phenomena that are observed during cell-fate determination.

The process of modeling a piece of biology is very different from that of modeling a human-made system. The motivation is different, the goals are very different, the people involved are also different, and the scales are different.

Bounded asynchrony is but one example of how the modeling of biological systems may give rise to new variations of reactive concepts. Once it is agreed upon what the most suitable interpretation of "parallel progress" means for a collection of cells, the communication between these cells needs to be modeled. Again, there are several standard alternatives, designed for modeling hardware and software, which may not be readily adequate for our purposes. Traditionally, the interaction between concurrent processes can happen through sharing state or through sending messages. This distinction has led, for example, to thread-based programming languages on one hand, and actor-based languages on the other. The transfer of molecules between cells seems more akin to message passing, but within that paradigm itself there exists a wide spectrum of design choices, ranging from rendezvous, which stops the execution of two processes until simultaneously the sending process is ready to send and the receiving process is ready to receive a message, to unbounded message buffers, which retain sent messages until they are received. Explicitly spatial models[11,69] are natural candidates for capturing morphogen gradients.

While a biological scenario may suggest a particular execution and interaction model, often it is a matter of style and the availability of analysis tools that determines the choice of model. If there is any lesson to be learned from several decades of research on concurrency theory in computer science, it is that we should not preach the use of any particular modeling languages or methods, but the very notion of reactive modeling itself, and the freedom of choice it offers through a wide variety of different execution and interaction mechanisms.

Towers of abstraction. Every model is necessarily a simplification of a physical situation. The modeler is interested in some, but not all, aspects of the physical system. The key to modeling a complex system is to design a simplification, such that (1) the behavior of the model is correlated to behavior of the system and (2) the model is amenable to mathematical or computational analysis. Point (1) measures models according to their *precision*; point (2), according to their *performance*. Precision must be sacrificed in order to gain performance, but this must be done in a quantifiable way.

Computer science offers a very rich and useful theory, called *abstraction*, for trading off precision against performance in a principled manner.[17] The theory of abstraction is used in computer science very beneficially in work on verification and testing, and is aimed at relating models of different precision. It thus differs greatly from theories of approximation, which measure the precision of a model in terms of an error bound on how much a model may deviate from the system. A theory of abstraction, by contrast, measures the precision of a model in terms of which properties of the system are preserved in the model—hence the relevance to verification.

Consider, for example, the property that event *A* is never followed by event *B*. A model may preserve such a property, despite possibly introducing a very large error. On the other hand, even a model that introduces only a very small error may violate the property when the actual system does not. The preservation of properties can ensure that it suffices to check a property on the model (simple) in order to conclude that it holds for the real system (complex). This principle lies at the heart of all hardware and software design. Modern computer systems could not be built without several layers of abstraction—the gate-level abstraction of the underlying physics (transistors); the register level; the machine instruction level; the programming language; among others.

Verification can become very expensive, or impossible, for human-made systems, and biological systems are often even more complex. The good news is program and system verification has made enormous strides in recent years.

In analogy, a biological system may be viewed at many different levels, ranging from the molecular level to the cell level to the level of organs and entire organisms. The power of an abstraction layer derives from the fact that all lower-level details may be omitted from consideration. For example, when designing a logical circuit from Boolean gates, the designer does not need to know about voltages and capacities. Different layers may exhibit different scales in space and time, even switching between continuous and discrete scales (for example, continuous voltages representing Boolean values). By contrast, in biology, we have not (yet) been able to identify building blocks from which we can explain metabolic pathways and cell behavior without referring to the underlying biochemical (molecular) mechanisms.

In multiscale reactive systems, an additional characteristic phenomenon is the *emergence* of new high-level properties. An emergent property is a behavior of the system that is not easily expressed at a lower scale. Life, for example, is an emergent property; none of the component molecules of a cell are alive, only a whole cell lives. The cell as a whole emerges only when we zoom out, so to speak, reaching the scale at which it functions as an object with its own interactions with other cells and molecules. Thus, interactions at one scale create new objects at a higher scale, which is the essence of emergence. Quoting from Cohen and Harel,[16] "A major goal of systems biology is to learn how the concurrent reactions and interactions of the lower-scale components of a cell, organism, or society generate emergent properties visible at higher scales and higher layers of reality."

All this leads to the need to be able to observe and manipulate biological systems on multiple scales. Abstraction can work wonders here, and if carried out multiple times we get a *tower of abstractions*. A unique feature of software that helps in handling multiple scales

is *inheritance*, where an existing behavior is taken and augmented with a few modifications. This encapsulates the previous behavior and overrides or extends parts of it. One of the major challenges in biological modeling is to find similar means to encapsulate biological and biochemical complexity that will allow us to use abstraction beneficially to bridge and relate different scales in order to manage the immense complexity observed in living systems. Once we have such multiscale models we will need to search for the right computational frameworks that will allow us to zoom back and forth between lower-scale data and higher-scale behavior, while experimenting *in-silico*. This, we feel, is an ideal way to study emergence computationally. A modest attempt in this direction can be found in the *Biocharts* approach of Kugler et al.[52]

Noise and choice. Stochasticity has received much attention in systems biology,[4,55,56] as numerous experimental studies have reported the presence of probabilistic mechanisms in cellular processes.[26,29,61] The investigation of stochastic properties of biological systems requires that computational models take into consideration the inherent randomness of biochemical reactions. Stochastic kinetic approaches give rise to dynamics that differ significantly from those predicted by deterministic models because a system may follow very different scenarios with non-zero but varying likelihoods.

The dogma for this kind of modeling assumes that a molecular mixture is well stirred and has fixed volume and temperature (though PDEs can be used to model variations in these too). The state of a network of biochemical reactions at any point in time is then given by the population vector of the involved chemical species (such as, molecules). The temporal evolution of the system can be described by a continuous-time Markov process,[37] which is usually represented as a system of ordinary differential equations (ODEs) called the *chemical master equation* (CME). While individual system parameters, such as the mean of the state distribution changing in time, can be studied using deterministic differential equations, this is inadequate for uncovering branching, switching, or oscillatory behavior, such as cell fate determination (the mean between two alternative cell fates is hardly meaningful). Such phenomena require a fully stochastic analysis.

However, building a stochastic model that would mimic sufficiently accurately the stochastic behavior of the actual biological system is extremely difficult when sufficiently accurate rates of change are not known, as is usually the case. In addition, we have no satisfactory theory for abstracting stochastic models. This becomes a central issue when we wish to analyze the higher-level tiers of a biological model—say, those at the intercellular level—by hiding the underlying molecular reactions. In such situations, a nondeterministic modeling of the possible behavioral alternatives of a system may be justified. For example, for determining possible cell fates, it has proved fruitful to quantize concentration levels of molecules into a few discrete ranges (for example, low, medium, high) with nondeterministic transitions between the possible ranges.[32,34] A nondeterministic model can only provide potential outcomes, without the corresponding probabilities, but it does provide hypotheses that can be confirmed or refuted experimentally.

In computer science, many formalisms have been designed—or existing ones have been extended—to support nondeterministic transitions for modeling alternative choices; for example, Petri nets, various kinds of interacting state machines, live sequence charts (for example, Peterson,[62] Harel,[38] Harel and Gery,[41] Damm and Harel[19]). Many of these formalisms have been used to model biology as well.[6,23,25,32,49,73] The question of how such discrete, nondeterministic models relate to the underlying continuous, stochastic mechanisms, and which properties they preserve, remains an interesting topic of investigation. Hybrid (mixed discrete-continuous) abstractions can also play a central role to bridge the gap.[35,45,74] To find an optimal trade-off between precision and performance, it may be best to treat some system parameters as variables that change continuously in time, while others can be safely represented as Boolean switches.

One main advantage of nondeterministic over stochastic models, and of discrete over continuous models, is the former more efficiently support a broad class of techniques, generally subsumed under the title of *verification*, which we discuss here.

From simulation to verification. Computer science offers a rich spectrum of means for assessing the dynamic properties of reactive models, ranging from simulation to verification. While simulation generates one behavior of a model at a time, verification looks systematically at the set of all possible behaviors.

Simulation has a long tradition in computational science, based mostly on numerical methods for solving equational models of a system. For example, from the CME for a system of (bio) chemical reactions, stochastic simulation can be used to generate trajectories of the underlying Markov process.[36] Simulation methods are in widespread use because they are easy to implement, and each simulation run can be viewed as a single "in silico'' experiment, thus fitting well into the methodology of experimental science. However, if we are interested in properties of the set of all runs, such as estimates for probability distributions on the system state at a given point in time, then the number of trajectories required for statistical accuracy is very large.[21] This is because in order to halve the confidence interval of an estimate, four times more trajectories have to be generated. Consequently, even when computationally feasible, stochastic simulation may often result in a very low level of confidence in the accuracy of the results.

More information can be obtained, for example, by a reachability analysis of the model, which explores the state space from an initial state or state distribution in a breadth-first, rather than depth-first, manner. The distinction between simulation and reachability analysis is akin to the distinction between program testing and program verification. A reachability analysis can provide insights into biochemical models,[22] but many techniques that have been developed for coping with large and unbounded state spaces in the reachability analysis of nondeterministic models—such as model abstraction, model decomposition, symbolic data structures, symmetry, and partial-order reduction—have yet to be adapted satisfactorily to stochastic models.

It goes without saying that one must

not forget the main difference between simulation and verification: the worst-case computational intractability of the latter. In general, as is so well-known, verification can become very expensive, or impossible, for human-made systems, and biological systems are often even more complex. The good news is that program and system verification has made enormous strides in recent years: the worst-case performance can sometimes be alleviated by clever and powerful means, though obviously this is outside the scope of the present article. It is these advances that we hope to be able to utilize in modeling and verifying biology, too.

As discussed earlier, higher precision in the model generally means lower performance in the analysis. The modeler therefore aims at the lowest possible precision that preserves the property of interest. Reachability analyses offer the possibility of dynamically changing the level of abstraction during the analysis.[14] In this way, the precision of a model can be refined on demand, precisely in those areas of the state space where more detail is required for determining the truth or falsehood of a given property. Reachability analysis can also be equipped with rich languages for defining and checking temporal properties of systems, such as temporal logics—a research direction known as *model checking*; for example, see Calzone et al.,[10] which describes the BIOCHEM tool used in the arena of systems biology.

Reactive models come with an operational semantics; that is, every model defines a virtual machine with instructions for executing the model step by step. This is true not only of textual modeling and programming languages, but also of visual formalisms, such as Petri nets[62] and Statecharts.[38] This is in contrast with most equational and denotational models, where generating trajectories is a mathematical task that requires algorithmic and numeric insights. This benefit of reactive models has led to the term "executable biology."[32] An operational semantics is not only enormously helpful in simulation and reachability analysis, but also offers the possibility of interactive execution—or "playing *in silico*."[39]

A second important attribute of models of reactivity is that they offer a compact syntactic description of a dynamical system. The actual number of states and transitions of a biological system—its semantics—is usually very large or unbounded. Scientists and mathematicians often make little distinction between the semantics of a system, say, as a Markov process, and its description, say, as a transition probability matrix. By not differentiating sufficiently between the two, a potential critical advantage of syntax is lost; namely, that the description of a system can be much smaller than the system itself. For example, the rule-based description of a (bio)chemical system can be exponentially smaller than the matrix description of the same system; in fact, even many infinite state systems have finite descriptions. The syntax of a language can offer scaling operations, such as parallel composition and encapsulation, which greatly magnify this effect.

Syntax matters also in other ways. For one, the right choice of syntax can substantially improve the performance of analysis methods. Certain crucial optimizations of reachability analysis, such as on-the-fly state-space generation and partial-order reduction (things that can be extremely helpful when analyzing a piece of biology), are only available when the individual transitions of the underlying system are described compactly, by a syntactic expression (a rule, a process algebraic term, or a state machine) rather than a matrix equation. Furthermore, an inductively defined syntax, which features operators on basic expressions for constructing more complex expressions, offers the possibility of defining a structured operational semantics. In such semantics, the execution engine is defined compositionally; that is, it is put together from small primitives by using the syntactic operators of the language. Finally, a visual syntax makes modeling appeal to a larger group of people, such as biologists, and reactive models offer natural opportunities for visual representations.

State of the Art and Challenges

A large number of efforts to construct and analyze complete reactive models of various biological systems are under way. For lack of space, we can point only to a few of these efforts. We arrange them in a way that proceeds from more detailed, molecular-level models to more abstract, cell-level models. Alas, establishing formal relationships between the various levels, that would enable seamless combinations of executable models, still remains one of the key challenges in this area (see Kugler[52]).

Individual molecules can be modeled and combined as processes within a process algebra. Such *process-calculus models* stress the importance of concurrency and interaction between molecules as the main driver behind the dynamics of biological systems. Initial work suggested the use of the pi-calculus[59] as a modeling language for molecular interactions,[70] using it to study the cancer-related signal transduction pathway RTK-MAPK and to build the BioSPI simulation environment. The language was later extended to the stochastic pi-calculus[66] in order to model a gene-regulatory positive feedback loop.[68] These initial successes have led to the design of several bio-inspired and location-aware process calculi, such as BIO-PEPA, the ambient calculus,[69] and the brane calculus.[11] The methodology was applied, among others, to transcription factor activation and the glycolysis pathway,[18] RKIP inhibition of ERK,[9] the FGF pathway,[51] and EGFR signaling.[76] These approaches are very beneficial on the level of pathways and molecular interactions, but lack natural power of expression when dealing with larger biological systems, say, on the intercellular level.

On a higher level, instead of representing individual molecules as computational objects, we may refer to quantities of molecules through variables in a programming language. A single reaction may increment or decrement such variables. Inspired by guarded commands and reactive modules,[2] languages in this style were used for building qualitative models[7,32] and discrete-time Markov processes.[50] They were later extended to continuous time, for describing biochemical reaction networks. Such *transition-class models* may use general arithmetic expressions for specifying reaction rates.[45] These models can be executed (interpreted) directly, without the need for constructing a separate simulation engine. They can compactly represent unbounded quan-

tities of molecules. It is their executability and compactness that allows the stochastic analysis and model checking of complex molecular systems, often involving many different molecule types and very large molecule quantities, such as a genetic toggle switch. Further extensions lead to hybrid systems, handling particularly large quantities in a continuous domain.[36,45]

Another class of languages for modeling biology is based on term-rewrite systems.[24] *Rule-based models* can offer an even more compact syntax than guarded-command definitions of molecular reactions, by defining the reactive behavior of molecules in terms of what happens at individual binding sites within molecules. By formulating rewrite rules whose patterns are matched against fragments of molecules, one can avoid referring explicitly to the state of an entire molecule, and instead specify only the state of the affected sites before and after the application of the rule. Such rules, which may simultaneously apply to many different sites within a single complex molecule, can lead to a further reduction in the size of the description of a model. Rule-based modeling has been applied to an increasing number of systems such as signal transduction in the immune system,[29,54,55] bacterial migration,[6,61] cancer-related signaling,[9,21] mechanisms by which various proteins regulate cell signaling through their association with membrane proteins,[44] and more.[48] Ideas from programming languages, such as abstract interpretation, have influenced the design of more recent rule-based languages for biological applications.[31]

At the highest, non-molecular level, *state-machine based models* provide a visual approach for defining the behavior of complex objects, such as collections of cells, over time. Interaction mechanisms between state machines can specify causal relationships between events and state changes in different objects. A hierarchical structure allows one to view a system at different levels of detail (for example, whole organism, tissues, cells). For example, the language of Statecharts supports interacting and hierarchical state-based modeling,[39] based on a visual syntax, and has been used to model immune-cell activation and differentiation;[25,49] cellular decision-making processes during animal development;[32,35,50,72] as well as organ formation.[74] State-machine models are particularly suitable for describing mechanistic models of multicellular systems that are well-understood qualitatively. Such models do not require detailed quantitative data relating to the number of molecules and reaction rates, and indeed are inadequate when it comes to modeling pathways and molecular interactions. The possibility of hierarchical structuring is particularly useful in cases where behavior is distributed over many cells and where multiple copies of the same process are executed in parallel. These models also allow the application of strong analysis tools such as model checking. Combined methods seem very promising when it comes to systems for which one wants to model intercellular as well as intermolecular behavior, such as the Biocharts approach that is sketched in Kugler, Larjo and Harel.[53]

The research directions described in this article are intended, first and foremost, to yield beneficial results in biology and medicine, thus enhancing our ability to improve our lives.

There are many additional efforts in modeling biological systems that have not found their way into this article, mainly because they are less along the reactive system lines presented here. Some of these are particularly exciting and insightful. They include abstract chemical machines,[13] work on the brane and ambient calculi mentioned earlier, and other efforts to integrate behavior directly with space and movement considerations. In addition, the reader is referred to Priami's recent article in *Communications*[68] for a different perspective, more algorithmic, and thus somewhat complementary to ours.

Are we doing science or engineering? It is worth briefly addressing the connections between biological modeling and both science and engineering. Our discussion follows recent insights voiced by Luca Cardelli.

In science and engineering we find notions of "systems," which are the objects of study, and "models," which are formal or semiformal descriptions of those systems. The *scientific method* starts from a given natural system of interest, and through a discovery process we gain knowledge about the system, until we are in a position to formulate a model that aims to characterize its important features. Scientists then attempt to falsify that model, showing that it is inaccurate or incorrect, usu-

ally by experimentation based on the model's predictions. This process of falsification is the defining characteristic of scientific models.[65] If it is successful, we have discovered a property of the system that is not captured correctly by the model, which may lead to an improved model and to a new falsification cycle. If it is unsuccessful, the model stands, which does not necessarily mean that it is correct: it simply means that it has not yet been proven wrong (and we should keep trying). A main feature of the scientific method is that the "truth" is in the system, while the model is in principle never fully correct.

This Popperian approach has been adopted as part of the recent idea of a Turing test aimed at biological modelling,[40] where the model is deemed valid if it cannot be told apart from the actual biology. Here, of course, we do not advocate comparison of the actual material, but just of the behavior (as is the case for Turing's original test for machine-generated intelligence). However, in contrast to Turing's original test, falsifying the model here is something that we actually strive for, since it is a wonderful way to encourage further research.

The *engineering method* starts by producing a model (for example, a blueprint, or a specification) of what we want to build, and proceeds by building it. We then aim to show that what we built is in fact an implementation of the model. Such a verification process compares the outcomes of the system to the predicted outcomes of the model, by testing and model checking, which is in many ways similar to scientific experimentation. If this is unsuccessful, it means that we have discovered a property of the model that is not correctly implemented by the system, which may lead to an improved system and a new verification cycle. If it is successful, the system stands, which does not mean that it is correct: it simply means that we have not yet found the next bug (and we should keep trying, by making the model/specification more complete). A main feature of the engineering method is that the "truth" is in the model, while the system is in principle never fully correct. Thus, science and engineering work in opposite directions.

Incidentally, *reverse engineering*, the process of deriving an unknown model from an existing system, follows very much the scientific method. Conversely, (direct) engineering could be also called reverse science. As an example of the difference within a single discipline, consider systems biology, which is largely a scientific enterprise, as opposed to synthetic biology, which is largely an engineering enterprise. Of course, there are strong interactions between science and engineering, with one inspiring the other. Many engineered systems are inspired by biological systems that have been scientifically investigated (for example, genetic algorithms, neural networks), and conversely, as we have argued, modeling biological systems can be inspired by modeling techniques in engineering.

Indeed, when considering complex systems, both in science and in engineering one is usually in a position where the model is so complex that it is in constant flux, and where new knowledge about the system is expanding so fast that it is difficult to tell what "the system" actually is. In these situations, what emerges is a joint iterative method, in which our understanding of the system continuously improves, as a result of the modeling being continuously refined, which is turn is done by discovering the discrepancies between system and model; and all this in an endless cycle. This situation is actually quite common in software engineering, possibly more than in any other branch of engineering, where the model (the specification) typically evolves while the system (the code) is being built. And this situation is also quite common in modern biology, where scientific discovery is closely coupled with the construction of artificial systems, for example, by genetic engineering, so that not even nature is taken as a given. In such complex situations we must combine the conflicting views of systems and models into a wider *scientific-engineering method*, which still works by two opposite cycles. Discovery can still be coupled with falsification (when starting from the system) and construction can still be coupled with verification (when starting from the model), but there is no longer a privileged starting point for the process. In this sense, computing and biology are already remarkably close to each other, in the kind of general methods they use to expand knowledge.

Modeling a complete organism. We feel that it might be beneficial to use the ideas and methods discussed here to model a complete biological system. In fact, a "grand challenge" of modeling a full multicellular organism has been proposed,[39] motivated by the belief that unprecedented depth of understanding life and its mechanisms will result from such a model. The dream is to model the organism as a reactive system, the backbone of which would be its multitude of cells and their interactions, but to include the relevant inner behavioral aspects of the cell on the molecular and biochemical level as well. The 1000-cell *Caenorhabditis elegans* nematode worm, better known simply as *C. elegans*, was suggested in Harel[39] as a possible system to model.

Obviously, this is less ambitious than modeling, say, the entire population of a species, and more ambitious than modeling a mere cell. The choice of which system to address is a matter of taste, but our feeling is that an organism would be a good compromise that would yield enormous benefits, if it can indeed be done satisfactorily. The question of when to stop, that is, when is the model deemed valid or complete, is a very interesting one, and we have proposed that the Turing test mentioned previously could be a good first approximation: We are done when the model's behavior cannot be distinguished from that of the real thing, in which case the model can be said to be a theory of the organism; see Harel.[40]

This *whole organism project* (WOP) would take many years of work, and would entail using a variety of methods and to interconnect them all smoothly into a full, true-to-all-known-facts, 4-dimensional model of the creature. We would want the model to be easily modifiable and extendable as new facts are discovered, to have an animated, anatomically correct front-end, which would have to be tightly linked to a reactive system model of the organism. The front-to-back linking could be done using the idea of *reactive animation*.[24] Most importantly, the model would enable realistic simulation of the organism's development and behavior (this is the fourth dimension), and would lend itself to the kinds of analysis techniques discussed earlier. All of this could help uncover gaps, correct errors, suggest

new experiments, and help predict unobserved phenomena. More generally, the expectation is that it would allow researchers to see and understand the organism, its development, and its behavior in ways not otherwise possible.

Of course, this idea might be far too vast to be practical, but it seems worthy of consideration, if only as a very distant holy grail of sorts, toward which it would be beneficial to aspire.

Challenges for computer science. The research directions described in this article are intended, first and foremost, to yield beneficial results in biology and medicine, thus enhancing our ability to improve our lives. The central challenges they raise are also biological in nature, involving the need for biology to become a more formal, precise, and quantitative science, and the need for acquiring and consolidating sufficient information about the biology of interest to model it as a reactive system. This is especially true of the WOP and the work that is necessary to lead up to it. However, most readers of this article are computer scientists, who will be primarily interested in the new challenges this area of work raises for computer science, and in the benefits it can yield "at home." The two are linked, of course: once our field rises to the relevant challenges, the new ideas that are found to work well in the modeling of complex biological systems will benefit the development of human-made computerized software and systems as well. So, what are the main challenges for computer science? What new ideas are needed, and what kinds of extensions should be sought for the methods used in the modeling efforts mentioned earlier?

Our feeling is that we need ways to build models that seamlessly combine qualitative and quantitative data, and which come with appropriately powerful analysis methods. And we need to find ways to make our models more robust and less sensitive to faults and gaps in the available data. In other words, not only biology needs to become a more quantitative science, also computer science needs to become more quantitative. Formal methods have excelled in structuring and handling large, complex discrete systems, but we have neglected the incorporation of quantitative data. Similarly, we need to move our focus away from Boolean properties of systems, such as correctness (which really has no meaning in biology), toward quantitative properties such as fitness, robustness, and resilience. We believe the study of such quantitative properties will greatly benefit computer science itself which, as an engineering discipline, ought to have ways of expressing and measuring quantitative preferences between different implementations of a system, and estimating their reliability, cost, and performance. Preliminary ideas in this direction can be found in Cerny et al.[13] Needless to say, by studying biological systems in this way, we may also learn a thing or two about building more adaptive and robust software and hardware systems.

Two major deficiencies of current reactive models that need to be researched thoroughly are *genericity* and *linkage*. Genericity is related to inheritance, but for temporal, reactive behavior. We would like to have a generic model of, say, a cell or a central intracell substance, and be able to specialize it to specific types of cells or substances in a relatively painless way; see Amir-Kroll et al.[3] for a preliminary attempt at this. As far as linkage is concerned, we attach great importance to developing means for linking heterogeneous parts of biological models both horizontally and vertically, to yield compound models that can be seamlessly visualized, executed, and analyzed. *Horizontal linkage* refers to compositionality—the ability to compose side-by-side parts of the desired model into a whole, which is a particular challenge when the individual parts have different execution semantics,[72] an issue that is central also to embedded-systems design.[46] *Vertical linkage* is related to abstraction, and is the ability to link higher levels of the model with lower levels, for example, models of the intracellular pathway and network information with models of the reactive intercellular effects.[52] Ideally, we hope to provide biologists with computational "experimentation environments," where they can effortlessly play in a cycle of changing the model and looking at the resulting behaviors, all the time zooming in and out between different levels. We believe that such experimentation environments will go a long way toward efficiently identifying new, interesting hypotheses, testing them first "in-silico," and ultimately comparing them with nature.

We hope this article will help increase interest, within the computer science community, in the process of modeling and analyzing biological systems, viewing them as reactive systems of the most complex and challenging kind. The potential benefits of this, we feel, are difficult to overestimate, and we believe that concepts and ideas from software and systems engineering can form the basis of such work. Computer science is thus poised to play a role in the science of the 21st century, which will be dominated by the life sciences, similar to the role played by mathematics in the science of the 20th century, much of which was dominated by the physical sciences.

Acknowledgments

We would like to thank our past collaborators on these topics, for the wisdom and ideas they have contributed to us over the years. They include Luca Cardelli, Yaron Cohen, Sol Efroni, Walter Fontana, Alex Hajnal, Jane Hubbard, Na'aman Kam, Maria Mateescu, Nir Piterman, Yaki Setty, Michael Stern, Verena Wolf, and the late Amir Pnueli.

This research was supported in part by the ERC Advanced Grant LIBPR (Liberating Programming) awarded to DH, by the John von Neumann Minerva Center for the Development of Reactive Systems at the Weizmann Institute of Science, and by the ERC Advanced Grant QUAREM (Quantitative Reactive Modeling), awarded to TAH.

References

1. Alur, R. and Dill, D. Automata for modeling real-time systems. In *Proc. Int. Conf. Automata, Languages, and Programming 17* (1990), 322–335.
2. Alur, R. and Henzinger, T.A. Reactive modules. *Formal Methods in System Design 15*, 7 (1999), 48.
3. Amir-Kroll, H., Sadot, A., Cohen, I.R. and Harel, D. GemCell: A generic platform for modeling multi-cellular biological systems. *Theoret. Comput. Sci. 391*, 3 (2008), 276–290.
4. Arkin, A., Ross, J. and McAdams, H.H. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics 149* (1998), 1633–1648.
5. Baker, M.D., Wolanin, P.M. and Stock, J.B. Signal transduction in bacterial chemotaxis. *Bioessays 28* (2006), 9–22.
6. Barjis, J. and Barjis, I. Formalization of the protein production by means of petri nets. *Proc. Int. Conf. on Information Intelligence Systems* (1999), IEEE.
7. Batt, G., Ropers, D., de Jong, H., Geiselman, J., Mateescu, R., Page, M., and Schneider, D. Validation

of qualitative models of genetic regulatory networks by model checking: Analysis of the nutritional stress response in Escherichia coli. *Bioinformatics 21* (2005), 19–28.
8. Blinov, M.L., Faeder, J.R., Goldstein, B. and Hlavacek, W.S. BioNetGen: Software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics 20* (2001), 3289–3291.
9. Calder, M., Vyshemirsky, V., Gilbert, D. and Orton, R. Analysis of signaling pathways using the prism model checker. In *Proc. 3rd International Conference on Computational Methods in Systems Biology*. G. Plotkin, ed. (Edinburgh, Scotland, 2005), 179–190.
10. Calzone, L., Fages, F. and Soliman, S. BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics 22*, 14 (2006), 1805–1807.
11. Cardelli, L. Brane caluli. In *Proc. Computational Methods in Systems Biology* (Paris, May 26, 2004) V. Danos and V. Schächter, eds. LNCS 3082, 257 (2004).
12. Cardelli, L. Abstract machines of systems biology. In *Transactions on Computational Systems Biology III*. C. Priami et al. eds. LNCS 3737, (2005), 145–168.
13. Cerny, P., Henzinger, T.A. and Radhakrishna, A. Simulation distances. In *Proc. Concurrency Theory 2010*. LNCS 2011, 253–268.
14. Clarke, E., Grumberg, O., Jha, S., Lu, Y. and Veith, H. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM 50*, (2003), 752–794.
15. Clarke, E., Grumberg, O. and Peled, D. *Model Checking*, MIT Press, 2000.
16. Cohen, I.R. and Harel, D. Explaining a complex living system: Dynamics, multi-scaling and emergence. *J. Royal Society Interface 4*, (2007), 175–182.
17. Cousot, P. and Cousot, R. Abstract interpretation. In *Proc. Symp. Principles of Programming Languages 4*, (1977), 238–252.
18. Curti, M., Degano, P., Priami, C. and Baldari, C. Modeling biochemical pathways through enhanced pi-calculus. *Theor. Comput. Sci. 325*, (2004), 111–140.
19. Damm, W. and Harel, D. LSCs: Breathing life into message sequence charts. *Formal Methods in System Design 19*, 1, (2001), 45–80.
20. Danos, V., Feret, J., Fontana, W., Harmer, R. and Krivine, J. Rule-based modelling of cellular signalling. *Concur* 2007, LNCS 4703, 17–41.
21. Didier, F., Henzinger, T.A., Mateescu, M. and Wolf, V. Approximation of event probabilities in noisy cellular processes. *Computational Methods in Systems Biology 7*, (2009) 173–188.
22. Didier, F., Henzinger, T.A., Mateescu, M. and Wolf, V. Fast adaptive uniformization of the chemical master equation. *High-Performance Computational Systems Biology 1*, (2009).
23. Dill, D.L., Knapp, M.A., Gage, P., Talcott, C., Laderoute, K. and Lincoln, P. The pathalyzer: A tool for analysis of signal transduction pathways. In *Proc. 1st Annual Recomb Satellite Workshop on Systems Biology*, 2005.
24. Efroni, S., Harel, D. and Cohen, I.R. Reactive animation: Realistic modeling of complex dynamic systems. *Computer 38*, (2005), 38–47.
25. Efroni, S., Harel, D., and Cohen, I.R. Emergent dynamics of thymocyte development and lineage determination. *PLoS Computational Biology 3*, 1 (2007), 127–136.
26. Elowitz, M.B., Levine, A.J., Siggia, E.D. and Swain, P.S. Stochastic gene expression in a single cell. *Science 297*, (2002), 1183–1186.
27. Emerson, E.A. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, (1990), 995–1072.
28. Faeder, J.R., Hlavacek, J.S., Reischl, I., Blinov, M.L., Metzger, H., Redondo, A., Wofsy, C. and Goldstein, B. Investigation of early events in FcεRI-mediated signaling using a detailed mathematical model. *J. Immunol. 170*, (2003), 3769–3781.
29. Fedoroff, N. and Fontana, W. Small numbers of big molecules. *Science 297*, (2002), 1129–1131.
30. Feret, J., Danos, V., Krivine, J., Harmer, R. and Fontana, W. Internal coarse-graining of molecular systems. *Proc. Natl. Acad. Sci. 106*, 16, (2009), 6453–6458.
31. Fisher, J., Piterman, N., Hajnal, A., and Henzinger, T.A. Predictive modeling of signaling crosstalk during C. elegans vulval development. *PLoS Computational Biology 3*, 5, (2007) 92.
32. Fisher, J. and Henzinger, T.A. Executable cell biology. *Nat. Biotechnol. 25*, 11, (2007), 1239–49.
33. Fisher, J., Henzinger, T.A., Mateescu, M. and Piterman, N. Bounded asynchrony: A biologically-inspired notion of concurrency. In *Proc. of FMSB '08 Cambridge, UK*. Springer, 2008.
34. Fisher, J., Piterman, N., Hubbard, E.J., Stern, M.J. and Harel, D. Computational insights into Caenorhabditis elegans vulval development. In *Proc. Natl. Acad. Sci. USA 102*, (2005), 1951–1956.
35. Ghosh, R. and Tomlin, C.J. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modeling: Delta-Notch protein signaling. *IEE Transactions on Systems Biology 1*, 1, (2004), 170–183.
36. Gillespie, D.T. Exact stochastic simulation of coupled chemical reactions. *J. of Physical Chemistry 81*, (1977), 2340–2361.
37. Gillespie, D.T. Markov Processes. 1992: Academic Press.
38. Harel, D. Statecharts: A visual formalism for complex systems. *Sci. Comput. Programming 8*, (1987), 231–274.
39. Harel, D. A grand challenge for computing: Full reactive modeling of a multi-cellular animal. Bulletin of the EATCS 81, (2003), 226-235. (Reprinted in *Current Trends in Theoretical Computer Science: The Challenge of the New Century, Algorithms and Complexity*, Vol I. G. Paun et al. eds. World Scientific, (2004), 559–568.
40. Harel, D. A Turing-like test for biological modelling. *Nature Biotechnology 23*, (2005), 495–496.
41. Harel, D. and Gery, E. Executable object modeling with statecharts. *Computer 30*, 7, (July 1997). IEEE Press, 31–42.
42. Harel, D. and Pnueli, A. On the development of reactive systems. In *Logics and Models of Concurrent Systems*. K.R. Apt, ed. NATO ASI Series, Vol. F-13, Springer-Verlag, NY, (1985), 477–498.
43. Haugh, J.M., Schneider, I.C. and Lewis, J.M. On the cross-regulation of protein tyrosine phosphatases and receptor tyrosine kinases in intracellular signaling. *J. Theor. Biol. 230*, (2004), 119–132.
44. Henzinger, T.A., Jobstmann, B. and Wolf, V. Formalisms for specifying Markovian population models. In *Proc. 3rd Int. Workshop on Reachability Problems*. LNCS 5797, Springer, 2009.
45. Henzinger, T.A., Mateescu, M., Mikeev, L. and Wolf, V. Hybrid numerical solution of the chemical master equation. In *Proc. 8th Int. Conf. on Computational Methods in Systems Biology*. Lecture Notes in Bioinformatics, Springer, 2010.
46. Henzinger, T.A., and Sifakis, J. The discipline of embedded systems design. *IEEE Computer 40*, 10, (2007), 36–44.
47. Hlavacek, W.S., Faeder, J.R., Blinov, M.L., Posner, R.G., Hucka, M., and Fontana, W. Rules for modelling signal transduction systems. *Science STKE* 2006/344/re6.
48. Kam, N., Harel, D. and Cohen, I.R. *Visual Languages and Formal Methods*. (Stressa, Italy, Sept. 5–7, 2001). IEEE, 2001.
49. Kam, N., Kugler, H., Marelly, R., Appleby, L., Fisher, J., Pnueli, A., Harel, D., Stern, M.J. and Hubbard, E.J.A. Scenario-based approach to modeling development: A prototype model of *C. Elegans* vulval cell fate specification. *Developmental Biology 323* (2008), 1–5.
50. Kwiatkowska, M., Norman, G. and Parker, D. PRISM: Probabilistic symbolic model checker. In *Proc. TOOLS 2002*. T. Field et al. eds. LNCS 2324, (2002), 200–204.
51. Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O., Heath, J., and Gaffney, E. Simulation and verification for computational modeling of signalling pathways. In *Proc. Winter Simulation Conference* (Monterey, CA, Dec. 2–6, 2006). IEEE, 2006, 1666–1674 .
52. Kugler, H., Larjo, A. and Harel, D. Biocharts: A visual formalism for complex biological systems. *J. Royal Society Interface*, 2010.
53. Lee, K.H., Dinner, A.R., Tu, C., Campi, G., Raychaudhuri, S., Varma, R., Sims, T.N., Burack, W.R., Wu, H., Wang, J., Kanagawa, O., Markiewicz, M., Allen, P.M., Dustin, M.L., Chakraborty, A.K. and Shaw, A.S. The immunological synapse balances T cell receptor signaling and degradation. *Science 302*, (2003), 1218–1222.
54. Li, Q.J., Dinner, A.R., Qi, S., Irvine, D.J., Huppa, J.B., Davis, M.M., and Chakraborty, A.K. CD4 enhances T cell sensitivity to antigen by coordinating Lck accumulation at the immunological synapse. *Nat. Immunol. 5*, (2004)791–799.
55. McAdams, H.H. and Arkin, A. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Science 94*, (1997), 814–819.
56. McAdams, H.H. and Arkin, A. It's a noisy business! *Trends in Genetics 15*, 2 (1999), 65–69.
57. Milner, R. *A Calculus of Communicating Systems*. Springer Verlag, 1980
58. Milner, R. Operational and algebraic semantics of concurrent processes. *Handbook of Theoretical Computer Science B*, (1990), 1201–1242.
59. Milner, R. *Communicating and Mobile Systems: The pi-Calculus*. Cambridge University Press, Cambridge, UK, 1999).
60. Parkinson, J.S., Ames, P. and Studdert, C.A. Collaborative signaling by bacterial chemoreceptors. *Curr. Opin. Microbiol. 8*, (2005), 116–121.
61. Paulsson, J. Summing up the noise in gene networks. *Nature 427*, (2004), 415–418.
62. Peterson, J.L. *Petri Net Theory and the Modeling of Systems*. Prentice Hall 1981
63. Pnueli, A. The temporal logic of programs. In *Proc. Symp. Found. Computer Science*, (1977) 46–57.
64. Pnueli, A. and Rosner, R. On the synthesis of a reactive module. In *Proc. Symp. Principles of Programming Languages 16*, (1989), 179–190.
65. Popper, K. *The logic of scientific discovery*. Hutchinson, London, 1959.
66. Priami, C. The stochastic pi-calculus. *Comp. J. 38*, (1995), 578–589.
67. Priami, C. Algorithmic systems biology. *Comm. ACM 52*, 5, (May 2009) 80–88.
68. Priami, C., Regev, A., Shapiro, E.Y., and Silverman, W. Application of a stochastic name passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett. 80*, (2001) 25–31.
69. Regev, A., Panina, E.M., Silverman, W., Cardelli, L., and Shapiro, E.Y. Bioambients: An abstraction for biological compartments. *Theor. Comput. Sci. 325*, (2004), 141–167.
70. Regev, A., Silverman, W., and Shapiro, E. Representation and simulation of biochemical processes using the pi-calculus process algebra. *Pac. Symp. Biocomput.* (2001) 459–470.
71. Sadot, A., Fisher, J., Barak, D., Admanit, Y., Stern, M.J., Hubbard, E.J.A. and Harel, D. Towards verified biological models, *Transactions on Computational Biology and Bioinformatics 5*,(2008), 1–12.
72. Schaub, M.A., Henzinger, T.A., and Fisher, J. Qualitative networks: A symbolic approach to analyze biological signaling networks. *BMC Systems Biology 1*, (2007).
73. Setty, Y., Cohen, I.R., Dor, Y., and Harel, D. Four-dimensional realistic modeling of pancreatic organogenesis. In *Proc. Natl. Acad. Sci. 105*, 51 (2008), 20374–20379.
74. Shen, X., Collier, J., Dill, D., Shapiro, L., Horowitz, M. and McAdams, H.H. Architecture and inherent robustness of a bacterial cell-cycle control system. *PNAS. 105*, 32 (2008), 11340–11345.
75. Thomas, W. Automata on infinite objects. In *Handbook of Theoretical Computer Science B*, (1990), 133–192.
76. Wang, D., Cardelli L., Phillips A., Piterman N. and Fisher J. Computational modeling of the EGFR network elucidates control mechanisms regulating signal dynamics. In *BMC Systems Biology 3*, 118, (2009), 22.
77. Wolf, V., Goel, R., Mateescu, M. and Henzinger, T.A. Solving the chemical master equation using sliding windows. *BMC Systems Biology 4*, 42, (2010).

Jasmin Fisher (Jasmin.Fisher@microsoft.com) is a researcher at Microsoft Research Cambridge in the Programming Principles and Tools Group, Cambridge, UK.

David Harel (dharel@weizmann.ac.il) is the William Sussman Professorial Chair of the Dept. of Computer Science and Applied Mathematics at the Weizmann Institute of Science, Rehovot, Israel.

Thomas A. Henzinger (tah@ ist.ac.at) is president of the Institute of Science and Technology Austria (IST Austria) and an adjunct professor of electrical engineering and computer sciences at the University of California, Berkeley.

research highlights

DOI:10.1145/2001269.2001290

Technical Perspective
Power Efficiency as the #1 Design Constraint

By Charles Moore

MOORE'S LAW,[2] AND the associated observations on scaling by Bob Dennard,[1] describe many of the key technical foundations that have given rise to the amazing growth of the modern semiconductor industry. But, taking a step back from these insightful assertions, we can see an even bigger picture emerge related to energy usage and power consumption. The earliest computers, such as the machine envisioned by Charles Babbage in 1822, were mechanical marvels. Although simultaneously amazing and underappreciated, they certainly set the stage for modern computing era.

As it turns out, these machines consumed very large amounts of power (for the time), and were quickly replaced by more power-efficient, electro-mechanical, relay-based systems. This shift enabled larger machines with more capability, but they too soon hit the practical "power wall" of the time. In 1946, the first vacuum tube-based computers were designed, and once again, these set a new standard in capability and power efficiency for the day, eventually replacing the relay-based systems. This pattern repeated with the invention of the discrete transistor, the integrated circuit, the bipolar-based integrated circuits, and the FET-based integrated circuits. The key point here is that these technology transitions were driven in equal parts by the new capability they brought to light *and* the improved power efficiency they offered.

That brings us to modern VLSI-based systems. Once again, our current technology of choice, CMOS-based integrated circuits in particular, has hit a modern day "power wall." There are, of course, lots of highly innovative process technology tweaks and circuit design tricks to extend the life of the CMOS-based era, but these are really only buying time until a fundamentally new technology option becomes practical. Unfortunately, although there are several interesting contenders, it doesn't appear any of these will be practical within the next decade.

While the technologists and circuit designers continue to extend the life of CMOS, it is also time for computer architects to innovate on fundamentally more power-efficient algorithms and machine organizations. The following paper by Hameed et al. provides a deep dive into a specific application to highlight some of the most significant differences in power efficiency between a general-purpose processor and a special-purpose ASIC. By looking at a range of design options associated with improving the power efficiency of a general-purpose processor running H.264 HD video encode, they uncover an option that effectively uses the instruction orchestration aspect of the general-purpose processor to control the sequencing through a pipeline of customized special-purpose blocks. The new complexity associated with these customized logic blocks notwithstanding, they illustrate a power efficiency improvement of nearly three orders of magnitude.

This paper uncovers some of the most significant differences in power efficiency between a general-purpose processor and a special-purpose ASIC.

The power efficiency and performance advantages of special-purpose ASICs versus general-purpose processors in not new, nor is it surprising. In fact, the essence of these differences is reflected in the computer architects' mantra of "optimizing for the common case." In the 1968 seminar paper "*On the Design of Display Processors*" by T.H. Myer and Ivan Sutherland,[3] this trade-off between generality and complexity is particularly well described with relevant examples from that time period. In particular, they observe that the appropriate solution is both application and technology dependent, and from that, they coin the phrase "*The Wheel of Reincarnation*" to illustrate these shifting optimizations.

But this paper goes a step beyond the general observation and quantitative analysis of a particular application. It also sets the stage for designing future machines that are prepared for higher-level hardware abstractions. This proposal implies some profound implications for application analysis, algorithm design, machine organization, and associated design methodologies. Combined, they may offer improvements in power efficiency, raw performance, and design productivity. This triple play is particularly significant at this point in time, as the industry must simultaneously work around the ongoing CMOS "power wall" while also investing to find that next technology to reset the power-efficiency bar. C

References

1. Dennard, R. et al. Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE Journal of Solid State Circuits SC-9*, 5 (Oct. 1974).
2. Moore, G. Cramming more components onto integrated circuits. *Electronics Magazine*, Apr. 1965.
3. Myer, T.H., Sutherland, I.E. On the design of display processors. *Commun. ACM 11*, 6 (June 1968).

Charles Moore (chuck.moore@amd.com) is a Corporate Fellow at AMD, and the CTO for AMD's Technology Group.

DOI:10.1145/2001269.2001291

Understanding Sources of Inefficiency in General-Purpose Chips

By Rehan Hameed, Wajahat Qadeer, Megan Wachs, Omid Azizi, Alex Solomatnikov, Benjamin C. Lee, Stephen Richardson, Christos Kozyrakis, and Mark Horowitz

Abstract

Scaling the performance of a power limited processor requires decreasing the energy expended per instruction executed, since energy/op * op/second is power. To better understand what improvement in processor efficiency is possible, and what must be done to capture it, we quantify the sources of the performance and energy overheads of a 720p HD H.264 encoder running on a general-purpose four-processor CMP system. The initial overheads are large: the CMP was 500× less energy efficient than an Application Specific Integrated Circuit (ASIC) doing the same job. We explore methods to eliminate these overheads by transforming the CPU into a specialized system for H.264 encoding. Broadly applicable optimizations like single instruction, multiple data (SIMD) units improve CMP performance by 14× and energy by 10×, which is still 50× worse than an ASIC. The problem is that the basic operation costs in H.264 are so small that even with a SIMD unit doing over 10 ops per cycle, 90% of the energy is still overhead. Achieving ASIC-like performance and efficiency requires algorithm-specific optimizations. For each subalgorithm of H.264, we create a large, specialized functional/storage unit capable of executing hundreds of operations per instruction. This improves energy efficiency by 160× (instead of 10×), and the final customized CMP reaches the same performance and within 3× of an ASIC solution's energy in comparable area.

1. INTRODUCTION

Most computing systems today are power limited, whether it is the 1 W limit of a cell phone system on a chip (SoC), or the 100 W limit of a processor in a server. Since power is ops/second * energy/op, we need to decrease the energy cost of each op if we want to continue to scale performance at constant power. Traditionally, chip designers were able to make increasingly complex designs both by increasing the system power, and by leveraging the energy gains from technology scaling. Historically each factor of 2 in scaling made each gate evaluation take 8× less energy.[7] However, technology scaling no longer provides the energy savings it once did,[9] so designers must turn to other techniques to scale energy cost. Most designs use processor-based solutions because of their flexibility and low design costs, however, these are usually not the most energy-efficient solutions. A shift to multi-core systems has helped improve the efficiency of processor systems but that approach is also going to hit a limit pretty soon.[8] On the other hand, using hardware that has been customized for a specific application (an Application Specific Integrated Circuit or ASIC) can be three orders of magnitude better than a processor in both energy/op and ops/area.[6] This paper compares ASIC solutions to processor-based solutions, to try to understand the sources of inefficiency in general-purpose processors. We hope this information will prove to be useful both for building more energy-efficient processors and understanding why and where customization must be used for efficiency.

To build this understanding, we start with a single video compression application, 720p HD H.264 video encode, and transform the hardware it runs on from a generic multiprocessor to a custom multiprocessor with ASIC-like specialized hardware units. On this task, a general-purpose software solution takes 500× more energy per frame and 500× more area than an ASIC to reach the same performance. We choose H.264 because it demonstrates the large energy advantage of ASIC solutions (500×) and because there exist commercial ASICs that can serve as a benchmark. Moreover, H.264 contains a variety of computational motifs, from highly data-parallel algorithms (motion estimation) to control intensive ones (Context Adaptive Binary Arithmetic Coding [CABAC]).

To better understand the potential of producing general-purpose chips with better efficiency, we consider two broad strategies for customized hardware. The first extends the current trend of creating general data-parallel engines on our processors. This approach mimics the addition of SSE instructions, or the recent work in merging graphic processors on die to help with other applications. We claim these are similar to general functional units since they typically have some special instructions for important applications, but are still generally useful. The second approach creates application-specific data storage fused with functional units. In the limit this should be an ASIC-like solution. The first has the advantage of being a programmable solution, while the second provides potentially greater efficiency.

The results are striking. Starting from a 500× energy penalty, adding relatively wide SSE-like parallel execution engines and rewriting the code to use them improves performance/

A previous version of this paper was published in *Proceedings of the 37th Annual International Symposium on Computer Architecture* (2010), ACM, NY.

area by 14× and energy efficiency by 10×. Despite these customizations, the resulting solution is still 50× less energy efficient than an ASIC. An examination of the energy breakdown in the paper clearly demonstrates why. Basic arithmetic operations are typically 8–16 bits wide, and even when performing more than 10 such operations per cycle, arithmetic unit energy comprises less than 10% of the total. One must consider the energy cost of the desired operation compared with the energy cost of one processor cycle: for highly efficient machines, these energies should be similar.

The next section provides the background needed to understand the rest of the paper. Section 3 then presents our experimental methodology, describing our baseline, generic H.264 implementation on a Tensilica CMP. The performance and efficiency gains are described in Section 4, which also explores the causes of the overheads and different methods for addressing them. Using the insight gained from our results, Section 5 discusses the broader implications for efficient computing and supporting application driven design.

2. BACKGROUND

We first review the basic ways one can analyze power, and some previous work in creating energy-efficient processors. With this background, we then provide an overview of H.264 encoding and its main compute stages. The section ends by comparing existing hardware and software implementations of an H.264 encoder.

2.1. Power-constrained design and energy efficiency

Power is defined to be energy per second, which can be broken up into two terms, energy/op * ops/second. Thus there are two primary means by which a designer can reduce power consumption: reduce the number of operations per second or reduce the energy per operation. The first approach—reducing the operations per second—simply reduces performance to save power. This approach is analogous to slowing down a factory's assembly line to save electricity costs; although power consumption is reduced, the factory output is also reduced and the energy used (i.e., the electricity bill) per unit of output remains unchanged. If, on the other hand, a designer wishes to maintain or improve the performance under a fixed power budget, a reduction in the fundamental energy per operation is required. It is this reduction in energy per operation—not power—that represents real gains in efficiency.

This distinction between power and energy is an important one. Even though designers typically face physical *power* constraints, to increase efficiency requires that the fundamental *energy* of operations be reduced. Although one might be tempted to report power numbers when discussing power efficiency, this can be misleading if the performance is not also reported. What may seem like a power efficiency gain may just be a modulation in performance. Using energy per operation, however, is a performance-invariant metric that represents the fundamental efficiency of the work being done. Thus, even though the designer may be facing a *power* constraint, it is *energy per operation* that the designer needs to focus on improving.

Reducing the energy required for the basic operation can be achieved through a number of techniques, all of which fundamentally reduce the overhead affiliated with the work being done. As one simple example, clock gating improves energy efficiency by eliminating spurious activity in a chip that otherwise causes energy waste.[8] As another example, customized hardware can increase efficiency by eliminating overheads. The next section further discusses the use of customization.

2.2. Related work in efficient computing

Processors are often customized to improve their efficiency for specific application domains. For example, SIMD architectures achieve higher performance for multimedia and other data-parallel applications, while DSP processors are tailored for signal-processing tasks. More recently, ELM[1] and AnySP[24] have been optimized for embedded and mobile signal processing applications, respectively, by reducing processor overheads. While these strategies target a broad spectrum of applications, special instructions are sometimes added to speed up specific applications. For example, Intel's SSE4[10] includes instructions to accelerate matrix transpose and sum-of-absolute-differences.

Customizable processors allow designers to take the next step, and create instructions tailored to applications. Extensible processors such as Tensilica's Xtensa provide a base design that the designer can extend with custom instructions and datapath units.[15] Tensilica provides an automated ISA extension tool,[20] which achieves speedups of 1.2× to 3× for EEMBC benchmarks and signal processing algorithms.[21] Other tools have similarly demonstrated significant gains from automated ISA extension.[4, 5] While automatic ISA extensions can be very effective, manually creating ISA extensions gives even larger gains: Tensilica reports speedups of 40× to 300× for kernels such as FFT, AES, and DES encryption.[18, 19, 22]

Recently researchers have proposed another approach for achieving energy efficiency—reducing the cost of creating customized hardware rather than customizing a processor. Examples of the latter include using higher levels of abstraction (e.g., C-to-RTL[13]) and even full chip generators using extensible processors.[16] Independent of whether one customizes a processor, or creates customized hardware, it is important to understand in quantitative terms the types and magnitudes of energy overheads in processors.

While previous studies have demonstrated significant improvements in performance and efficiency moving from general-purpose processors to ASICs, we explore the reasons for these gains, which is essential to determine the nature and degree of customization necessary for future systems. Our approach starts with a generic CMP system. We incrementally customize its memory system and processors to determine the magnitude and sources of overhead eliminated in each step toward achieving a high efficiency 720p HD H.264 encoder. We explore the basic computation in H.264 next.

2.3. H.264 computational motifs

H.264 is a block-based video encoder which divides each

video frame into 16×16 macro-blocks and encodes each one separately. Each block goes through five major functions:

i. IME: Integer Motion Estimation
ii. FME: Fractional Motion Estimation
iii. IP: Intra Prediction
iv. DCT/Quant: Transform and Quantization
v. CABAC: Context Adaptive Binary Arithmetic Coding

IME finds the closest match for an image block versus a previous reference image. While it is one of the most compute intensive parts of the encoder, the basic algorithm lends itself well to data-parallel architectures. On our base CMP, IME takes up 56% of the total encoder execution time and 52% of total energy.

The next step, FME, refines the initial match from integer motion estimation and finds a match at quarter-pixel resolution. FME is also data parallel, but it has some sequential dependencies and a more complex computation kernel that makes it more difficult to parallelize. FME takes up 36% of the total execution time and 40% of total energy on our base CMP design. Since FME and IME together dominate the computational load of the encoder, optimizing these algorithms is essential for an efficient H.264 system design.

IP uses previously encoded neighboring image blocks within the current frame to form an alternate prediction for the current image-block. While the algorithm is still dominated by arithmetic operations, the computations are much less regular than the motion estimation algorithms. Additionally, there are sequential dependencies not only within the algorithm but also with the transform and quantization function.

Next, in DCT/Quant, the difference between a current and predicted image block is transformed and quantized to generate coefficients to be encoded. The basic function is relatively simple and data parallel. However, it is invoked a number of times for each 16×16 image block, which calls for an efficient implementation. For the rest of this paper, we merge these operations into the IP stage. The combined operation accounts for 7% of the total execution time and 6% of total energy.

Finally, CABAC is used to entropy-encode the coefficients and other elements of the bit-stream. Unlike the previous algorithms, CABAC is sequential and control dominated. While it takes only 1.6% of the execution time and 1.7% of total energy on our base design, CABAC often becomes the bottleneck in parallel systems due to its sequential nature.

2.4. Current H.264 implementations

The computationally intensive H.264 encoding algorithm poses a challenge for general-purpose processors, and is typically implemented as an ASIC. For example, T. C. Chen et al. implement a full-system H.264 encoder and demonstrate that real-time HD H.264 encoding is possible in hardware using relatively low power and area cost.[2]

H.264 software optimizations exist, particularly for motion estimation, which takes most of the encoding time. For example, sparse search techniques speed performance of IME and FME by up to 10×.[14, 25] Combining aggressive algorithmic modifications with multiple cores and SSE extensions leads to highly optimized H.264 encoders on Intel processors.[3, 12]

Despite these optimizations, software implementations of H.264 lag far behind dedicated ASICs. Table 1 compares a software implementation of a 480p SD encoder[12] to a 720p HD ASIC implementation.[2] The software implementation employs a 2.8 GHz Intel Pentium 4 executing highly optimized SSE code. This results in very high energy consumption and low area efficiency. It is also worth noting that the software implementation relies on various algorithmic simplifications, which drastically reduce the computational complexity, but result in a 20% decrease in compression efficiency for a given SNR. The ASIC hardware, on the other hand, consumes over 500× less energy and is far more efficient in its use of silicon area and has a negligible drop in compression efficiency.

3. EXPERIMENTAL METHODOLOGY

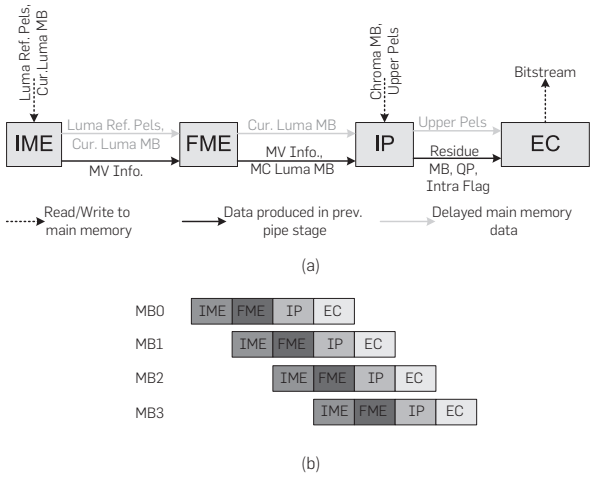
To understand what is needed to gain ASIC level efficiency, we use existing H.264 partitioning techniques, and modify the H.264 encoder reference code JM 8.6[11] to remove dependencies and allow mapping of the five major algorithmic blocks to the four-stage macro-block (MB) pipeline shown in Figure 1. This mapping exploits task level parallelism at the macro-block level and significantly reduces the inter-processor communication bandwidth requirements by sharing data between pipeline stages.

Table 1. Intel's optimized H.264 encoder versus a 720p HD ASIC.

	FPS	Area (mm²)	Energy/Frame (mJ)
Intel (720 × 480 SD)	30	122	742
Intel (1280 × 720 HD)	11	122	2023
ASIC	30	8	4

The second row gives Intel's SD data scaled to HD. ASIC data is scaled from 180 down to 90nm.

Figure 1. Four stage macroblock partition of H.264. (a) Data flow between stages. (b) How the pipeline works on different macroblocks. IP includes DCT+Quant. EC is CABAC.



In the base system, we map this four-stage macro-block partition to a four-processor CMP system where each processor has 16KB 2-way set associative instruction and data caches. Figure 2 highlights the large efficiency gap between our base CMP and the reference ASIC for individual 720p HD H.264 subalgorithms. The energy required for each RISC instruction is similar and as a result, the energy required for each task (shown in Figure 3) is related to the cycles spent on that task. The RISC implementation of IME, which is the major contributor to performance and energy consumption, has a performance gap of 525× and an energy gap of over 700× compared to the ASIC. IME and FME dominate the overall energy and thus need to be aggressively optimized. However, we also note that while IP, DCT, Quant, and CABAC are much smaller parts of the total energy/delay, even they need about 100× energy improvement to reach ASIC levels.

At approximately 8.6B instructions to process 1 frame, our base system consumes about 140 pJ per instruction—a reasonable value for a general-purpose system. To further analyze the energy efficiency of this base CMP implementation we break the processor's energy into different functional units as shown in Figure 4. This data makes it clear how far we need to go to approach ASIC efficiency. The energy spent in instruction fetch (IF) is an overhead due to the programmable nature of the processors and is absent in a custom hardware state machine, but eliminating all this overhead only increases the energy efficiency by less than one third. Even if we eliminate everything but the functional unit energy, we still end up with energy savings of only 20×—not nearly enough to reach ASIC levels.

Figure 2. The performance and energy gap for base CMP implementation when compared to an equivalent ASIC. Intra combines IP, DCT, and Quant.

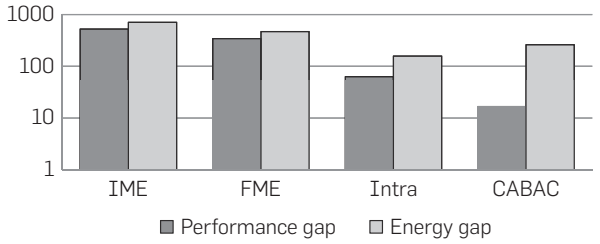
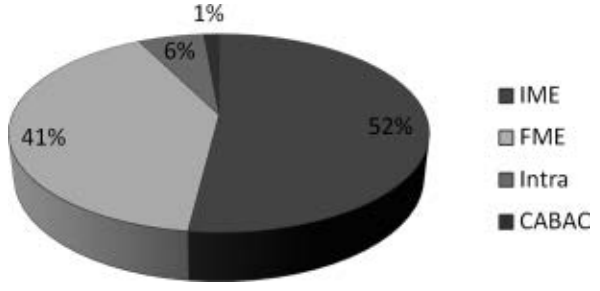


Figure 3. Processor energy breakdown for base implementation, over the different H.264 subalgorithms. Intra combines IP, DCT, and Quant.



The next section explores what customizations are needed to reach the efficiency goals.

4. CUSTOMIZATION RESULTS

At first, we restrict our customizations to datapath extensions inspired by GPUs and Intel's SSE instructions. Such extensions are relatively general-purpose data-parallel optimizations and consist of single instruction, multiple data (SIMD) and multiple instruction issue per cycle (we use long instruction word, or LIW), with a limited degree of algorithm-specific customization coming in the form of operation fusion—the creation of new instructions that combine frequently occurring sequences of instructions. However, much like their SSE and GPU counterparts, these new instructions are constrained to the existing instruction formats and datapath structures. This step represents the datapaths in current state-of-the-art optimized CPUs. In the next step, we replace these generic datapaths by custom units, and allow unrestricted tailoring of the datapath by introducing arbitrary new compute operations as well as by adding custom register file structures.

The results of these customizations are shown in Figures 5 through 7. The rest of this section describes these results in detail and evaluates the effectiveness of these three customization strategies. Collectively, these results describe how efficiencies improve by 170× over the baseline of Section 3.

Figure 4. Processor energy breakdown for base implementation. IF is instruction fetch/decode. D-$ is data cache. P Reg includes the pipeline registers, buses, and clocking. Ctrl is miscellaneous control. RF is register file. FU is the functional units.

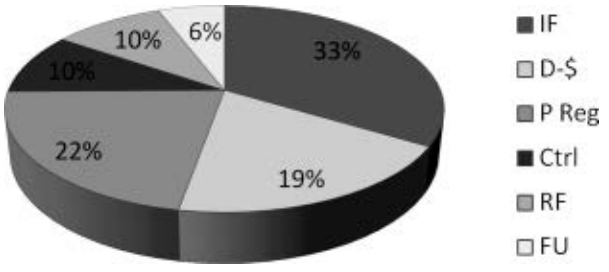
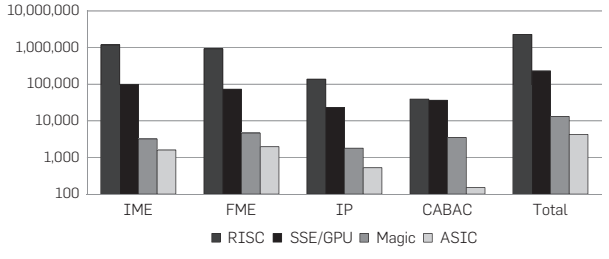


Figure 5. Each set of bar graphs represents energy consumption (μJ) at each stage of optimization for IME, FME, IP and CABAC respectively. The first bar in each set represents base RISC energy; followed by RISC augmented with SSE/GPU style extensions; and then RISC augmented with "magic" instructions. The last bar in each group indicates energy consumption by the ASIC.



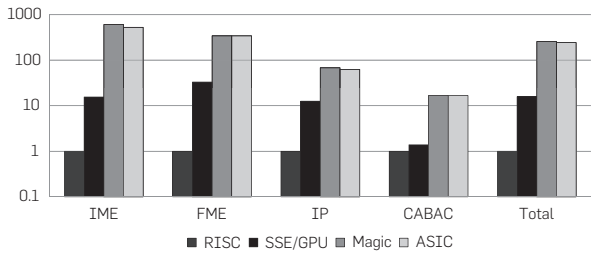
4.1. SSE/GPU style enhancements

Using Tensilica's TIE extensions we add LIW instructions and SIMD execution units with vector register files of custom depths and widths. A single SIMD instruction performs multiple operations (8 for IP, 16 for IME, and 18 for FME), reducing the number of instructions and consequently reducing IF energy. LIW instructions execute 2 or 3 operations per cycle, further reducing cycle count. Moreover, SIMD operations perform wider register file and data cache accesses which are more energy efficient compared to narrower accesses. Therefore all components of instruction energy depicted in Figure 4 get a reduction through the use of these enhancements.

We further augment these enhancements with operation fusion, in which we fuse together frequently occurring complex instruction sub-graphs for both RISC and SIMD instructions. To prevent the register file ports from increasing, these instructions are restricted to use up to two input operands and can produce only one output. Operation fusion improves energy efficiency by reducing the number of instructions and also reducing the number of register file accesses by internally consuming short-lived intermediate data. Additionally, fusion gives us the ability to create more energy-efficient hardware implementations of the fused operations, e.g., multiplication implemented using shifts and adds. The reductions due to operation fusion are less than 2× in energy and less than 2.5× in performance.

With SIMD, LIW and Op Fusion support, IME, FME and IP processors achieve speedups of around 15×, 30× and 10×, respectively. CABAC is not data parallel and benefits only from LIW and op fusion with a speedup of merely 1.1× and almost no change in energy per operation. Overall, the application gets an energy efficiency gain of almost 10×, but still uses greater than 50× more energy than an ASIC. To reach ASIC levels of efficiency, we need a different approach.

Figure 6. Speedup at each stage of optimization for IME, FME, IP and CABAC.

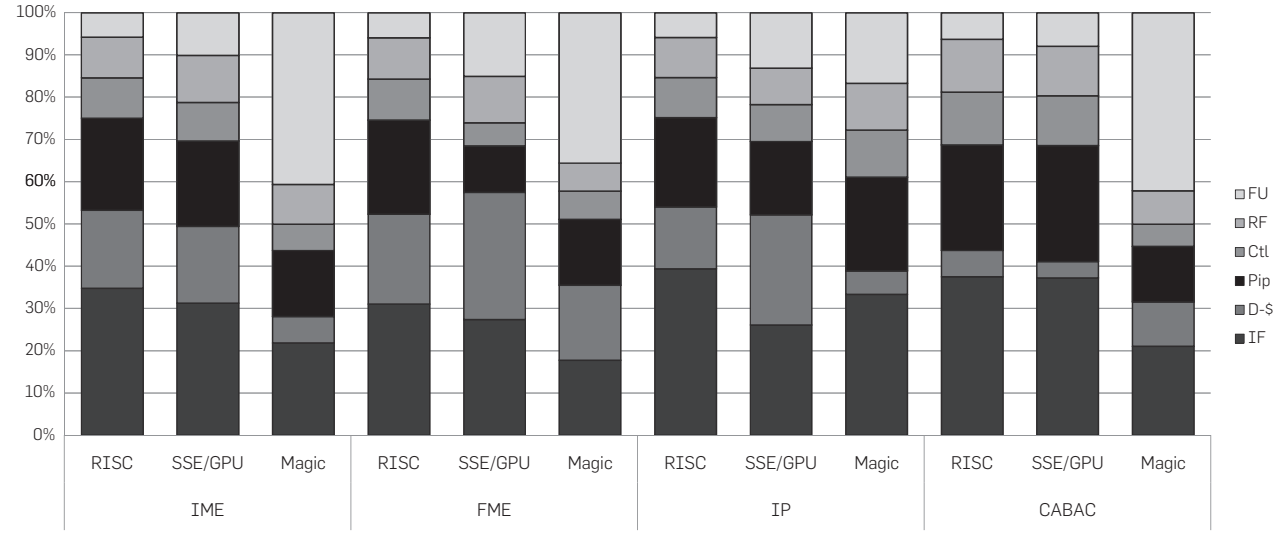


4.2. Algorithm specific instructions

The root cause of the large energy difference is that the basic operations in H.264 are very simple and low energy. They only require 8–16 bit integer operations, so the fundamental energy per operation bound is on the order of hundreds of femtojoules in a 90 nm process. All other costs in a processor—IF, register fetch, data fetch, control, and pipeline registers—are much larger (140 pJ) and dominate overall power. Standard SIMD and simple fused instructions can only go so far to improve the performance and energy efficiency. It is hard to aggregate more than 10–20 operations into an instruction without incurring growing inefficiencies, and with tens of operations per cycle we still have a machine where around 90% of the energy is going into overhead functions. It is now easy to see how an ASIC can be 2–3 orders of magnitude lower energy than a processor. For computationally limited applications with low-energy operations, an ASIC can implement hardware which both has low overheads, and is a perfect structural match to the application. These features allow it to exploit large amounts of parallelism efficiently.

To match these results in a processor we must amortize the per-instruction energy overheads over hundreds of these

Figure 7. Processor energy breakdown for H.264. IF is instruction fetch/decode. D-$ is data cache. Pip is the pipeline registers, buses, and clocking. Ctl is random control. RF is the register file. FU is the functional elements. Only the top bar or two (FU, RF) contribute useful work in the processor. For this application it is hard to achieve much more than 10% of the power in the FU without adding custom hardware units.

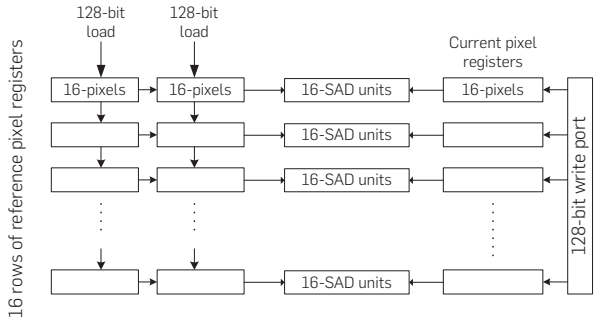


simple operations. To create instructions with this level of parallelism requires custom storage structures with algorithm-specific communication links to directly feed large amounts of data to custom functional units without explicit register accesses. These structures also substantially increase data-reuse in the datapath and reduce communication bandwidth and power at all levels of the memory hierarchy (register, cache, and memory).

Once this hardware is in place, the machine can issue "magic" instructions that accomplish large amounts of computation at very low cost. This type of structure eliminates almost all the processor overheads for these functions by eliminating most of the communication overhead associated with processors. We call these instructions "magic" because they can have a large effect on both the energy and performance of an application and yet they would be difficult to derive directly from the code. Such instructions typically require an understanding of the underlying algorithms, as well as the capabilities and limitations of existing hardware resources, thus requiring greater effort on the part of the designer. Since the IP stage uses techniques similar to FME, the rest of the section will focus on IME, FME, and CABAC.

IME Strategy: To demonstrate the nature and benefit of magic instructions we first look at IME, which determines the best alignment for two image blocks. The best match is defined by the smallest sum-of-absolute-differences (SAD) of all of the pixel values. Since finding the best match requires scanning one image block over a larger piece of the image, one can easily see that while this requires a large number of calculations, it also has very high data locality. Figure 8 shows the custom datapath elements added to the IME processor to accelerate this function. At the core is a 16×16 SAD array, which can perform 256 SAD operations in 1 cycle. Since our standard vector register files cannot fed enough data to this unit per cycle, the SAD unit is fed by a custom register structure, which allows parallel access to all 16-pixel rows and enables this datapath to perform one 256-pixel computation per cycle. In addition, the intermediate results of the pixel operations need not be stored since they can be reduced in place (summed) to create the single desired output. Furthermore, because we need to check many possible alignments, the custom storage structure has support for parallel shifts in all four directions, thus allowing one to shift the entire comparison image in only one cycle. This feature drastically reduces the instructions wasted on loads, shifts, and pointer arithmetic operations as well as data cache accesses. "Magic" instructions and storage elements are also created for other major algorithmic functions in IME to achieve similar gains.

Figure 8. Custom storage and compute for IME 4 × 4 SAD. Current and ref-pixel register files feed all pixels to the 16 × 16 SAD array in parallel. Also, the ref-pixel register file allows horizontal and vertical shifts.



Thus, by reducing instruction overheads and by amortizing the remaining overheads over larger datapath widths, this functional unit finally consumes around 40% of the total instruction energy. The performance and energy efficiency improve by 200–300× over the base implementation, match the ASIC's performance and come within 3× of ASIC energy. This customized solution is 20–30× better than the results using only generic data-parallel techniques.

FME Strategy: FME improves the output of the IME stage by refining the alignment to a fraction of a pixel. To perform the fractional alignment, the FME stage interpolates one image to estimate the values of a 4×4 pixel block at fractional pixel coordinates. This operation is done by a filter and upsample block, which again has high arithmetic intensity and high data locality. In H.264, upsampling uses a six tap FIR filter that requires one new pixel per iteration. To reduce IFs and register file transfers, we augment the processor register file with a custom 8 bit wide, 6 entry shift register structure which works like a FIFO: every time a new 8 bit value is loaded, all elements are shifted. This eliminates the use of expensive register file accesses for either data shifting or operand fetch, which are now both handled by short local wires. All six entries can now be accessed in parallel and we create a six input multiplier/adder which can do the calculation in a single cycle and also can be implemented much more efficiently than the composition of normal 2-input adders. Finally, since we need to perform the upsampling in 2-D, we build a shift register structure that stores the horizontally upsampled data, and feeds its outputs to a number of vertical upsampling units (Figure 9).

This transformation yields large savings even beyond the savings in IF energy. From a pure datapath perspective (register file, pipeline registers, and functional units), this approach dissipates less than 1/30th the energy of a traditional approach.

A look at the FME SIMD code implementation highlights the advantages of this custom hardware approach versus the use of larger SIMD arrays. The SIMD implementation suffers from code replication and excessive local memory and register file accesses, in addition to not having the most efficient functional units. FME contains seven different sub-block sizes ranging from 16×16 pixel blocks to 4×4 blocks, and not all of them can fully exploit the 18-way SIMD datapath. Additionally, to use the 18-way SIMD datapath, each sub-block requires a slightly different code sequence, which results in code replication and more I-fetch power because of the larger I-cache.

To avoid these issues, the custom hardware upsampler processes 4×4 pixels. This allows it to reuse the same computation loop repeatedly without any code replication,

Figure 9. FME upsampling unit. Customized shift registers, directly wired to function logic, result in efficient upsampling. Ten integer pixels from local memory are used for row upsampling in RFIR blocks. Half upsampled pixels along with appropriate integer pixels are loaded into shift registers. CFIR accesses six shift registers in each column simultaneously to perform column upsampling.

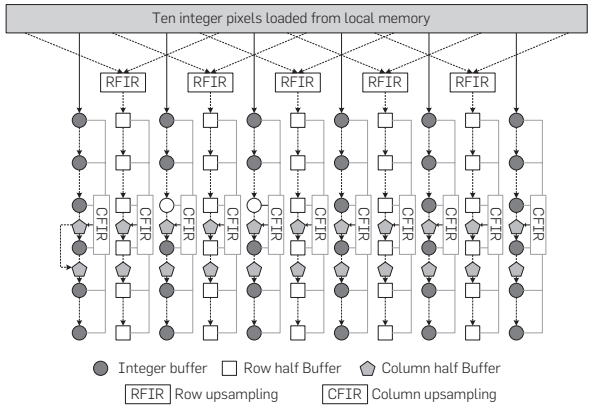
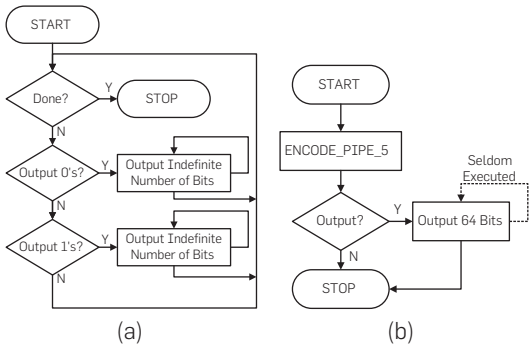


Figure 10. CABAC Arithmetic Encoding Loop (a) H.264 reference code. (b) After insertion of "magic" instructions. Much of the control logic in the main loop has been reduced to one constant time instruction ENCODE_PIPE_5.



which, in turn, lets us reduce the I-cache from a 16KB 4-way cache to a 2KB direct-mapped cache. Due to the abundance of short-lived data, we remove the vector register files and replace them with custom storage buffers. The "magic" instruction reduces the instruction cache energy by 54× and processor fetch and decode energy by 14×. Finally, as Figure 7 shows, 35% of the energy is now going into the functional units, and again the energy efficiency of this unit is close to an ASIC.

CABAC Strategy: CABAC originally consumed less than 2% of the total energy, but after data-parallel components are accelerated by "magic" instructions, CABAC dominates the total energy. However, it requires a different set of optimizations because it is control oriented and not data parallel. Thus, for CABAC, we are more interested in control fusion than operation fusion.

A critical part of CABAC is the arithmetic encoding stage, which is a serial process with small amounts of computation, but complex control flow. We break arithmetic coding down into a simple pipeline and drastically change it from the reference code implementation, reducing the binary encoding of each symbol to five instructions. While there are several if–then–else conditionals reduced to single instructions (or with several compressed into one), the most significant reduction came in the encoding loop, as shown in Figure 10a. Each iteration of this loop may or may not trigger execution of an internal loop that outputs an indefinite number of encoded bits. By fundamentally changing the algorithm, the while loop was reduced to a single constant time instruction (ENCODE_PIPE_5) and a rarely executed while loop, as shown in Figure 10b.

The other critical part of CABAC is the conversion of non-binary-valued DCT coefficients to binary codes in the binarization stage. To improve the efficiency of this step, we create a 16-entry LIFO structure to store DCT coefficients. To each LIFO entry, we add a single-bit flag to identify zero-valued DCT coefficients. These structures, along with their corresponding logic, reduce register file energy by bringing the most frequently used values out of the register file and into custom storage buffers. Using "magic" instructions we produce Unary and Exponential-Golomb codes using simple operations, which help reduce datapath energy. These modifications are inspired by the ASIC implementation described in Shojania and Sudharsanan.[17] CABAC is optimized to achieve the bit rate required for H.264 level 3.1 at 720p video resolution.

Magic Instructions Summary: To summarize, the magic instructions perform up to hundreds of operations each time they are executed, so the overhead of the instruction is better balanced by the work performed. Of course this is hard to do in a general way, since bandwidth requirements and utilization of a larger SIMD array would be problematic. Therefore we solved this problem by building custom storage units tailored to the application, and then directly connecting the necessary functional units to these storage units. These custom storage units greatly amplified the register fetch bandwidth, since data in the storage units is used for many different computations. In addition, since the intra-storage and functional unit communications were fixed and local, they can be managed at ASIC-like energy costs.

After this effort, the processors optimized for data-parallel algorithms have a total speedup of up to 600× and an energy reduction of 60–350× compared to our base CMP. For CABAC total performance gain is 17× and energy gain is 8×. Figure 7 provides the final energy breakdowns. The efficiencies found in these custom datapaths are impressive, since, in H.264 at least, they take advantage of data sharing patterns and create very efficient multiple-input operations. This means that even if researchers are able to a create a processor which decreases the instruction and data fetch parts of a processor by more than 10×, these solutions will not be as efficient as solutions with "magic" instructions.

Achieving ASIC-like efficiency required 2–3 special hardware units for each subalgorithm, which is significant customization work. Some might even say we are just building an ASIC in our processor. While we agree that creating "magic" instructions requires a thorough understanding of

the application as well as hardware, we feel that adding this hardware in an extensible processor framework has many advantages over just designing an ASIC. These advantages come from the constrained processor design environment and the software, compiler, and debugging tools available in this environment. Many of the low-level issues, like interface design and pipelining, are automatically handled. In addition, since all hardware is wrapped in a general-purpose processor, the application developer retains enough flexibility in the processor to make future algorithmic modifications.

5. ENERGY-EFFICIENT COMPUTERS

It is important to remember that the "overhead" of using a processor depends on the energy required for the desired operation. Floating point (FP) energy costs are about 10× the small integer ops we have explored in this paper, so machines with 10 wide FP units will not be far from the maximum efficiency possible for that class of applications. Similarly, customizing the hardware will not have a large impact on the energy efficiency of an application dominated by memory costs; an ASIC and a processor's energy will not be that different. For these applications, optimization that restructures the algorithm and/or the memory system is needed to reduce energy, and can yield large savings.[23]

Unfortunately, as we drive to more energy-efficient solutions, we will find ways to transform FP code to fixed point operations, and restructure our algorithms to minimize the memory fetch costs. Said differently, if we want ASIC-like energy efficiencies—100× to 1000× more energy efficient than general-purpose CPUs—we will have to transform our algorithms to be dominated by the simple, low-energy operations we have been studying in this paper. Since the energy of these operations is very low, any overhead, from the register fetch to the pipeline registers in a processor, is likely to dominate energy costs. The good news is that this large overhead per instruction makes estimating the energy savings easy—you simply look at the performance gains—but the bad news is that adding state-of-the art data-parallel hardware like wide SIMD units and media extensions will still leave you far from the desired efficiency.

It is encouraging that we were able to achieve ASIC energy levels in a customized processor by creating customized hardware that easily fit inside a processor framework. Extending a processor instead of building an ASIC seems like the correct approach, since it provides a number of software development advantages and the energy cost of this option seems small. However, building such custom datapaths still requires a significant effort and thus the key challenge now is to build a design system that lets application designers create and exploit such customizations with much greater ease. The key is to find a parameterization of the space which makes sense to application designers in a specific application domain.

For example, often a number of algorithms in a domain share similar data flow and computation structures. In H.264 a common computational motif is based on a convolution-like data flow: apply a function to all the data, then perform a reduction, then shift the data and add a small amount of new data, and repeat. A similar pattern of convolution-like computations also exists in a number of other image processing and media processing algorithms. While the exact computation is going to be different for each particular algorithm, we believe that by exploiting the common data-flow structure of these algorithms we can create a generalized convolution abstraction which application designers can customize. If this abstraction is useful for application designers, one can imagine implementing it by creating a flexible hardware unit that is significantly more efficient than a generic SIMD/SSE unit. We also believe that similar patterns exist in other domains that may allow us to create a set of customized units for each domain.

Even if we could come up with such a set of customized functional units, it is likely that some degree of per algorithm configurability will be required. For example, in a convolution engine, the convolution size and resulting datapath size could vary from algorithm to algorithm and thus potentially needs to be tuned on a per processor basis. This leads to the idea of creating a two-step design process. The first step is when a set of chip experts design a processor generator platform. This is a meta-level design which "knows" about the special functional units and control optimization, and provides the application designer an application-tailored interface. The application designers can then co-optimize their code and the interface parameters to meet their requirements. After this co-optimization, an optimized implementation based on these parameters is automatically generated. In fact, such a platform will also help in building the more generic domain customized functional units mentioned earlier by facilitating the process of rapidly creating and evaluating new designs.

A reconfigurable processor generator alone is not a sufficient solution, since one still needs to take one or more of these processors and create a working chip system. Designing and validating a chip is an extremely hard and expensive task. If application customization will be needed for efficiency—and our data indicates it will be—we need to start creating systems that will efficiently allow savvy application experts to create these optimized chip level solutions. This will require extending the ideas for extensible processors to full chip generation systems. We are currently working on creating this type of system.[16]

Acknowledgments

This work would have not been possible without great support and cooperation from many people at Tensilica including Chris Rowen, Dror Maydan, Bill Huffman, Nenad Nedeljkovic, David Heine, Govind Kamat, and others. The authors acknowledge the support of the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation subsidiary, and earlier support from DARPA. This material is based upon work partially supported under a Sequoia Capital Stanford Graduate Fellowship. The National Science Foundation under Grant #0937060 to the Computing Research Association also supports this material for the CIFellows Project. Any opinions, findings,

and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the National Science Foundation or the Computing Research Association.

References

1. Balfour, J., Dally, W., Black-Schaffer, D., Parikh, V., Park, J. An energy-efficient processor architecture for embedded systems. *Comput. Archit. Lett. 7*,1 (2007), 29–32.
2. Chen, T.C. Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder. *IEEE Trans. Circuits Syst. Video Technol. 16*, 6 (2006), 673–688.
3. Chen, Y.-K., Li, E.Q., Zhou, X., Ge, S. Implementation of H.264 encoder and decoder on personal computers. *J Vis. Commun. Image Represent. 17* (2006), 509–532.
4. Clark, N., Zhong, H., Mahlke, S. Automated custom instruction generation for domain-specific processor acceleration. *IEEE Trans. Comput. 54*, 10 (2005), 1258–1270.
5. Cong, J., Fan, Y., Han, G., Zhang, Z. Application-specific instruction generation for configurable processor architectures. In *12th International Symposium on Field Programmable Gate Arrays* (2004), 183–189.
6. Davis, W., Zhang, N., Camera, K., Chen, F., Markovic, D., Chan, N., Nikolic, B., Brodersen, R. A design environment for high throughput, low power, dedicated signal processing systems. In *Custom Integrated Circuits Conference* (*CICC*) (2001).
7. Dennard, R., Gaensslen, F., Yu, H., Rideout, V., Bassous, E., LeBlanc, A. Design of ion-implanted MOSFET's with very small physical dimensions. *Proc. IEEE (reprinted from IEEE J Solid-State Circuits, 1974)*, *87*, 4 (1999), 668–678.
8. Esmaeilzadeh, H., Blem, E., Amant, R.S., Sankaralingam, K., Burger, D. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th International Symposium on Computer Architecture* (June 2011).
9. Horowitz, M. Scaling, power and the future of CMOS. In *Proceedings of the 20th International Conference on VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems* (2007), 23–23.
10. Intel Corporation. Motion Estimation with Intel Streaming SIMD Extensions 4 (Intel SSE4) (2008).
11. ITU-T. Joint Video Team Reference Software JM8.6 (2004).
12. Iverson, V., McVeigh, J., Reese, B. Real-time H.264/AVC Codec on Intel architectures. In *IEEE International Conference on Image Processing ICIP'04* (2004).
13. Kathail, V. Creating power-efficient application engines for SOC design. *SOC Central* (2005).
14. MPEG, I., VCEG, I.-T. Fast integer pel and fractional pel motion estimation for JVT. *JVT-F017* (2002).
15. Rowen, C., Leibson, S. Flexible architectures for engineering successful SOCs. In *Proceedings of the 41st Annual Design Automation Conference* (2004), 692–697.
16. Shacham, O., Azizi, O., Wachs, M., Qadeer, W., Asgar, Z., Kelley, K., Stevenson, J., Solomatnikov, A., Firoozshahian, A., Lee, B., Richardson, S., Horowitz, M. Why design must change: Rethinking digital design. *IEEE Micro 30*, 6 (Nov.–Dec. 2010), 9–24.
17. Shojania, H., Sudharsanan, S. A VLSI architecture for high performance CABAC encoding. In *Visual Communications and Image Processing* (2005).
18. Tensilica Inc. Implementing the advanced encryption standard on Xtensa processors. *Application Notes* (2009).
19. Tensilica Inc. Implementing the fast Fourier transform (FFT). *Application Notes* (2005).
20. Tensilica Inc. The what, why, and how of configurable processors (2008).
21. Tensilica Inc. Xtensa LX2 benchmarks (2005).
22. Tensilica Inc. Xtensa processor extensions for data encryption standard (DES). *Application Notes* (2008).
23. Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., Demmel, J. Optimization of sparse matrix-vector multiplication on emerging multicore platforms. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing* (2007).
24. Woh, M., Seo, S., Mahlke, S., Mudge, T., Chakrabarti, C., Flautner, K. AnySP: Anytime anywhere anyway signal processing. *SIGARCH Comp. Arch. News 37*, 3 (2009), 128–139.
25. Yin, P., Tourapis, H.-Y.C., Tourapis, A.M., Boyce, J. Fast mode decision and motion estimation for JVT/H.264. In *Proceedings of IEEE International Conference on Image Processing* (2003).

Rehan Hameed (rhameed@stanford.edu), Stanford University, Stanford, CA.

Wajahat Qadeer (wqadeer@stanford.edu), Stanford University. Stanford, CA.

Megan Wachs (wachs@stanford.edu), Stanford University, Stanford, CA.

Omid Azizi (oazizi@gmail.com), Hicamp Systems, Menlo Park, CA.

Alex Solomatnikov (Solomatnikov@gmail.com), Hicamp Systems, Menlo Park, CA.

Benjamin C. Lee (benjamin.c.lee@duke.edu), Duke University, Durham, NC.

Stephen Richardson (steveri@stanford.edu), Stanford University, Stanford, CA.

Christos Kozyrakis (kozyraki@stanford.edu), Stanford University, Stanford, CA.

Mark Horowitz (horowitz@stanford.edu), Stanford University, Stanford, CA.

DOI:10.1145/2001269.2001294

Technical Perspective
A Better Way to Learn Features

By Geoffrey E. Hinton

A TYPICAL MACHINE learning program uses weighted combinations of features to discriminate between classes or to predict real-valued outcomes. The art of machine learning is in constructing the features, and a radically new method of creating features constitutes a major advance.

In the 1980s, the new method was backpropagation, which uses the chain rule to backpropagate error derivatives through a multilayer, feed-forward, neural network and adjusts the weights between layers by following the gradient of the backpropagated error. This worked well for recognizing simple shapes, such as handwritten digits, especially in convolutional neural networks that use local feature detectors replicated across the image.[5] For many tasks, however, it proved extremely difficult to optimize deep neural nets with many layers of non-linear features, and a huge number of labeled training cases was required for large neural networks to generalize well to test data.

In the 1990s, Support Vector Machines (SVMs)[8] introduced a very different way of creating features: the user defines a kernel function that computes the similarity between two input vectors, then a judiciously chosen subset of the training examples is used to create "landmark" features that measure how similar a test case is to each training case. SVMs have a clever way of choosing which training cases to use as landmarks and deciding how to weight them. They work remarkably well on many machine learning tasks even though the selected features are non-adaptive.

The success of SVMs dampened the earlier enthusiasm for neural networks. More recently, however, it has been shown that multiple layers of feature detectors can be learned greedily, one layer at a time, by using unsupervised learning that does not require labeled data. The features in each layer are designed to model the statistical structure of the patterns of feature activations in the previous layer. After learning several layers of features this way without paying any attention to the final goal, many of the high-level features will be irrelevant for any particular task, but others will be highly relevant because high-order correlations are the signature of the data's true underlying causes and the labels are more directly related to these causes than to the raw inputs. A subsequent stage of fine-tuning using backpropagation then yields neural networks that work much better than those trained by backpropagation alone and better than SVMs for important tasks such as object or speech recognition.[1,2,4] The neural networks outperform SVMs because the limited amount of information in the labels is not being used to create multiple features from scratch; it is only being used to adjust the class boundaries by slightly modifying the features.

The following paper by Lee et al. is the first impressive demonstration that greedy layer-by-layer feature creation can be applied to large images. To make this work, they had to use replicated local feature detectors, and they had to solve a tricky technical problem in probabilistic modeling of convolutional neural networks. These networks summarize the outputs of nearby copies of the same feature detector by simply reporting their maximum value. For unsupervised learning to work properly, this operation must be given a sensible probabilistic interpretation. The authors solve this problem by using a soft, probabilistic version of the maximum function, and they show that this allows them to learn an impressive feature hierarchy in which the first layer represents oriented edge filters, the second layer represents object parts, and the third represents larger parts or whole objects. They also show their model can combine bottom-up and top-down inference, using more global context to select appropriately between local features.

The learning algorithm used by the authors is designed to produce a composite generative model called a "deep belief net,"[3] but they perform top-down inference as if it were a different generative model called a "deep Boltzmann machine." They achieve quite good results at image completion, and even better results might be obtained if they fine-tuned their generative model as a deep Boltzmann machine using a recent algorithm developed by Salakhutdinov.[7]

Machine learning still has some way to go before it can efficiently create the complicated features like SIFT[6] used in many leading systems for computer vision. However, this paper should seriously worry those computer vision researchers who still believe that hand-engineered features have a long-term future. Further improvements from unsupervised learning also seem likely: biology tells us that applying high-resolution filters across an entire image is not the best way to use a neural net, even if it has billions of neurons. C

References

1. Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems*. B. Schoelkopf, J. Platt, and T. Hoffman, Eds. MIT Press, Cambridge, MA, 2007, 19.
2. Dahl, G., Mohamed, A. and Hinton, G.E. Acoustic modeling using deep belief networks. *IEEE Trans. on Audio, Speech, and Language Processing 19*, 8 (2011).
3. Hinton, G.E., Osindero, S. and The, Y.T. A fast learning algorithm for deep belief nets. *Neural Computation 18* (2006).
4. Hinton, G.E and Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science 313* (2006), 504–507.
5. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE 86*, 11 (1998), 2278–2324.
6. Lowe, D.G. Object recognition from local scale-invariant features. In *Proc. International Conference on Computer Vision*, 1999.
7. Salakhutdinov, R.S. Learning Deep Generative Models. PhD thesis, University of Toronto, 2009.
8. Vapnik, V.N. *The Nature of Statistical Learning Theory*. Springer, New York, NY, 2000.

Geoffrey E. Hinton (hinton@cs.toronto.edu) is a professor of computer science at the University of Toronto, Canada.

DOI:10.1145/2001269.2001295

Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks

By Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng

Abstract

There has been much interest in unsupervised learning of hierarchical generative models such as deep belief networks (DBNs); however, scaling such models to full-sized, high-dimensional images remains a difficult problem. To address this problem, we present the *convolutional deep belief network*, a hierarchical generative model that scales to realistic image sizes. This model is translation-invariant and supports efficient bottom-up and top-down probabilistic inference. Key to our approach is *probabilistic max-pooling*, a novel technique that shrinks the representations of higher layers in a probabilistically sound way. Our experiments show that the algorithm learns useful high-level visual features, such as object parts, from unlabeled images of objects and natural scenes. We demonstrate excellent performance on several visual recognition tasks and show that our model can perform hierarchical (bottom-up and top-down) inference over full-sized images.

1. INTRODUCTION

Machine learning has been highly successful in tackling many real-world artificial intelligence and data mining problems, such as optical character recognition, face detection, autonomous car driving, data mining of biological data, and Web search/information retrieval. However, the success of machine learning systems often requires a large amount of labeled data (which is expensive to obtain) and significant manual feature engineering. These feature representations are often hand-designed, require significant amounts of domain knowledge and human labor, and do not generalize well to new domains. Therefore, it is desirable to be able to develop feature representations automatically while using a small amount of labeled data.

Given these issues, we consider the problem of learning feature representations from unlabeled data, which we call *unsupervised feature learning*. Here, we are interested in primarily using unlabeled data because we can easily obtain a virtually unlimited amount of unlabeled data via the Internet. In fact, even though we do not have labels, there often exist rich structures in unlabeled data. For example, if we look at images of a specific object (e.g., a face), we can easily discover high-level structures such as object parts (e.g., face parts). Given natural images, we may be able to discover low-level structures such as edges, as well as high-level structures such as corners, local curvatures, and shapes. The main assumption of unsupervised feature learning is that such structures in unlabeled data can be useful in machine learning tasks. For example, if the input data have structures generated from specific object classes (e.g., cars vs. faces), then discovering class-specific patterns (e.g., car wheels or face parts) will be useful for classification, possibly combined with a small amount of labeled data. Similarly, even simple image features (e.g., edges or corners) learned from unlabeled natural images can be useful for object recognition tasks that deal with completely unrelated images. In this context, how can we discover such useful high-level features from unlabeled data?

In recent years, "deep learning" approaches have gained significant interest as a way of building hierarchical representations from unlabeled data.[2, 10, 15, 26, 28] Deep architectures attempt to learn hierarchical structures and seem promising in learning simple concepts first and then successfully building up more complex concepts by composing the simpler ones together. Specifically, deep architectures consist of feature detector units arranged in layers. Lower layers detect simple features and feed into higher layers, which in turn detect more complex features. In particular, the DBN[10] is a multilayer generative model where each layer encodes statistical dependencies among the units in the layer below, and it can be trained to (approximately) maximize the likelihood of its training data. DBNs have been successfully used to learn high-level structures in a wide variety of domains, including handwritten digits[10] and human motion capture data.[31] We build upon the DBN in this paper because we are interested in learning a generative model of images that can be trained in a purely unsupervised manner.

While DBNs have been successful in controlled domains, scaling them to realistic-sized (e.g., 200 × 200 pixel) images remains challenging for two reasons. First, images are high-dimensional, so the algorithms must scale gracefully and be

A previous version of this paper appeared in *Proceedings of the 26th International Conference on Machine Learning* (Montreal, Canada, 2009).

computationally tractable even when applied to large images. Second, objects can appear at arbitrary locations in images; thus, it is desirable that representations be invariant at least to local translations of the input. We address these issues by incorporating translation invariance. Like LeCun et al.[17] and Grosse et al.,[7] our algorithm learns feature detectors shared among all locations in an image because a feature detector that captures useful information in one part of an image can pick up the same information elsewhere. Thus, our model can represent large images using a small number of feature detectors.

This paper presents the *convolutional deep belief network*, a hierarchical generative model that scales to full-sized images. We also present *probabilistic max-pooling*, a novel technique that allows higher-layer units to cover larger areas of the input in a probabilistically sound way. To the best of our knowledge, ours is the first unsupervised, translation-invariant deep learning model that scales to realistic image sizes and supports full probabilistic inference. The first, second, and third layers of our network learn edge detectors, object parts, and objects, respectively. We show that these representations achieve excellent performance on several visual recognition tasks and allow hidden object parts to be inferred from high-level object information.

2. PRELIMINARIES

2.1. Restricted Boltzmann machines

In this section, we briefly review the restricted Boltzmann machine (RBM) and DBN models.

The RBM is a two-layer, bipartite, undirected graphical model[a] with a set of binary hidden random variables (units) $\mathbf{h}$ of dimension K, a set of (binary or real-valued) visible random variables (units) $\mathbf{v}$ of dimension D, and symmetric connections between these two layers represented by a weight matrix $W \in \mathbb{R}^{D \times K}$. (See Figure 1 for an illustration of the RBM.) Intuitively, the RBM can be viewed as a Markov Random Field that tries to represent the input data (visible units) with latent factors (hidden units). Here, the weights encode a statistical relationship between the hidden nodes and visible nodes. For example, the weights between the jth hidden node (h_j) and all visible nodes are denoted as jth "basis" vector, and h_j are assigned to 1 with high probability whenever the input data match the jth basis vector (see Equation 4). The formal probabilistic semantics for an RBM is defined by its energy function as follows:

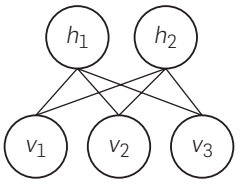
$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z}\exp(-E(\mathbf{v}, \mathbf{h})), \tag{1}$$

where Z is a normalization constant. If the visible units are binary valued, the energy function can be defined as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{D}\sum_{j=1}^{K} v_i W_{ij} h_j - \sum_{j=1}^{K} b_j h_j - \sum_{i=1}^{D} c_i v_i, \tag{2}$$

[a] See Koller and Friedman[14] for background on undirected graphical models. In short, the undirected graphical models denote probabilistic models whose joint probability can be written as the product of non-negative potential functions, as in Equations 1–3.

Figure 1. An example RBM with three visible units ($D = 3$) and two hidden units ($K = 2$). See text for details.



where b_j are hidden unit biases ($\mathbf{b} \in \mathbb{R}^K$) and c_i are visible unit biases ($\mathbf{c} \in \mathbb{R}^D$). If the visible units are real-valued, we can define the energy function by adding a quadratic term to make the distribution well defined:

$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2}\sum_{i=1}^{D} v_i^2 - \sum_{i=1}^{D}\sum_{j=1}^{K} v_i W_{ij} h_j - \sum_{j=1}^{K} b_j h_j - \sum_{i=1}^{D} c_i v_i. \tag{3}$$

The above energy function defines a joint probability distribution and conditional probability distribution. From the energy function, it is clear that the hidden units are conditionally independent of one another given the visible layer, and vice versa. In particular, the units of a binary hidden layer (conditioned on the visible layer) are independent Bernoulli random variables as follows:

$$P(h_j = 1 \mid \mathbf{v}) = \sigma\left(\sum_i W_{ij} v_i + b_j\right), \tag{4}$$

where $\sigma(s) = \frac{1}{1+\exp(-s)}$ is the sigmoid function. Similarly, if the visible layer is binary-valued, the visible units (conditioned on the hidden layer) are independent Bernoulli random variables as follows:

$$P(v_i = 1 \mid \mathbf{h}) = \sigma\left(\sum_j W_{ij} h_i + c_i\right). \tag{5}$$

If the visible layer is real-valued, the visible units (conditioned on the hidden layer) are independent Gaussians with diagonal covariance as follows:

$$P(v_i \mid \mathbf{h}) = \mathcal{N}\left(\sum_j W_{ij} h_j + c_i, 1\right), \tag{6}$$

where $\mathcal{N}(\cdot,\cdot)$ is a Gaussian distribution. Therefore, we can perform efficient block Gibbs sampling by alternately sampling each layer's units (in parallel) given the other layer. We will often refer to a unit's expected value as its *activation*.

The RBM is a generative model, so, in principle, its parameters can be optimized by performing stochastic gradient descent on the log-likelihood of training data. Unfortunately, computing the exact gradient of the log-likelihood is intractable. Instead, one typically uses the contrastive divergence approximation,[8] which has been shown to work well in practice.

2.2. Deep belief networks

The RBM by itself is limited in what it can represent. Its real power emerges when RBMs are stacked to form a DBN, a generative model consisting of many layers. In a DBN,

each layer comprises a set of binary or real-valued units. Two adjacent layers have a full set of connections between them, but no two units in the same layer are connected. Hinton et al.[10] proposed an efficient algorithm for training DBNs, by greedily training each layer (from lowest to highest) as an RBM using the previous layer's activations as inputs.

For example, once a layer of the network is trained, the parameters W_{ij}, b_j, c_i's are frozen and the hidden unit values (given the data) are inferred. These inferred values serve as the input data used to train the next higher layer in the network. Hinton et al.[10] showed that by repeatedly applying such a procedure, one can learn a multilayered DBN. In some cases, this iterative greedy algorithm can be shown to be optimizing a variational lower-bound on the data likelihood, if each layer has at least as many units as the layer below. This greedy layer-wise training approach has been shown to provide a good initialization for parameters for the multilayered network.

3. ALGORITHM

Both RBMs and DBNs ignore the 2D structure of images, so weights that detect a given feature must be learned separately for each location. This redundancy makes it difficult to scale these models to full images. One possible way of scaling up is to use massive parallel computation, such as using GPUs, as shown in Raina et al.[25] However, this method may still suffer from having a huge number of parameters. In this section, we present a new method that scales up DBNs using weight-sharing. Specifically, we introduce our model, the convolutional DBN (CDBN), where weights are shared among all locations in an image. This model scales well because inference can be done efficiently using convolution.

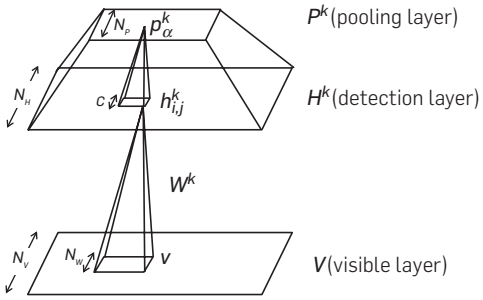
3.1. Notation

For notational convenience, we will make several simplifying assumptions. First, we assume that all inputs to the algorithm are $N_V \times N_V$ images, even though there is no requirement that the inputs be square, equally sized, or even 2D. We also assume that all units are binary-valued, while noting that it is straightforward to extend the formulation to the real-valued visible units (see Section 2.1). We use $*$ to denote convolution,[b] and $\bullet$ to denote an element-wise product followed by summation, i.e., $A \bullet B = \mathrm{tr}\, A^T B$. We place a tilde above an array ($\tilde{A}$) to denote flipping the array horizontally and vertically.

3.2. Convolutional RBM

First, we introduce the convolutional RBM (CRBM). Intuitively, the CRBM is similar to the RBM, but the weights between the hidden and visible layers are shared among all locations in an image. The basic CRBM consists of two layers: an input layer V and a hidden layer H (corresponding to the lower two layers in Figure 2). The input layer consists of an $N_V \times N_V$ array of binary units. The hidden layer consists of K groups, where each group is an $N_H \times N_H$ array of binary units, resulting in $N_H^2 K$ hidden units. Each of the K groups is associated with a $N_W \times N_W$ filter ($N_W \triangleq N_V - N_H + 1$); the filter weights are shared across all the hidden units within the group. In addition, each hidden group has a bias b_k and all visible units share a single bias c.

Figure 2. Convolutional RBM with probabilistic max-pooling. For simplicity, only group k of the detection layer and the pooling layer are shown. The basic CRBM corresponds to a simplified structure with only visible layer and detection (hidden) layer. See text for details.



We define the energy function $E(\mathbf{v}, \mathbf{h})$ as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{k=1}^{K} \sum_{i,j=1}^{N_H} \sum_{r,s=1}^{N_W} h_{ij}^k W_{rs}^k v_{i+r-1,j+s-1} - \sum_{k=1}^{K} b_k \sum_{i,j=1}^{N_H} h_{ij}^k - c \sum_{i,j=1}^{N_V} v_{ij}. \quad (7)$$

Using the operators defined previously,

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{k=1}^{K} h^k \bullet \left(\tilde{W}^k * v\right) - \sum_{k=1}^{K} b_k \sum_{i,j} h_{i,j}^k - c \sum_{i,j} v_{ij}. \quad (8)$$

As with standard RBMs (Section 2.1), we can perform block Gibbs sampling using the following conditional distributions:

$$P(h_{ij}^k = 1 \,|\, \mathbf{v}) = \sigma\left((\tilde{W}^k * v)_{ij} + b_k\right), \quad (9)$$

$$P(v_{ij} = 1 \,|\, \mathbf{h}) = \sigma\left(\left(\sum_k W^k * h^k\right)_{ij} + c\right), \quad (10)$$

where $\sigma(\cdot)$ is the sigmoid function.[c] Gibbs sampling forms the basis of our inference and learning algorithms.

3.3. Probabilistic max-pooling

To learn high-level representations, we stack CRBMs into a multilayer architecture analogous to DBNs. This architecture is based on a novel operation that we call *probabilistic max-pooling*.

In general, higher-level feature detectors need information from progressively larger input regions. Existing

[b] The convolution of an $m \times m$ array with an $n \times n$ array ($m > n$) may result in an $(m+n-1) \times (m+n-1)$ array (full convolution) or an $(m-n+1) \times (m-n+1)$ array (valid convolution). Rather than inventing a cumbersome notation to distinguish between these cases, we let it be determined by context.

[c] For the case of real-valued visible units, we can follow the standard formulation as in Section 2.1 and show that

$$P(v_{ij} \,|\, \mathbf{h}) = \mathcal{N}\left(\sum_k (W^k * h^k)_{ij} + c, 1\right). \quad (11)$$

translation-invariant representations (e.g., convolutional networks) often involve two kinds of alternating layers: "detection" layers, where responses are computed by convolving a feature detector with the previous layer, and "pooling" layers, which shrink the representation of the detection layers by a constant factor. More specifically, each unit in a pooling layer computes the maximum activation of the units in a small region of the detection layer. Shrinking the representation with max-pooling allows higher-layer representations to be invariant to small translations of the input and reduces the computational burden.

Max-pooling was intended only for deterministic and feed-forward architectures,[17] and it is difficult to perform probabilistic inference (e.g., computing posterior probabilities) since max-pooling is a deterministic operator. In contrast, we are interested in a *generative* model of images that supports full probabilistic inference. Hence, we designed our generative model so that inference involves max-pooling-like behavior.

To simplify the notation, we consider a model with a visible layer V, a detection layer H, and a pooling layer P, as shown in Figure 2. The detection and pooling layers both have K groups of units, and each group of the pooling layer has $N_P \times N_P$ binary units. For each $k \in \{1, \ldots, K\}$, the pooling layer P^k shrinks the representation of the detection layer H^k by a factor of C along each dimension, where C is a small integer such as 2 or 3. In other words, the detection layer H^k is partitioned into blocks of size $C \times C$, and each block α is connected to exactly one binary unit p_α^k in the pooling layer (i.e., $N_P = N_H/C$). Formally, we define $B_\alpha \triangleq \{(i,j) : h_{ij}$ belongs to the block $\alpha\}$.

The detection units in the block B_α and the pooling unit p_α are connected in a single potential which enforces the following constraints: at most one of the detection units may be on, and the pooling unit is on if and only if a detection unit is on. By adding this constraint, we can efficiently sample from the network without explicitly enumerating all 2^{C^2} configurations, as we show later. With this constraint, we can consider these $C^2 + 1$ units as a single (softmax) random variable which may take on one of $C^2 + 1$ possible values: one value for each of the detection units being on, and one value indicating that all units are off.

We formally define the energy function of this simplified probabilistic max-pooling-CRBM as follows:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_k \sum_{i,j} \left(h_{i,j}^k (\tilde{W}^k * v)_{i,j} + b_k h_{i,j}^k \right) - c \sum_{i,j} v_{i,j}$$
$$\text{subject to} \sum_{(i,j) \in B_\alpha} h_{i,j}^k \leq 1, \quad \forall k, \alpha. \tag{12}$$

We now discuss sampling the detection layer H and the pooling layer P given the visible layer V. Note that hidden units in group k receive the following bottom-up signal from layer V:

$$I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k * v)_{ij}. \tag{13}$$

Now, we sample each block independently as a multinomial function of its inputs. Suppose $h_{i,j}^k$ is a hidden unit contained in block α (i.e., $(i,j) \in B_\alpha$), the increase in energy caused by turning on unit $h_{i,j}^k$ is $-I(h_{i,j}^k)$, and the conditional probability is given by

$$P(h_{i,j}^k = 1 \mid \mathbf{v}) = \frac{\exp(I(h_{i,j}^k))}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k))} \tag{14}$$

$$P(p_\alpha^k = 0 \mid \mathbf{v}) = \frac{1}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k))}. \tag{15}$$

In our implementation, we sample the random variables $\{h_{i,j}^k\}$ and p_α^k in each block α from a multinomial distribution, and this can be done in parallel since the blocks are disjoint (i.e., each hidden unit belongs to only one block). Sampling the visible layer V given the hidden layer H can be performed in the same way as described in Section 3.2 (e.g., Equation 10 or 11).

3.4. Training via sparsity regularization

Our model is overcomplete in that the size of the representation is much larger than the size of the inputs. In fact, since the first hidden layer of the network contains K groups of units, each roughly the size of the image, it is overcomplete roughly by a factor of K. In general, overcomplete models run the risk of learning trivial solutions, such as feature detectors representing single pixels. One common solution is to force the representation to be "sparse," meaning only a tiny fraction of the units should be active in relation to a given stimulus. Following Lee et al.,[18] we regularize the objective function (log-likelihood) to encourage each hidden unit group to have a mean activation close to a small constant. Specifically, we find that the following simple update (followed by contrastive divergence update) works well in practice:

$$\Delta b_k^{sparsity} \propto p - \frac{1}{N_H^2} \sum_{i,j} P(h_{ij}^k = 1 \mid \mathbf{v}), \tag{16}$$

where p is a target sparsity, and each image is treated as a mini-batch. The learning rate for sparsity update is chosen as a value that makes the hidden group's average activation (over the entire training data) close to the target sparsity, while allowing variations depending on specific input images. The overall training algorithm for the convolutional RBM (with probabilistic max-pooling) is described in Algorithm 1.[d]

3.5. Convolutional deep belief network

Finally, we are ready to define the CDBN, our hierarchical generative model for full-sized images. Analogous to DBNs, this architecture consists of several max-pooling-CRBMs stacked on top of one another. The network defines an energy function by summing the energy functions for all the individual pairs of layers. Training is accomplished with the same greedy, layer-wise procedure described in Section 2.2: once a given layer is trained, its weights are frozen, and its activations are used as input for the next layer. There is one technical point about learning the biases for each intermediate hidden layer.

[d] To reduce the variance, we followed Hinton and Salakhutdinov[11] by setting $V^n := \mathbb{E}_{p(V|H^{(n-1)})}[V|H^{(n-1)})]$; also, we used 1-step **CD** ($N_{cd} = 1$).

Algorithm 1 A training algorithm for the convolutional RBM

repeat {over the training data (e.g., a set of training images)}
Set $V^{(0)} := V$ (e.g., set the current image as a mini-batch)
Compute the posterior $Q^{(0)} \triangleq P(H|V^{(0)})$ (Equations 14 and 15).
Sample $H^{(0)}$ from $Q^{(0)}$.
for $n = 1$ to N_{cd} **do**
Sample V^n from $P(V|H^{(n-1)})$ (Equation 10 or 11).[c]
Compute the posterior $Q^{(n)} \triangleq P(H|V^n)$ (Equations 14 and 15).
Sample $H^{(n)}$ from $Q^{(n)}$.
end for
Update weights and biases with contrastive divergence and sparsity regularization:

$$\Delta W^k \propto \frac{1}{N_H^2}(\tilde{Q}^{(0),k} * V^{(0)} - \tilde{Q}^{(n),k} * V^{(n)}) \tag{17}$$

$$\Delta b_k \propto \frac{1}{N_H^2}\sum_{ij}(Q_{ij}^{(0),k} - Q_{ij}^{(n),k}) + \Delta b_k^{sparsity} \tag{18}$$

$$\Delta c \propto \frac{1}{N_v^2}\sum_{ij}(V_{ij}^{(0)} - V_{ij}^{(n)}) \tag{19}$$

until convergence

Specifically, the biases of a given layer are learned twice: once when the layer is treated as the "hidden" layer of the CRBM (using the lower layer as visible units), and once when it is treated as the "visible" layer (using the upper layer as hidden units). We resolved this problem by simply fixing the biases with the learned hidden biases in the former case (i.e., using only the biases learned when treating the given layer as the hidden layer of the CRBM). However, we note that a potentially better solution would be to jointly train all the weights for the entire CDBN, using the greedily trained weights as the initialization (e.g., Hinton et al.[10, 29]).

3.6. Hierarchical probabilistic inference

Once the parameters have all been learned, we compute the network's representation of an image by sampling from the joint distribution over all of the hidden layers conditioned on the input image. To sample from this distribution, we use block Gibbs sampling, where each layer's units are sampled in parallel (see Sections 2.1 and 3.3).

To illustrate the algorithm, we describe a case with one visible layer V, a detection layer H, a pooling layer P, and another, subsequently higher detection layer H'. Suppose H' has K' groups of nodes, and there is a set of shared weights $\Gamma = \{\Gamma^{1,1}, ..., \Gamma^{K,K'}\}$ where $\Gamma^{k,\ell}$ is a weight matrix connecting pooling unit P^k to detection unit H'^{ℓ}. The definition can be extended to deeper networks in a straightforward way.

Note that an energy function for this sub-network consists of two kinds of potentials: unary terms for each of the groups in the detection layers and interaction terms between V and H and between P and H':[e]

$$E(\mathbf{v}, \mathbf{h}, \mathbf{p}, \mathbf{h}') = -\sum_k v \bullet (W^k * h^k) - \sum_k b_k \sum_{ij} h_{ij}^k - \sum_{k,l} p^k \bullet (\Gamma^{k\ell} * h'^{\ell}) - \sum_l b'_l \sum_{ij} h'^{\ell}_{ij} \tag{20}$$

To sample the detection layer H and pooling layer P, note that the detection layer H^k receives the following bottom-up signal from layer V:

$$I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k * v)_{ij}, \tag{21}$$

and the pooling layer P^k receives the following top-down signal from layer H':

$$I(p_\alpha^k) \triangleq \sum_l (\Gamma^{kl} * h'^{\ell})\alpha. \tag{22}$$

Now, we sample each of the blocks independently as a multinomial function of their inputs, as in Section 3.3. If $(i,j) \in B_\alpha$, the conditional probability is given by

$$P(h_{i,j}^k = 1 | \mathbf{v}, \mathbf{h}') = \frac{\exp(I(h_{i,j}^k) + I(p_\alpha^k))}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k) + I(p_\alpha^k))} \tag{23}$$

$$P(p_\alpha^k = 0 | \mathbf{v}, \mathbf{h}') = \frac{1}{1 + \sum_{(i',j') \in B_\alpha} \exp(I(h_{i',j'}^k) + I(p_\alpha^k))}. \tag{24}$$

As an alternative to block Gibbs sampling, mean-field (e.g., Salakhutdinov et al.[30]) can be used to approximate the posterior distribution. In all our experiments except for Section 4.5, we used the mean-field approximation to estimate the hidden layer activations given the input.[f]

3.7. Discussion

Our model used undirected connections between layers. This approach contrasts with Hinton et al.,[10] which used undirected connections between the top two layers, and top-down directed connections for the layers below. Hinton et al.[10] proposed approximating the posterior distribution using a single bottom-up pass. This feed-forward approach can often effectively estimate the posterior when the image contains no occlusions or ambiguities,[g] but the higher layers cannot help resolve ambiguities in the lower layers. This is due to feed-forward computation, where the lower layer activations are not affected by the higher layer activations. Although Gibbs sampling may more accurately estimate the posterior, applying block Gibbs sampling would be difficult because the nodes in a given layer are not conditionally independent of one another given the layers above and below. In contrast, our treatment using undirected edges enables combining bottom-up and top-down information more efficiently, as shown in Section 4.5.

In our approach, probabilistic max-pooling helps to address scalability by shrinking the higher layers. Moreover, weight-sharing (convolutions) speeds up the algorithm further.

[e] To avoid clutter, we removed all the terms that do not depend on **h** and **p**.

[f] We found that a small number of mean-field iterations (e.g., five iterations) sufficed.

[g] In our experiments, this feed-forward approximation scheme also resulted in similar posteriors of the hidden units and classification performance in most cases.

For example, convolutions between K filters and an input image are more efficient both in memory and time than repeating $K N_H^2$ times of inner products between the input image and each of the basis vectors (without weight sharing). As a result, inference in a three-layer network (with 200×200 input images) with weight-sharing but without max-pooling is about 10 times slower. Without weight-sharing, it is more than 100 times slower.

In contemporary work that was done independently of ours, Desjardins and Bengio[4] and Norouzi et al.[21] also applied convolutional weight-sharing to RBMs. Our work, however, developed more sophisticated elements such as probabilistic max-pooling to make the algorithm more scalable.

In another contemporary work, Salakhutdinov and Hinton[29] proposed an algorithm to train Boltzmann machines with layer-wise connections (i.e., the same topological structure as in DBNs, but with undirected connections). They called this model the deep Boltzmann machine (DBM). Specifically, they proposed algorithms for pretraining and fine-tuning DBMs. Our treatment of undirected connections is closely related to DBMs. However, our model is different from theirs because we apply convolutional structures and incorporate probabilistic max-pooling into the architecture. Although their work is not convolutional and does not scale to as large images as our model, we note that their pretraining algorithm (a modification of contrastive divergence that duplicates the visible units or hidden units when training the RBMs) or fine-tuning algorithm (joint training of all the parameters using a stochastic approximation procedure[32, 35, 37]) can also be applied to our model to improve the training procedure.

4. EXPERIMENTAL RESULTS

4.1. Learning hierarchical representations from natural images

We first tested our model's ability to learn hierarchical representations of natural images. Specifically, we trained a CDBN with two hidden layers from the Kyoto natural image dataset.[h] The first layer consisted of 24 groups (or "bases")[i] of 10×10 pixel filters, while the second layer consisted of 100 bases, each one 10×10 as well. Since the images were real-valued, we used Gaussian visible units for the first-layer CRBM. The pooling ratio C for each layer was 2, so the second-layer bases covered roughly twice as large an area as the first-layer bases. We used 0.003 as the target sparsity for the first layer and 0.005 for the second layer.

As Figure 3 (top) shows, the learned first layer bases are oriented, localized edge filters; this result is consistent with much previous work.[1, 9, 22, 23, 28, 33] We note that sparsity regularization during training was necessary to learn these oriented edge filters; when this term was removed, the algorithm failed to learn oriented edges. The learned second layer bases are shown in Figure 3 (bottom), and many of them empirically responded selectively to contours, corners, angles, and surface boundaries in the images. This result is qualitatively consistent with previous work.[12, 13, 18]

[h] Available at http: //www. cnbc. cmu.edu/cplab/data_kyoto.html

[i] We will call one hidden group's weights a "basis."

Figure 3. The first layer bases (top) and the second layer bases (bottom) learned from natural images. Each second layer basis (filter) was visualized as a weighted linear combination of the first layer bases.

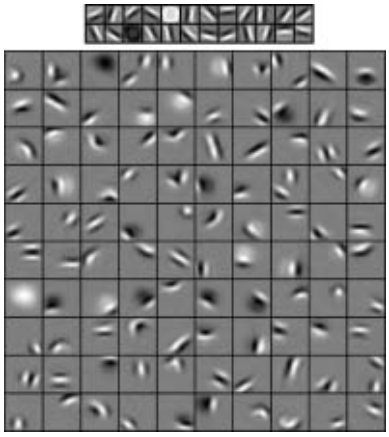


Table 1. Test classification accuracy for the Caltech-101 data.

Training size (per class)	15	30
CDBN (first layer)	53.2% ± 1.2%	60.5% ± 1.1%
CDBN (first + second layer)	57.7% ± 1.5%	65.4% ± 0.5%
Raina et al.[24]	46.6%	—
Ranzato et al.[27]	—	54.0%
Mutch and Lowe[20]	51.0%	56.0%
Lazebnik et al.[16]	54.0%	64.6%
Zhang et al.[38]	59.0% ± 0.56%	66.2% ± 0.5%

4.2. Self-taught learning for object recognition

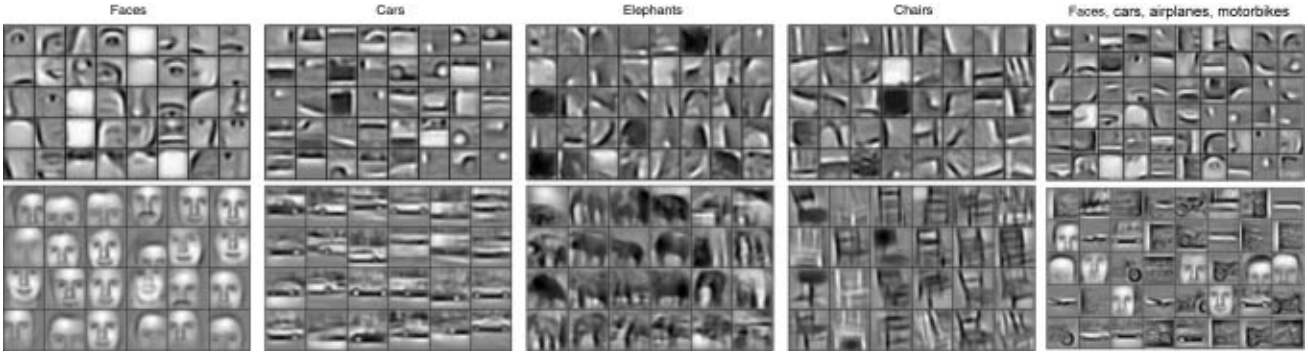
In the self-taught learning framework,[24] a large amount of unlabeled data can help supervised learning tasks, even when the unlabeled data do not share the same class labels or the same generative distribution with the labeled data. In previous work, sparse coding was used to train single-layer representations from unlabeled data, and the learned representations were used to construct features for supervised learning tasks.

We used a similar procedure to evaluate our two-layer CDBN, described in Section 4.1, on the Caltech-101 object classification task. More specifically, given an image from the Caltech-101 dataset,[5] we scaled the image so that its longer side was 150 pixels and computed the activations of the first and second (pooling) layers of our CDBN. We repeated this procedure after reducing the input image by half and concatenated all the activations to construct features. We used an SVM with a spatial pyramid matching kernel for classification, and the parameters of the SVM were cross-validated. We randomly selected 15 or 30 images per class for training test and testing set, and normalized the result such that classification accuracy for each class was equally weighted (following the standard protocol). We report results averaged over 10 random trials, as shown in Table 1. First, we observe that combining the first and second layers significantly improves the

Table 2. Test error for MNIST dataset.

Labeled Training Samples	1,000	2,000	3,000	5,000	60,000
CDBN	2.62% ± 0.12%	2.13% ± 0.10%	1.91% ± 0.09%	1.59% ± 0.11%	0.82%
Ranzato et al.[27]	3.21%	2.53%	—	1.52%	0.64%
Hinton and Salakhutdinov[11]	—	—	—	—	1.20%
Weston et al.[34]	2.73%	—	1.83%	—	1.50%

Figure 4. Columns 1–4: the second layer bases (top) and the third layer bases (bottom) learned from specific object categories. Column 5: the second layer bases (top) and the third layer bases (bottom) learned from a mixture of four object categories (faces, cars, airplanes, motorbikes).



classification accuracy relative to the first layer alone. Overall, we achieve 57.7% test accuracy using 15 training images per class, and 65.4% test accuracy using 30 training images per class. Our result is competitive with state-of-the-art results using a single type of highly specialized features, such as SIFT, geometric blur, and shape-context.[3, 16, 38] In addition, recall that the CDBN was trained entirely from natural scenes, which are completely unrelated to the classification task. Hence, the strong performance of these features implies that our CDBN learned a highly general representation of images.

We note that current state-of-the-art methods use multiple kernels (or features) together, instead of using a single type of features. For example, Gehler and Nowozin[6] rveported a better performance than ours (77.7% for 30 training images/class), but they combined many state-of-the-art features (or kernels) to improve performance. In another approach, Yu et al.[36] used kernel regularization using a (previously published) state-of-the-art kernel matrix to improve the performance of their convolutional neural network model (achieving 67.4% for 30 training examples/class). However, we expect our features can also be used in both settings to further improve performance.

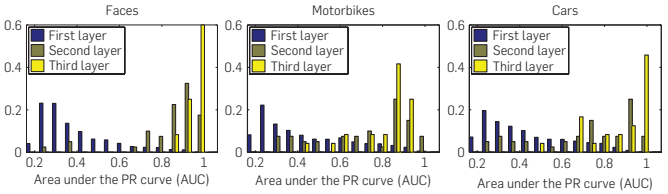
4.3. Handwritten digit classification

We also evaluated the performance of our model on the MNIST handwritten digit classification task, a widely used benchmark for testing hierarchical representations. We trained 40 first layer bases from MNIST digits, each 12×12 pixels, and 40 second layer bases, each 6×6. The pooling ratio C was 2 for both layers. The first layer bases learned pen-strokes that comprise the digits, and the second layer bases learned bigger digit-parts that combine the pen-strokes. We constructed feature vectors by concatenating the first and second (pooling) layer activations, and used an SVM for classification using these features. For each labeled training set size, we report the test error averaged over 10 randomly chosen training sets, as shown in Table 2. For the full training set, we obtained 0.8% test error. Our result is comparable to the state of the art.[27]

4.4. Unsupervised learning of object parts

We now show that our algorithm can learn hierarchical object-part representations without knowing the position of the objects and the object-parts. Building on the first layer representation learned from natural images, we trained two additional CDBN layers using unlabeled images from single Caltech-101 categories. Training was performed on up to 100 images, and testing was performed on images different than those in the training set. The pooling ratio for the first layer was set as 3. The second layer contained 40 bases, each 10×10, and the third layer contained 24 bases, each 14×14. The pooling ratio in both cases was 2. We used 0.005 as the target sparsity level in both the second and third layers. As shown in Figure 4, the second layer learned features that corresponded to object parts, even though the algorithm was not given any labels that specified the locations of either the objects or their parts. The third layer learned to combine the second layer's part representations into more complex, higher-level features. Our model successfully learned hierarchical object-part representations of most of the other Caltech-101 categories as well. We note that some of these categories (such as elephants and chairs) have fairly high intra-class appearance variation, due to deformable shapes or different viewpoints. Despite this variation, our model still learns hierarchical, part-based representations fairly robustly.

Figure 5. (top) Histogram of the area under the precision-recall curve (AUC-PR) for three classification problems using class-specific object-part representations. (bottom) Average AUC-PR for each classification problem.



Features	Faces	Motorbikes	Cars
First layer	0.39 ± 0.17	0.44 ± 0.21	0.43 ± 0.19
Second layer	0.86 ± 0.13	0.69 ± 0.22	0.72 ± 0.23
Third layer	0.95 ± 0.03	0.81 ± 0.13	0.87 ± 0.15

Higher layers in the CDBN learn features that are not only higher level, but also more specific to particular object categories. We quantitatively measured the specificity of each layer by determining how indicative each individual feature is of object categories. (This setting contrasts with most work in object classification, which focuses on the informativeness of the entire feature set, rather than individual features.) More specifically, we considered three CDBNs trained on faces, motorbikes, and cars, respectively. For each CDBN, we tested the informativeness of individual features from each layer for distinguishing among these three categories. For each feature, we computed the area under the precision-recall curve (larger means more specific). In detail, for any given image, we computed the layer-wise activations using our algorithm, partitioned the activation into $L \times L$ regions for each group, and computed the q% highest quantile activation for each region and each group. If the q% highest quantile activation in region i was γ, we then defined a Bernoulli random variable $X_{i,L,q}$ with probability γ of being 1. To measure the informativeness between a feature and the class label, we computed the mutual information between $X_{i,L,q}$ and the class label. We report results using (L, q) values that maximized the average mutual information (averaging over i). Then for each feature, by comparing its values over positive and negative examples, we obtained the precision-recall curve for each classification problem. As shown in Figure 5, the higher-level representations are more selective for the specific object class.

We further tested if the CDBN can learn hierarchical object-part representations when trained on images from several object categories, rather than just one. We trained the second and third layer representations using unlabeled images randomly selected from four object categories (cars, faces, motorbikes, and airplanes). As shown in Figure 4 (far right), the second layer learns class-specific and shared parts, and the third layer learns more object-specific representations. The training examples were unlabeled, so, in a sense, the third layer implicitly clusters the images by object category. As before, we quantitatively measured the specificity of each layer's individual features to object categories. Since the training was completely unsupervised, whereas the AUC-PR statistic requires knowing which specific

Figure 6. Histogram of conditional entropy for the representation learned from the mixture of four object classes.

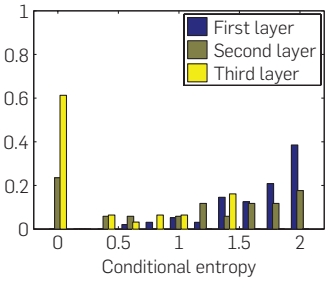


Figure 7. Hierarchical probabilistic inference. For each column: (top) input image; (middle) reconstruction from the second layer units after single bottom-up pass, by projecting the second layer activations into the image space; (bottom) reconstruction from the second layer units after 20 iterations of block Gibbs sampling.

object or object parts the learned bases should represent, we computed the conditional entropy instead. Specifically, we computed the quantile features γ for each layer as previously described, and measured conditional entropy $H(class \mid \gamma > 0.95)$. Informally speaking, conditional entropy measures the entropy of the posterior over class labels when a feature is active. Since lower conditional entropy corresponds to a more peaked posterior, it indicates greater specificity. As shown in Figure 6, the higher-layer features have progressively less conditional entropy, suggesting that they activate more selectively to specific object classes.

4.5. Hierarchical probabilistic inference

Lee and Mumford[19] proposed that the human visual cortex can be modeled conceptually as performing "hierarchical Bayesian inference." For example, imagine that you observe a face image with its left half in dark illumination, then you would still be able to recognize the face and further infer the darkened parts by combining the image with your prior knowledge of faces. In this experiment, we show that our model can tractably perform such (approximate) hierarchical probabilistic inference in full-sized images. More specifically, we tested the network's ability to infer the locations of hidden object parts.

To generate examples for evaluation, we used Caltech-101 face images (distinct from the ones the network was trained on). For each image, we simulated an occlusion by zeroing out the left half of the image. We then sampled from the joint posterior over all the hidden layers by performing

Gibbs sampling. Figure 7 shows a visualization of these samples. To ensure that the filling-in required top-down information, we compared with a control condition where only a single upward pass was performed.

In the control (upward-pass only) condition, since there is no evidence from the first layer, the second layer does not respond to the left side. However, with full Gibbs sampling, the bottom-up inputs combine with the context provided by the third layer which has detected the object. This combined evidence significantly improves the second layer representation. Selected examples are shown in Figure 7. Our method may not be competitive to state-of-the-art face completion algorithms using significant prior knowledge and heuristics (e.g., symmetry). However, we find these results promising and view them as a proof of concept for top-down inference.

5. CONCLUSION

We presented the CDBN, a scalable generative model for learning hierarchical representations from un-labeled images, and showed that our model performs well in a variety of visual recognition tasks. We believe our approach holds promise as a scalable algorithm for learning hierarchical representations from high-dimensional, complex data.

Acknowledgments

We give warm thanks to Daniel Oblinger and Rajat Raina for helpful discussions. This work was supported by the DARPA transfer learning program under contract number FA8750-05-2-0249. C

References

1. Bell, A.J., Sejnowski, T.J. The 'independent components' of natural scenes are edge filters. *Vis. Res. 37*, 23 (1997), 3327–3338.
2. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, 2007.
3. Berg, A.C., Berg, T.L., Malik, J. Shape matching and object recognition using low distortion correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.
4. Desjardins, G., Bengio, Y. Empirical evaluation of convolutional RBMs for vision. Technical report, University of Montreal, Monreal, Quebec, Canada, 2008.
5. Fei-Fei, L., Fergus, R., Perona, P. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. In *CVPR Workshop on Generative Model Based Vision*, 2004.
6. Gehler, P., Nowozin, S. On feature combination for multiclass object classification. In *Proceedings of the International Conference on Computer Vision*, 2009.
7. Grosse, R., Raina, R., Kwong, H., Ng, A.Y. Shift-invariant sparse coding for audio classification. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2007.
8. Hinton, G.E. Training products of experts by minimizing contrastive divergence. *Neural Comput. 14*, 8 (2002), 1771–1800.
9. Hinton, G.E., Osindero, S., Bao, K. Learning causally linked MRFs. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2005.
10. Hinton, G.E., Osindero, S., Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput. 18*, 7 (2006), 1527–1554.
11. Hinton, G.E., Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science 313*, 5786 (2006), 504–507.
12. Hyvarinen, A., Gutmann, M., Hoyer, P.O. Statistical model of natural stimuli predicts edge-like pooling of spatial frequency channels in V2. *BMC Neurosci. 6* (2005), 12.
13. Ito, M., Komatsu, H. Representation of angles embedded within contour stimuli in area V2 of macaque monkeys. *J. Neurosci. 24*, 13 (2004), 3313–3324.
14. Koller, D., Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, MA, 2009.
15. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the International Conference on Machine Learning*, 2007.
16. Lazebnik, S., Schmid, C., Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.
17. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D. Backpropagation applied to handwritten zip code recognition. *Neural Comput. 1* (1989), 541–551.
18. Lee, H., Ekanadham, C., Ng, A.Y. Sparse deep belief network model for visual area V2. In *Advances in Neural Information Processing Systems 20*, 2008.
19. Lee, T.S., Mumford, D. Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A 20*, 7 (2003), 1434–1448.
20. Mutch, J., Lowe, D.G. Multiclass object recognition with sparse, localized features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.
21. Norouzi, M., Ranjbar, M., Mori, G. Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
22. Olshausen, B.A., Field, D.J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature 381* (1993), 607–609.
23. Osindero, S., Welling, M., Hinton, G.E. Topographic product models applied to natural scene statistics. *Neural Comput. 18*, 2 (2006), 381–344.
24. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the International Conference on Machine Learning*, 2007.
25. Raina, R., Madhavan, A., Ng, A.Y. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the International Conference on Machine Learning*, 2009.
26. Ranzato, M., Boureau, Y., LeCun, Y. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems*, 2007.
27. Ranzato, M., Huang, F.-J., Boureau, Y.-L., LeCun, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.
28. Ranzato, M., Poultney, C., Chopra, S., LeCun, Y. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems* (2006), 1137–1144, 2006.
29. Salakhutdinov, R., Hinton, G.E. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.
30. Salakhutdinov, R., Mnih, A., Hinton, G. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine learning*, 2007.
31. Taylor, G.W., Hinton, G.E., Roweis, S.T. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems 19*, 2007.
32. Tieleman, T. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the International Conference on Machine Learning*, 2008.
33. van Hateren, J.H., van der Schaaf, A. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc. R. Soc. B 265* (1998), 359–366.
34. Weston, J., Ratle, F., Collobert, R. Deep learning via semi-supervised embedding. In *Proceedings of the International Conference on Machine Learning*, 2008.
35. Younes, L. Maximum of likelihood estimation for Gibbsian fields. *Probab. Theory Relat. Fields 82* (1989), 625–645.
36. Yu, K., Xu, W., Gong, Y. Deep learning with kernel regularization for visual recognition. In *Advances in Neural Information Processing Systems*, 2009.
37. Yuille, A.L. The convergence of contrastive divergences. In *Advances in Neural Information Processing Systems 17*, 2005.
38. Zhang, H., Berg, A.C., Maire, M., Malik, J. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

Honglak Lee (honglak@eecs.umich.edu), Computer Science and Engineering Division, University of Michigan, Ann Arbor, MI.

Roger Grosse (rgrosse@mit.edu), CSAIL, Massachusetts Institute of Technology, Cambridge, MA.

Rajesh Ranganath and Andrew Y. Ng ([rajeshr,ang]@cs.stanford.edu), Computer Science Department, Stanford University, Stanford, CA.

DOI:10.1145/2001269.2001292

Technical Perspective
Visual Reconstruction

By Carlo Tomasi

THE PAGES THAT FOLLOW boast impressive numbers: 496 processors with a total of 1,984 gigabytes of memory and 62 terabytes of disk digested nearly 460,000 Flickr pictures of Rome, Venice, and Dubrovnik. After 2.5 days, the processors output the detailed three-dimensional geometry and colors of famous landmarks and monuments in these cities. To computer vision researchers, these automatic visual reconstructions—awesome in their detail, magnitude, and fidelity—are a dream come true, never mind the occasional gaps that give the resulting scenes a faintly war-torn look.

It took decades to get here. In 1959, Edgar Hynes Thompson, then Professor of Photogrammetry at University College London, worked out the algebra for the smallest instance of the geometric side of visual reconstruction: If we take two pictures of the same scene from different viewpoints, the image coordinates of five world points are enough to compute where both points and cameras are in space. To this end, we need to know where each of the five points in one image shows up in the other, a task—called point correspondence—that in those days was performed by human operators. In 1934, Thompson himself, a young Captain of the British Royal Engineers, had designed a double microscope with reference grids and moving tables, the Cambridge Stereo-Comparator. An operator peering into the microscope wrote down coordinates of corresponding points in the two photographs. This elaborate apparatus did not just satisfy military exactness: Applied mathematicians soon proved visual reconstruction to be numerically ill-conditioned, thereby requiring extremely accurate data and carefully calibrated cameras to yield reasonable results.

In 1981, to lessen this difficulty, the British theoretical chemist and cognitive scientist Hugh Christopher Longuet-Higgins developed the first of a class of algorithms that use a large number of point pairs to solve an approximate but convex least-squares version of visual reconstruction. Unfortunately, the resulting estimates are statistically inconsistent, meaning that the output error does not vanish even as the amount of input data grows indefinitely. The modern way out is to compute an initial solution by one of the approximate methods—together with robust estimation techniques to confront omnipresent data outliers—and then refine that solution through numerical, local optimization. This refinement, called bundle adjustment, operates efficiently on large but sparse matrices with techniques that can be traced back to the 1880 work on nested dissection—a divide-and-conquer heuristic based on graph partitioning methods—by the German geodesist Friedrich Robert Helmert. Fortunately, bundle adjustment restores statistical consistency, thus opening the way to automatic computation on common imagery.

The topic of point correspondence—the other grand challenge of visual reconstruction—originated with the advent of digital cameras. People can easily judge if two image details look similar to each other, or if two pictures as a whole depict the same scene. Computers, however, find either task very difficult. In 1999, David Lowe, a professor of computer science at the University of British Columbia, showed how to describe image details well for computing point correspondences. Together with fast data structures for approximate nearest-neighbor search, Lowe's feature descriptors are the workhorse of visual matching in this paper—and of much else in computer vision.

How can visual reconstruction possibly work with images taken by unknown, disparate, uncalibrated cameras under varying weather, lighting, and exposure settings?

Yet the automated, bulk photogrammetry described here still seems an improbable achievement in the face of the difficulties of geometry and correspondence mentioned earlier. How can visual reconstruction possibly work with images taken by unknown, disparate, uncalibrated cameras under varying weather, lighting, and exposure settings?

In a way, success reveals as much about the input as it does about the computation. In a telltale statistic, only about 20% of the input images were eventually used in the reconstructions, the others being discarded at the many stations along the processing pipeline: Does the scene in this image match that of any other image in the set? Can individual features in this image be placed in accurate correspondence with those of other images? Is the resulting cloud of 3D points consistent with computed camera positions? Are colors similar enough across images to allow for texture mapping? A picture is about as likely to join the final elite as a high school senior is to make it into Duke or Cornell. Success, then, is in part tied to a sort of converse Murphy's Law that seems to hold for massive collections of tourist photographs: If something can go right, it will. If high-quality images are needed, taken under similar weather conditions and exposure settings, and from appropriately separate viewpoints that provide just the right coverage, then there are enough pictures out there that such a set will be found—if you know how. C

Carlo Tomasi (tomasi@cs.duke.edu) is professor and chair of computer science at Duke University.

DOI:10.1145/2001269.2001293

Building Rome in a Day

By Sameer Agarwal[a], Yasutaka Furukawa[a], Noah Snavely, Ian Simon[b], Brian Curless, Steven M. Seitz, and Richard Szeliski

Abstract

We present a system that can reconstruct 3D geometry from large, unorganized collections of photographs such as those found by searching for a given city (e.g., Rome) on Internet photo-sharing sites. Our system is built on a set of new, distributed computer vision algorithms for image matching and 3D reconstruction, designed to maximize parallelism at each stage of the pipeline and to scale gracefully with both the size of the problem and the amount of available computation. Our experimental results demonstrate that it is now possible to reconstruct city-scale image collections with more than a hundred thousand images in less than a day.

1. INTRODUCTION

Amateur photography was once largely a personal endeavor. Traditionally, a photographer would capture a moment on film and share it with a small number of friends and family members, perhaps storing a few hundred of them in a shoebox. The advent of digital photography, and the recent growth of photo-sharing Web sites such as Flickr.com, have brought about a seismic change in photography and the use of photo collections. Today, a photograph shared online can potentially be seen by millions of people.

As a result, we now have access to a vast, ever-growing collection of photographs the world over capturing its cities and landmarks innumerable times. For instance, a search for the term "Rome" on Flickr returns nearly 3 million photographs. This collection represents an increasingly complete photographic record of the city, capturing every popular site, façade, interior, fountain, sculpture, painting, and café. Virtually anything that people find interesting in Rome has been captured from thousands of viewpoints and under myriad illumination and weather conditions. For example, the Trevi Fountain appears in over 50,000 of these photographs.

How much of the city of Rome can be reconstructed in 3D from this photo collection? In principle, the photos of Rome on Flickr represent an ideal data set for 3D modeling research, as they capture the highlights of the city in exquisite detail and from a broad range of viewpoints. However, extracting high quality 3D models from such a collection is challenging for several reasons. First, the photos are *unstructured*—they are taken in no particular order and we have no control over the distribution of camera viewpoints. Second, they are *uncalibrated*—the photos are taken by thousands of different photographers and we know very little about the camera settings. Third, the *scale* of the problem is enormous—whereas prior methods operated on hundreds or at most a few thousand photos, we seek to handle collections two to three orders of magnitude larger. Fourth, the algorithms must be *fast*—we seek to reconstruct an entire city in a single day, making it possible to repeat the process many times to reconstruct all of the world's significant cultural centers.

Creating accurate 3D models of cities is a problem of great interest and with broad applications. In the government sector, city models are vital for urban planning and visualization. They are equally important for a broad range of academic disciplines including history, archeology, geography, and computer graphics research. Digital city models are also central to popular consumer mapping and visualization applications such as Google Earth and Bing Maps, as well as GPS-enabled navigation systems. In the near future, these models can enable augmented reality capabilities which recognize and annotate objects on your camera phone (or other) display. Such capabilities will allow tourists to find points of interest, driving directions, and orient themselves in a new environment.

City-scale 3D reconstruction has been explored previously.[2, 8, 15, 21] However, existing large scale systems operate on data that comes from a structured source, e.g., aerial photographs taken by a survey aircraft or street side imagery captured by a moving vehicle. These systems rely on photographs captured using the same calibrated camera(s) at a regular sampling rate and typically leverage other sensors such as GPS and Inertial Navigation Units, vastly simplifying computation. Images harvested from the Web have none of these simplifying characteristics. Thus, a key focus of our work has been to develop new 3D computer vision techniques that work "in the wild," on extremely diverse, large, and unconstrained image collections.

Our approach to this problem builds on progress made in computer vision in recent years (including our own recent work on Photo Tourism[18] and *Photosynth*), and draws from many other areas of computer science, including distributed systems, algorithms, information retrieval, and scientific computing.

2. STRUCTURE FROM MOTION

How can we recover 3D geometry from a collection of images? A fundamental challange is that a photograph is a two-dimensional *projection* of a three-dimensional world. Inverting this projection is difficult as we have lost the depth of each point in the image. As humans, we can experience this problem by closing one eye, and noting our diminished

The original version of this paper was published in the *Proceedings of the 2009 IEEE International Conference on Computer Vision*.

[a] This work was done when the author was a postdoctoral researcher at the University of Washington.

[b] Part of this work was done when the author was a graduate student at the University of Washington.

depth perception. Fortunately, we have two eyes, and our brains can estimate depth by correlating points between the two images we perceive. This gives us some hope that from *multiple* photos of a scene, we can recover the shape of that scene.

Consider the three images of a cube shown in Figure 1a. We do not know where these images were taken, and we do not know *a priori* that they depict a specific shape (in this case, a cube). However, suppose we *do* know that the corners of the cube as seen in the images, i.e., the 2D projections of the 3D corners, are in correspondence: we know that the 2D dots with the same color correspond to the same 3D points. This correspondence gives us a powerful set of constraints on the 3D geometry of the cameras and points.[10] One way to state these constraints is that given scene geometry (represented with 3D points) and camera geometry (a 3D position and orientation for each camera), we can predict where the 2D projections of each point should be in each image via the equations of perspective projection; we can then compare these projections to our original measurements.

Concretely, let X_i, $i = 1,\ldots, 8$ denote the 3D positions of the corners of the cube and let R_j, c_j, and f_j, $j = 1, 2, 3$ denote the orientation, position, and the focal length of the three cameras. Then if x_{ij} is the image of point X_i in image j, we can write down the image formation equations as

$$x_{ij} = f_j \prod (R_j(X_i - c_j)), \quad (1)$$

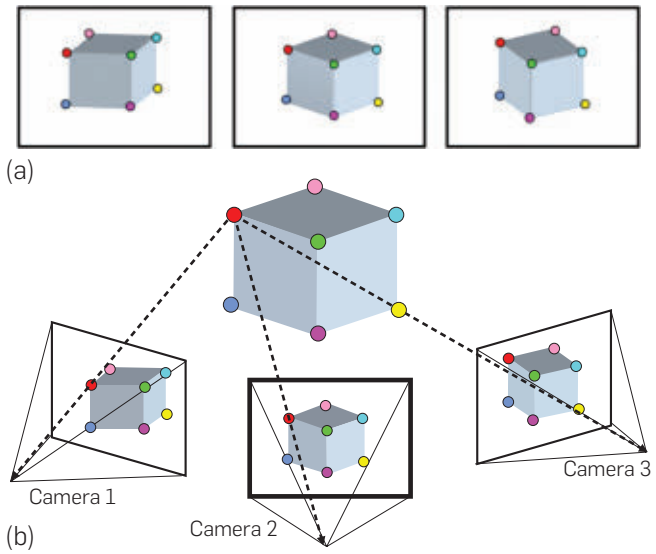
where Π is the projection function: $\Pi(x, y, z) = (x/z, y/z)$. The *Structure from Motion* (SfM) problem is to infer X_i, R_j, c_j, and f_j from the observations x_{ij}.

The standard way to do this is to formulate the problem as an optimization problem that minimizes the total squared reprojection error:

$$\arg \min_{X_i, R_j, c_j, f_j} \sum_{i \sim j} \left\| x_{ij} - f_j \Pi\left(R_j(X_i - c_j)\right) \right\|^2. \quad (2)$$

Here, $i \sim j$ indicates that the point X_i is visible in image j. An example reconstruction, illustrating reprojection error, is shown in Figure 1. While this toy problem is easily solved, (2) is in general a difficult nonlinear least squares problem with many local minima, and has millions of parameters in large scenes. Section 5 describes the various techniques we use to solve (2) at scale.

Figure 1. (a) Three images of a cube, from unknown viewpoints. The color-coded dots on the corners show the known correspondence between certain 2D points in these images; each set of dots of the same color are projections of the same 3D point. (b) A candidate reconstruction of the 3D points (larger colored points) and cameras for the image collection shown above. Each image in the collection has an associated position and orientation. This reconstruction largely agrees with the observed 2D projections; when the red 3D point is projected into each image (depicted with the dotted lines), the predicted projection is close to the observed one. In the case of Camera 3 the projection is slightly off; the resulting residual is called the *reprojection error*, and is what we seek to minimize.



3. THE CORRESPONDENCE PROBLEM

In the cube example above, we assumed that we were given as input a set of 2D correspondences between the input images. In reality, these correspondences are not given and also have to be estimated from the images. How can we do so automatically? This is the *correspondence* problem.

To solve the correspondence problem between two images, we might consider every patch in the first image and find the most similar patch in the second image. However, this algorithm quickly runs into problems. First, many image patches might be very difficult to match. For instance, a patch of clear blue sky is very challenging to match unambiguously across two images, as it looks like any other patch of sky, i.e., it is not *distinct*. Second, what happens if the second image is taken at a different time of day or with a different level of zoom?

The last 10 years have seen the development of algorithms for taking an image and detecting the most distinctive, repeatable features in that image. Such feature detectors not only reduce an image representation to a more manageable size, but also produce much more robust features for matching, invariant to many kinds of image transformations. One of the most successful of these detectors is SIFT (Scale-Invariant Feature Transform).[13]

Once we detect features in an image, we can match features across image pairs by finding similar-looking features. While exhaustive matching of all features between two images is prohibitively expensive, excellent results have been reported with approximate nearest neighbor search[18]; we use the ANN library.[3] For each pair of images, the features of one image are inserted into a k-d tree and the features from the other image are used as queries. For each query if the nearest neighbor returned by ANN is sufficiently far away from the next nearest neighbor, it is declared a match.[13]

Despite their scale invariance and robustness to appearance changes, SIFT features are *local* and do not contain any global information about the image or about the location of other features in the image. Thus feature matching based on SIFT features is still prone to errors. However, since we assume that we are dealing with rigid scenes, there are strong geometric constraints on the locations of the matching features and these constraints can be used to clean up the matches. In particular, when a rigid scene is imaged by two pinhole cameras, there exists a 3×3 matrix F, the *Fundamental matrix*, such that corresponding points x_{ij} and x_{ik} (represented in homogeneous coordinates) in two images j and k satisfy[10]:

$$x_{ij}^{\top} F x_{ij} = 0. \quad (3)$$

A common way to impose this constraint is to use a greedy randomized algorithm to generate suitably chosen random estimates of F and choose the one that has the largest support among the matches, i.e., the one for which the most matches satisfy (3). This algorithm is called Random Sample Consensus (RANSAC)[6] and is used in many computer vision problems.

4. CITY-SCALE MATCHING

Section 3 describes how to find correspondences between a pair of images. However, given a large collection with tens or hundreds of thousands of images, our task is to find correspondences spanning the entire collection. One way to think about this image matching problem is as a graph estimation problem where we are given a set of vertices corresponding to the images and we want to discover the set of edges connecting them. In this graph an edge connects a pair of images if and only if they are looking at the same part of the scene and have a sufficient number of feature matches between them. We will call this graph the *match graph*.

A naive way to determine the set of edges in the match graph is to perform all $O(n^2)$ image matches; for large collections, however, this is not practical. For a set of 100,000 images, this translates into 5,000,000,000 pairwise comparisons, which with 500 cores operating at 10 image pairs per second per core would require about 11.5 days to match, plus all of the time required to transfer the image and feature data between machines. Further, even if we were able to do all these pairwise matches, it would be a waste of computational effort since an overwhelming majority of the image pairs do not match, i.e., the graph is sparse. We expect this to be the case for images from an entire city.

Instead, our system uses a multiround scheme: in each round we propose a set of edges in the match graph, then verify each edge through feature matching. If we find more than a minimum number of features, we keep the edge; otherwise we discard it. Thus, the problem reduces to that of formulating a method for quickly predicting when two images match. We use two methods to generate proposals: whole image similarity and query expansion.

4.1. Whole image similarity

A natural idea is to come up with a compact representation for computing the overall *similarity* of two images, then use this metric to propose edges to test.

For text documents, there are many techniques for quickly comparing the content of two documents. One common method is to represent each document as a vector of weighted word frequencies[11]; the distance between two such vectors is a good predictor of the similarity between the corresponding documents.

Inspired by this work in document analysis, computer vision researchers have recently begun to apply similar techniques to visual object recognition with great success.[5, 14, 16, 17] The basic idea is to take the SIFT features in a collection of photos and cluster them into "visual words." By treating the images as documents consisting of these visual words, we can apply the machinery of document retrieval to efficiently match large data sets of photos. We use a fast tree-structured method for associating visual words with image features.[14] Each photo is represented as a sparse histogram of visual word which we weight using the well-known "Term Frequency Inverse Document Frequency" (TFIDF) method[11]; we compare two such histograms by taking their inner product. For each image, we determine the $k_1 + k_2$ most similar images, and verify the top k_1 of these. This forms the proposals for the first round of matching.

At this stage, we have a sparsely connected match graph. To derive the most comprehensive reconstruction possible, we want a graph with as few connected components as possible. To this end, we make further use of the proposals from the whole image similarity to try to connect the various connected components in this graph. For each image, we consider the next k_2 images suggested by the whole image similarity and verify those pairs which straddle two different connected components. We do this only for images which are in components of size two or more.[c]

4.2. Query expansion

After performing the two rounds of matching based on whole image similarity, we have a sparse match graph, but this graph is usually not dense enough to reliably produce a good reconstruction. To remedy this, we use another idea from text and document retrieval research—query expansion.[5]

In its original form, query expansion takes a set of documents that match a user's query, then queries again with these initial results, *expanding* the initial query. The final results are a combination of these two queries. If we define a graph on the set of documents (including the query), with similar documents connected by an edge, then query expansion is equivalent to finding all vertices that are within two steps of the query vertex.

In our system, for every vertex j in the match graph, if vertices i and k are connected to j, we propose that i and k are also connected, and verify the edge (i, k). This process can be

[c] We use $k_1 = k_2 = 10$ in all our experiments.

repeated a fixed number of times or until the match graph converges.

4.3. Distributed implementation

We now consider a distributed implementation of the ideas described above. Our matching system is divided into three distinct phases: (1) pre-processing (Section 4.3.1), (2) verification (Section 4.3.2), and (3) track generation (Section 4.3.3). The system runs on a cluster of computers (nodes) with one node designated as the master node, responsible for job scheduling decisions.

4.3.1. Preprocessing and feature extraction

We assume that the images are available on a central store from which they are distributed to the cluster nodes on demand in chunks of fixed size. Each node down-samples its images to a fixed size and extracts SIFT features. This automatically performs load balancing, with more powerful nodes receiving more images to process. This is the only stage requiring a central file server; the rest of the system operates without using any shared storage.

At the end of this stage, the set of images (along with their features) has been partitioned into disjoint sets, one for each node.

4.3.2. Verification and detailed matching

The next step is to propose and verify (via feature matching) candidate image pairs, as described in Section 3.

For the first two rounds of matching, we use the whole image similarity (Section 4.1), and for the next four rounds we use query expansion (Section 4.2).

If we consider the TFIDF vectors corresponding to the images to be the rows of a huge matrix T, then the process of evaluating the whole image similarity is equivalent to evaluating the outer product $S = TT'$. Each node in the cluster evaluates the block of rows corresponding to its images, chooses the top $k_1 + k_2$ entries in each row and reports them to the master node. Query expansion is a simple and cheap enough operation that we let the master node generate these proposals.

If the images were all located on a single machine, verifying each proposed pair would be a simple matter of running through the set of proposals and performing SIFT matching, perhaps paying some attention to the order of the verifications so as to minimize disk I/O. However, in our case, the images and features are distributed across the cluster. Asking a node to match the image pair (i, j) may require it to fetch the image features from two other nodes of the cluster. This is undesirable due to the large difference between network transfer speeds and local disk transfers, as well as creating work for three nodes. Thus, the candidate edge verifications should be distributed across the network in a manner that respects the locality of the data.

We experimented with a number of approaches with surprising results. Initially, we tried to optimize network transfers before performing any verification. In this setup, once the master node knows all the image pairs that need to be verified, it builds another graph connecting image pairs which share an image. Using `MeTiS`,[12] this graph is partitioned into as many pieces as there are compute nodes. Partitions are then matched to the compute nodes by solving a linear assignment problem that minimizes the number of network transfers needed to send the required files to each node.

This algorithm worked well for small problems, but not for large ones. Our assumption that verifying every pair of images takes the same constant amount of time was wrong; some nodes finished early and idled for up to an hour.

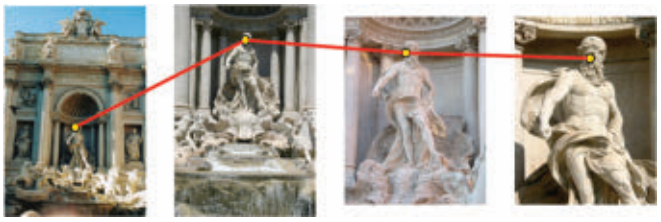
Our second idea was to over-partition the graph into small pieces, then parcel them out to nodes on demand. When a node requests a chunk of work, it is assigned the piece requiring the fewest network transfers. This strategy achieved better load balancing, but as the problem sizes grew, the graph we needed to partition became enormous and partitioning itself became a bottleneck.

The approach that gave the best result was to use a simple greedy bin-packing algorithm where each bin represents the set of jobs sent to a node. The master node maintains a list of images on each node. When a node asks for work, it runs through the list of available image pairs, adding them to the bin if they do not require any network transfers, until either the bin is full or there are no more image pairs to add. It then chooses an image (list of feature vectors) to transfer to the node, selecting the image that will allow it to add the maximum number of image pairs to the bin. This process is repeated until the bin is full. A drawback of this algorithm is that it can require multiple sweeps over all the remaining image pairs: for large problems this can be a bottleneck. A simple solution is to consider only a fixed sized subset of the image pairs for scheduling. This windowed approach works very well in practice and our experiments use this method.

4.3.3. Track generation

Until now, we have only compared two images at a time. However, when a 3D point is visible in more than two images and the features corresponding to this point have been matched across these images, we need to group these features together so that the geometry estimation algorithm can estimate a single 3D point from all the features. We call a group of features corresponding to a single 3D point a *feature track* (Figure 2); the final step in the matching process is to combine all the pairwise matching information to

Figure 2: A track corresponding to a point on the face of the central statue of Oceanus (the embodiment of a river encircling the world in Greek mythology).



generate consistent tracks across images.

The problem of track generation can be formulated as the problem of finding connected components in a graph where the vertices are the features in all the images and edges connect matching features. Since the matching information is stored locally on the compute node where the matches were computed, the track generation process is distributed and proceeds in two stages. First, each node generates tracks from its local matching data. This data is gathered at the master node and then broadcast over the network to all the nodes. Second, each node is assigned a connected component of the match graph (which can be processed independently of all other components), and stitches together tracks for that component.

5. CITY-SCALE SFM

Once the tracks are generated, the next step is to use a SfM algorithm on each connected component of the match graph to recover the camera poses and a 3D position for every track.

Directly solving Equation 2 is a hard nonlinear optimization problem. Most SfM systems for unordered photo collections are incremental, starting with a small reconstruction, then growing a few images at a time, triangulating new points, and doing one or more rounds of nonlinear least squares optimization (known as *bundle adjustment*[20]) to minimize the reprojection error. This process is repeated until no more images can be added. However, due to the scale of our collections, running such an incremental approach on all the photos at once was impractical.

To remedy this, we observed that Internet photo collections by their very nature are redundant. Many photographs are taken from nearby viewpoints (e.g., the front of the Colosseum) and processing all of them does not necessarily add to the reconstruction. Thus, it is preferable to find and reconstruct a minimal subset of photographs that capture the essential geometry of the scene (called a *skeletal set* in Snavely et al.[19]). Once this subset is reconstructed, the remaining images can be added to the reconstruction in one step by estimating each camera's pose with respect to known 3D points matched to that image. This process results in an order of magnitude or more improvement in performance.

Having reduced the SfM problem to its skeletal set, the primary bottleneck in the reconstruction process is the solution of (2) using bundle adjustment. Levenberg–Marquardt (LM) is the algorithm of choice for solving bundle adjustment problems; the key computational bottleneck in each iteration of LM is the solution of a symmetric positive definite linear system known as the *normal equations*.

We developed new high-performance bundle adjustment software that, depending upon the problem size, chooses between a truncated or an exact step LM algorithm. In the first case, a preconditioned conjugate gradient method is used to approximately solve the normal equations. In the second case, CHOLMOD,[4] a sparse direct method for computing Cholesky factorizations, is used. The first algorithm has low time complexity per iteration, but uses more LM iterations, while the second converges faster at the cost of more time and memory per iteration. The resulting code uses significantly less memory than the state-of-the-art methods and runs up to an order of magnitude faster. The runtime and memory savings depend upon the sparsity of the linear system involved.[1]

6. MULTIVIEW STEREO

SfM recovers camera poses and 3D points. However, the reconstructed 3D points are usually sparse, containing only distinctive image features that match well across photographs. The next stage in 3D reconstruction is to take the registered images and recover dense and accurate models using a multiview stereo (MVS) algorithm.

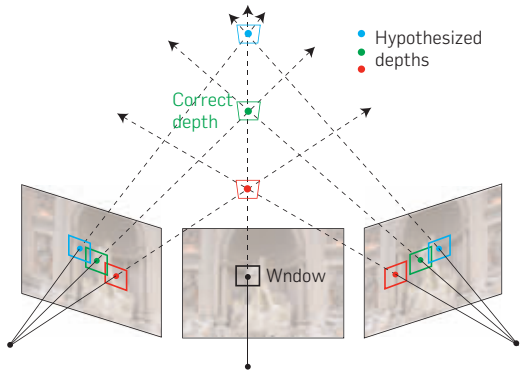
MVS algorithms recover 3D geometric information much in the same way our visual system perceives depth by fusing two views. In the MVS setting, we may have many images that see the same point and could be potentially used for depth estimation. Figure 3 illustrates how a basic algorithm estimates a depth value at a single pixel. To recover a dense model, we estimate depths for every pixel in every image and then merge the resulting 3D points into a single model.

For city-scale MVS reconstruction, the number of photos is well beyond what any standard MVS algorithm can operate on at once due to prohibitive memory consumption. Therefore, a key task is to group photos into a small number of manageable sized clusters that can each be used to reconstruct a part of the scene well.

Concretely, if we consider the SfM points as a sparse proxy for the dense MVS reconstruction, we want a clustering such that

1. Each SfM point is visible from enough images in a cluster.
2. The total number of clusters is small.
3. The size of each cluster is constrained to be lower than

Figure 3. A standard window-based multiview stereo algorithm. Given a pixel and an image window around it, we hypothesize a finite number of depths along its viewing ray. At each depth, the window is projected into the other images, and consistency among textures at these image projections is evaluated. At the true depth (highlighted in green), the consistency score is at its maximum.



a certain threshold, determined by the memory limitations of the machines.

The resulting clustering problem is a constrained discrete optimization problem (see Furukawa et al.[9] for algorithmic details).

After the clustering, we solve for scene geometry within each cluster independently using a MVS algorithm, and then combine the results.[9] This strategy not only makes it possible to perform the reconstruction, but also makes it straightforward to do so in parallel on many processors.

7. EXPERIMENTS

We report the results of running our system on three city-scale data sets downloaded from Flickr: Dubrovnik, Rome, and Venice.

The SfM experiments were run on a cluster of 62 nodes with dual quad-core processors, on a private network with 1GB/s Ethernet interfaces. Each node had 32GB of RAM and 1TB of local hard disk space with the Microsoft Windows Server 2008 64-bit operating system. For encoding the images as TFIDF vectors, we used a set of visual words, created from 20,000 images of Rome. The images used to create the visual word vocabulary were not used in any of the experiments.

Figure 4 shows reconstructions of the largest connected components of these data sets. Due to space considerations, only a sample of the results are shown here. Complete result are posted at http://grail.cs.washington.edu/rome.

For whole image similarity proposals, the top $k_1 = 10$ were used in the first verification stage, and the next $k_2 = 10$ were used in the second component matching stage. Four rounds of query expansion were done. In all cases, the ratio of the number of matches performed to the number of matches verified starts dropping off after four rounds. Table 1 summarizes statistics of the three data sets.

The SfM timing numbers in Table 1 bear some explanation. It is surprising that running SfM on Dubrovnik took so much more time than for Rome, and is almost the same as Venice, both of which are much larger data sets. The reason lies in the structure of the data sets. The Rome and Venice sets are essentially collections of landmarks which mostly have a simple geometry and visibility structure. The largest connected component in Dubrovnik, on the other hand, captures the entire old city. With its complex visibility and widely varying viewpoints, reconstructing Dubrovnik is a much more complicated SfM problem. This is reflected in the sizes of the skeletal sets associated with the largest connected components shown in Table 2.

Figure 4 also shows the results of running our MVS[9] on city-scale reconstructions produced by our matching and SfM system. Figure 4 shows MVS reconstructions (rendered as colored points) for St. Peter's Basilica (Rome), the Colosseum (Rome), Dubrovnik, and San Marco Square (Venice), while Table 3 provides timing and size statistics.

The largest dataset—San Marco Square—contains 14,000 input images which were processed into 67 clusters and yielded 28 million surface points in less than 3 h. While our system successfully reconstructs dense and high quality 3D points for these very large scenes, our models contain holes in certain places. For example, rooftops where image coverage is poor, and ground planes where surfaces are usually not clearly visible. On the other hand, in places with many images, the reconstruction quality is very high, as illustrated in the close-ups in Figure 4.

8. DISCUSSION

A search on Flickr.com for the keywords "Rome" or "Roma" results in over 4 million images. Our aim was to be able to reconstruct as much of the city as possible from these photographs in 24 h. Our current system is about an order of magnitude away from this goal. Since the original publication of this work, Frahm et al. have built a system that uses the massive parallelism of GPUs to do city scale reconstructions on a single workstation.[7]

In our system, the track generation, skeletal sets, and reconstruction algorithms are all operating on the level of connected components. This means that the largest few components completely dominate these stages. We are currently exploring ways of parallelizing all three of these steps, with particular emphasis on the SfM system.

Another issue with the current system is that it produces a set of disconnected reconstructions. If the images come with geotags/GPS information, our system can try and geo-locate the reconstructions. However, this information is frequently incorrect, noisy, or missing.

The runtime performance of the matching system depends critically on how well the verification jobs are distributed across the network. This is facilitated by the initial distribution of the images across the cluster nodes. An early decision to store images according to the name of the user and the Flickr ID of the image meant that most images taken by the same user ended up on the same cluster node. Looking at the match graph, it turns out (quite naturally in hindsight) that a user's own photographs have a high probability of matching amongst themselves. The ID of the person who took the photograph is just one kind of meta-data associated with these images. A more sophisticated strategy would exploit all the textual tags and geotags associated with the images to predict what images are likely to match distributing the data accordingly.

Finally, our system is designed with batch operation in mind. A more challenging problem is to make the system incremental.

Acknowledgments

This work was supported in part by SPAWAR, NSF grant IIS-0811878, the Office of Naval Research, the University of Washington Animation Research Labs, and Microsoft. We thank Microsoft Research for generously providing access to their HPC cluster and Szymon Rusinkiewicz for Qsplat software. The authors would also like to acknowledge discussions with Steven Gribble, Aaron Kimball, Drew Steedly and David Nister.

Figure 4. From left to right, sample input images, structure from motion reconstructions, and multiview stereo reconstructions.

Table 1. Matching and SfM statistics for the three cities.

						Time (h)		
Data set	**Images**	**Cores**	**Registered**	**Pairs verified**	**Pairs found**	**Matching**	**Skeletal sets**	**SfM**
Dubrovnik	57,845	352	11,868	2,658,264	498,982	5	1	16.5
Rome	150,000	496	36,658	8,825,256	2,712,301	13	1	7
Venice	250,000	496	47,925	35,465,029	6,119,207	27	21.5	16.5

Table 2. Reconstruction statistics for the largest connected components in the three data sets.

Data set	**CC1**	**CC2**	**Skeletal set**	**Reconstructed**
Dubrovnik	6,076	4,619	977	4,585
Rome	7,518	2,106	254	2,097
Venice	20,542	14,079	1,801	13,699

CC1 is the size of the largest connected component after matching, CC2 is the size of the largest component after skeletal sets. The last column lists the number of images in the final reconstruction.

Table 3. MVS reconstruction statistics for the four view clusters.

				Time (min)		
Landmark	**Images**	**Images in clusters**	**Clusters**	**MVS points**	**Clustering**	**Reconstruction**
St. Peter's Basilica	1,275	333	4	5,107,847	1.3	94.2
Colosseum	1,167	528	7	5,747,083	1.5	59.2
Dubrovnik	6,304	2,628	28	14,051,331	21.1	221.4
San Marco Square	13,709	5,917	67	27,707,825	39.3	176.3

References

1. Agarwal, S., Snavely, N., Seitz, S.M., Szeliski, R. Bundle adjustment in the large. In *ECCV (2)*, volume 6312 of *Lecture Notes in Computer Science* (2010). K. Daniilidis, P. Maragos, and N. Paragios, eds. Springer, Berlin, Germany, 29–42.
2. Antone, M.E., Teller, S.J. Scalable extrinsic calibration of omni-directional image networks. *Int. J. Comput. Vis. 49*, 2–3 (2002), 143–174.
3. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM 45*, 6 (1998), 891–923.
4. Chen, Y., Davis, T.A., Hager, W.W., Rajamanickam, S. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35*, 3 (2008), 1–14.
5. Chum, O., Philbin, J., Sivic, J., Isard, M., Zisserman, A. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV* (2007), IEEE, 1–8.
6. Fischler, M.A., Bolles, R.C. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach. 24* (1981), 381–395.
7. Frahm, J.-M., Georgel, P.F., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S. Building Rome on a cloudless day. In *ECCV (4)*, volume 6314 of *Lecture Notes in Computer Science* (2010). K. Daniilidis, P. Maragos, and N. Paragios, eds. Springer, Berlin, Germany, 368–381.
8. Früh, C., Zakhor, A. An automated method for large-scale, ground-based city model acquisition. *Int. J. Comput. Vis. 60*, 1 (2004), 5–24.
9. Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R. Towards internet-scale multi-view stereo. In *CVPR* (2010), IEEE, 1434–1441.
10. Hartley, R.I., Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, U.K., 2003.
11. Jones, K. A statistical interpretation of term specificity and its application in retrieval. *J. Doc. 60*, 5 (2004), 493–502.
12. Karypis, G., Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput. 20*, 1 (1998), 359–392.
13. Lowe, D. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis. 60*, 2 (2004), 91–110.
14. Nistér, D., Stewénius, H. Scalable recognition with a vocabulary tree. In *CVPR (2)* (2006), IEEE Computer Society, 2161–2168.
15. Pollefeys, M., Nister, D., Frahm, J., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S., Merrell, P. et al. Detailed real-time urban 3d reconstruction from video. *IJCV 78*, 2 (2008), 143–167.
16. Schindler, G., Brown, M., Szeliski, R. City-scale location recognition. In *CVPR* (2007), IEEE Computer Society.
17. Sivic, J., Zisserman, A. Video Google: A text retrieval approach to object matching in videos. In *ICCV* (2003), 1470–1477.
18. Snavely, N., Seitz, S.M., Szeliski, R. Photo Tourism: Exploring photo collections in 3d. *ACM Trans. Graph. 25*, 3 (2006), 835–846.
19. Snavely, N., Seitz, S.M., Szeliski, R. Skeletal graphs for efficient structure from motion. In *CVPR* (2008), IEEE Computer Society.
20. Triggs, B., McLauchlan, P., Hartley, R.I., Fitzgibbon, A. Bundle adjustment—A modern synthesis. In *Vision Algorithms '99* (1999), 298–372.
21. Zebedin, L., Bauer, J., Karner, K.F., Bischof, H. Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In *ECCV (4)*, volume 5305 of *Lecture Notes in Computer Science* (2008). D.A. Forsyth, P.H.S. Torr, and A. Zisserman, eds. Springer, Berlin, Germany, 873–886.

Sameer Agarwal (sameeragarwal@google.com), Google Inc., Seattle, WA.

Yasutaka Furukawa (furukawa@google.com), Google Inc., Seattle, WA.

Ian Simon (iansimon@microsoft.com), Microsoft Corporation, Redmond, WA.

Richard Szeliski (szeliski@microsoft.com), Microsoft Research, Redmond, WA.

Steven M. Seitz (seitz@cs.washington.edu), Google Inc. & University of Washington, Washington, Seattle, WA.

Brian Curless (curless@washington.edu), University of Washington, Washington, Seattle, WA.

Noah Snavely (snavely@cs.cornell.edu), Cornell University, Ithaca, NY.

American Association for the Advancement of Science

AAAS Science & Technology Policy Fellowships

Apply your science to serve society!

Since 1973, more than 2,200 scientists and engineers have contributed their analytical skills to policymaking in Washington, DC, while learning about the role of science in the federal government.

Career-enhancing opportunities are available in more than 30 Congressional offices and 15 federal agencies for science and engineering professionals at all career stages.

Applicants must be US citizens and hold a doctoral level degree (PhD, MD, DVM, etc.) in any scientific discipline, or a master's degree in engineering with three years of post-degree experience.

Application Deadline: December 5

Visit http://fellowships.aaas.org for more details.

Duke University

Department of Computer Science

The Department of Computer Science at Duke University in Durham, North Carolina, invites applications and nominations for tenure-track faculty positions at the assistant professor level, to begin August 2012. We are interested in strong candidates in all active research areas of computer science, as well as interdisciplinary areas.

The department is committed to increasing the diversity of its faculty, and we strongly encourage applications from women and minority candidates.

A successful candidate must have a solid disciplinary foundation and demonstrate promise of outstanding scholarship in every respect, including research and teaching. Please refer to www.cs.duke.edu for information about the department and to www.provost.duke.edu/faculty/ for information about the advantages that Duke offers to faculty.

Applications should be submitted online through the link provided at www.cs.duke.edu/facsearch. A Ph.D. in computer science or related area is required. To guarantee full consideration, applications and letters of reference should be received by December 1, 2011.

Durham, Chapel Hill, and the Research Triangle of North Carolina are vibrant, diverse, and thriving communities, frequently ranked among the best places in the country to live and work. Duke and the many other universities in the area offer a wealth of education and employment opportunities for spouses and families.

Duke University is an affirmative action, equal opportunity employer.

Fordham University

Assistant Professor, Computer & Information Science

The Department of Computer and Information Science (DCIS) invites applications for a tenure-track Assistant Professor to begin in Fall 2012. A Ph.D. in Computer Science or closely related field is required. The position requires excellence in teaching undergraduate and graduate courses, good communication skills, and demonstrated research potential with the ability to attract external research funding. Exceptional candidates in all areas are encouraged to apply but we are especially interested in candidates with expertise in informatics and data mining, bioinformatics, computational neuroscience, computer security, data communications and networking, and databases.

DCIS offers graduate and undergraduate programs at Fordham's Rose Hill campus in the Bronx and Lincoln Center campus in Manhattan. For information about the department please visit http://www.cis.fordham.edu.

Review of applications will begin February 1st. Please send a letter of application, research summary, curriculum vitae, statement of teaching philosophy, and three letters of reference to faculty_search@cis.fordham.edu, or to:

Faculty Search Committee Chair
CIS Department
Fordham University, JMH 340
441 E. Fordham Road
Bronx, NY 10458

Fordham is an independent, Catholic University in the Jesuit tradition that welcomes applications from men and women of all backgrounds. Fordham is an Equal Opportunity/Affirmative Action Employer.

Institute for Pure and Applied Mathematics (IMPA)

Research Position in Visual Computing

The Institute for Pure and Applied Mathematics (IMPA) invites applications for a research position in Visual Computing. This appointment has an initial duration of one year, with a possible extension to two years, and monthly take-home earnings R$ 7,500.00 (approximately US$ 4,500.00).

We seek an excellent researcher whose interests match the activities of the VISGRAF Laboratory at IMPA. Detailed information on the application procedures can be found at http://www.impa.br/opencms/en/pesquisa/postdoc/2011/computacao_grafica.html

Further inquiries should be addressed to pdopen@impa.br.

Indiana University

School of Informatics and Computing

The School of Informatics and Computing at Indiana University, Bloomington, invites applications for five positions beginning in Fall 2012, in the areas of bioinformatics, computer science (all subareas), computer science education research (joint position with School of Education), computer security, and social informatics. The School expects continued hiring in the coming years.

Positions are open at all levels. Applicants should have a Ph.D.in the relevant area and a well-established record (senior level) or demonstrable potential for excellence in research and teaching (junior level).

The IU Bloomington School of Informatics and Computing is the first of its kind and among the largest in the country, with unsurpassed breadth. It includes more than 70 faculty members, 500 graduate students, and strong undergraduate programs. Degrees offered include M.S. degrees in Computer Science, Bioinformatics, Human Computer Interaction Design, and Security Informatics, and Ph.D. degrees in Computer Science and in Informatics. The School has received public recognition as a "top-ten program to watch" (Computerworld) thanks to its excellence and leadership in academic programs, interdisciplinary research, placement, and outreach. The school offers excellent work conditions, including attractive salaries and research support, and low teaching loads in a setting of strong student growth.

Located in the wooded rolling hills of southern Indiana, Bloomington is a culturally thriving college town with a moderate cost of living and the amenities for an active lifestyle. IU is renowned for its top-ranked music school, high performance computing and networking facilities, and performing and fine arts.

Applicants should submit a curriculum vitae, a statement of research and teaching, and the names of 3 references (junior level) or 6 references (senior level) using the recruit link at http://hiring.soic.indiana.edu (preferred) or by mail to Faculty Search Committee, School of Informatics and Computing, 919 E 10th Street, Bloomington, IN 47408. Questions may be sent to faculty-search@soic.indiana.edu. To receive full consideration completed applications must be received by November 15, 2011.

Indiana University is an Equal Opportunity/Affirmative Action employer. Applications from women and minorities are strongly encouraged. IU Bloomington is vitally interested in the needs of Dual Career couples.

Ozyegin University

Faculty Positions

Ozyegin University, Istanbul, invites applications for faculty positions at all levels in

Computer Science. For details please see: http://www.ozyegin.edu.tr/Hakkimizda/Birimler-ve-Hizmetler/Insan-Kaynaklari/Is-Olanaklari/Faculty-Positions-in-Computer-Science

SAIC HPC Careers

SAIC, a FORTUNE 500® scientific, engineering, and technology applications company, is currently forming a dedicated team of large and small companies, as well as academic institutions, to advance the goals and objectives of the Maui High Performance Computing Center (MHPCC) Program.

The MHPCC Program objectives include software application engineering, in scalable computing technology, image and signal processing, large-scale data management, space physics, open source exploitation, and other key areas of science and engineering. SAIC is seeking experienced professionals and incumbent staff to join our team in Maui, Hawaii.

Opportunities available in the following areas:

- Administrative Support
- Applications Engineering/Visualization
- Computer Center Operations
- Configuration Management/Quality Assurance
- Data Analysis
- Facilitie/Office Management
- HPC Applications Training Specialists
- Information Assurance/Security
- Project Management
- Safety/OSHA Compliance
- Software/Database Engineering
- System/Network Engineering
- Technical Publications
- User/Technology Services

SAIC values the highly talented incumbent staff currently supporting MHPCC. All inquiries handled with strict confidentiality.

Please submit your resume at:
http://jobs.saic.com/job/Schofield-Barracks-Maui-High-Performance-Computing-Center-Job-HI-96857/1386209/?utm_source=ACM&utm_campaign=MHPCC

SAIC

Equal Opportunity Employer.

The iSchool at Drexel

Full-Time Tenure-Track Faculty Positions

The iSchool at Drexel University invites applications for several tenure-track positions at the assistant, associate, or full professor level. We welcome applications with a wide variety of teaching and research interests. We are particularly interested in applicants in the following areas; Information Security/Cyber, Security/Forensics, Archival Studies and Digital preservation & Curation. The successful candidate will have a completed doctorate in a related field, evidence of excellence in teaching and research and an interest in working with a highly collaborative, interdisciplinary faculty. To apply for this position, please apply online at: www.drexeljobs.com/applicants/Central?quickFind=74903 or visit www.drexeljobs.com and search for position number 4229.

The Johns Hopkins University

Tenure-track Faculty Positions

The Department of Computer Science at The Johns Hopkins University is seeking applications for tenure-track faculty positions. The search is open to all areas of Computer Science, with a particular emphasis on candidates with research interests in machine learning, systems, information security, computational biology, and data-intensive or health-related applications.

All applicants must have a Ph.D. in Computer Science or a related field and are expected to show evidence of an ability to establish a strong, independent, multidisciplinary, internationally recognized research program. Commitment to quality teaching at the undergraduate and graduate levels will be required of all candidates. Preference will be given to applications at the assistant professor level, but other levels of appointment will be considered based on area and qualifications. The department is committed to building a diverse educational environment; women and minorities are especially encouraged to apply.

A more extensive description of our search

VirginiaTech
Invent the Future

Tenure-track Faculty Positions in Computer Science at Virginia Tech

www.cs.vt.edu

Artificial Intelligence/Machine Learning Faculty Position
The Department of Computer Science at Virginia Tech invites applications for a full-time tenure-track position at any rank from candidates with expertise in artificial intelligence having specific emphasis on machine learning or probabilistic reasoning. The department plans on making multiple hires over multiple years in this area.

Candidates should have a record, appropriate to the desired rank, of scholarship, leadership, and collaboration in computing and interdisciplinary areas; demonstrated ability to contribute to teaching at the undergraduate and graduate levels in AI and related subjects; sensitivity to issues of diversity in the campus community; and the skills to establish and grow a multidisciplinary research group.

Salary for suitably qualified applicants is competitive and commensurate with experience. Applications must be submitted online to https://jobs.vt.edu for posting #0110872. Applicant screening will begin December 15, 2011 and continue until the position is filled. Inquiries should be directed to Dr. Dennis Kafura, kafura@cs.vt.edu.

Data-Intensive Computing Faculty Position
The Department of Computer Science at Virginia Tech invites applications from candidates in data-intensive computing for a full-time tenure-track position at any rank. Examples of research foci in the area of data-intensive computing include, but are not limited to, streaming and sensor data management, scientific databases, networked data management, query languages, workflow/provenance modeling, information integration, and database designs for modern architectures such as the cloud and pervasive appliances. The successful candidate must be able to conduct an active research program in the management and processing of massive data, such as arise in social networks, biology, GIS, astronomy, text, and/or other emerging large-scale applications.

Candidates should have a record, appropriate to their rank, of scholarship, leadership, and collaboration in computing and interdisciplinary areas; demonstrated ability to contribute to teaching at the undergraduate and graduate levels in data-related subjects (e.g., database design and system architectures); sensitivity to issues of diversity in the campus community; and the skills needed to establish and grow a multidisciplinary research group.

Salary for suitably qualified applicants is competitive and commensurate with experience. Applications must be submitted online to https://jobs.vt.edu for posting #0110874. Applicant screening will begin December 15, 2011 and continue until the position is filled. Inquiries should be directed to Dr. Lenwood S. Heath, heath@vt.edu.

Assistant Professor Computer Science (Compilers/Systems Software Engineering)
The Department of Computer Science at Virginia Tech invites applications for a full-time tenure-track position at the Assistant Professor rank from candidates with a research focus in the area of compilers, programming languages, or software engineering. The department plans to strategically grow its research presence in these areas over the coming years. Preference is given to candidates whose research efforts lie in applied domains with relevance to computer systems software, particularly for emerging architectures such as many-core processors and GPGPUs.

Candidates should have a doctoral degree in Computer Science or a cognate area, a record of significant research achievement and publication, a coherent research and teaching plan showing the potential to secure research funding, build a research program in their area of specialty, and contribute to the department's graduate/undergraduate teaching mission, and sensitivity to issues of diversity in the campus community.

The position is part of a coordinated cluster hire of a total of five positions on both the Blacksburg and National Capital Region campuses of Virginia Tech that is intended to synergistically increase research momentum in computer systems, software engineering, security, and cybersecurity by the Department of Computer Science (CS) and the Bradley Department of Electrical and Computer Engineering (ECE). The successful candidate will join the Department of Computer Science on the Blacksburg campus.

Applications must be submitted online to https://jobs.vt.edu for posting #0110873. Applicant screening will begin December 15, 2011 and continue until the position is filled. Inquiries should be directed to Dr. Godmar Back, gback@cs.vt.edu.

Virginia Tech is an Equal Opportunity/Affirmative Action Institution
See http://www.cs.vt.edu/FacultySearch for more details

and additional supporting information can be found at http://www.cs.jhu.edu/Search2012. More information on the department is available at http://www.cs.jhu.edu.

Applicants should apply using the online application which can be accessed from http://www.cs.jhu.edu/apply. Applications should be received by Dec 15, 2011 for full consideration. Questions should be directed to fsearch@cs.jhu.edu. The Johns Hopkins University is an EEO/AA employer.

Faculty Search
Johns Hopkins University
Department of Computer Science
Room 224 New Engineering Building
Baltimore, MD 21218-2694

Fax: 410-516-6134
Phone: 410-516-8775
fsearch@cs.jhu.edu
http://www.cs.jhu.edu/apply

UMBC - University of Maryland Baltimore County

An Honors University in Maryland
Information Systems Department

The Information Systems Department at UMBC invites applications for a tenure-track faculty position at the Assistant Professor level in the area of software engineering starting August 2012. Outstanding candidates in other areas will also be considered.

Candidates must have an earned PhD in Information Systems or a related field no later than August 2012. Individuals engaged in software engineering research with emphasis on empirical research, process and quality improvement, and cybersecurity are especially encouraged to apply. Ideal candidates will be engaged in research that spans two or more of these areas. Candidates should have a strong potential for excellence in research, the ability to develop and sustain an externally funded research program, and the ability to contribute to our graduate and undergraduate teaching mission.

The Department offers undergraduate degrees in Information Systems and Business Technology Administration as well as both the MS and PhD in Information Systems. In addition, the Department offers an MS and PhD in Human-Centered Computing. Consistent with the UMBC vision, the Department has excellent technical support and teaching facilities as well as outstanding laboratory space and state of the art technology. UMBC's Technology Center, Research Park, and Center for Entrepreneurship are major indicators of active research and outreach. Further details on our research, academic programs, and faculty can be found at http://www.is.umbc.edu/. Under-represented groups including women and minorities are especially encouraged to apply.

Applications will not be reviewed until the following materials are received: a cover letter, a one-page statement of teaching interests, a one-page statement of research interests, one or more sample research papers, and a CV. Applicants should also arrange to have three letters of recommendation sent to the department as soon as possible. Electronic submission of materials as PDF documents is preferred. Electronic copies should be sent to bmorris@umbc.edu. Copies can also be sent to: Dr. Aryya Gangopadhyay, Chair of Faculty Search Committee, Information Systems Department, UMBC, 1000 Hilltop Circle, Baltimore, MD 21250-5398. For inquiries, please contact Barbara Morris at (410) 455-3795 or bmorris@umbc.edu.

Review of applications will begin immediately and will continue until the position is filled. This position is subject to the availability of funds.

UMBC is an Affirmative Action/Equal Opportunity Employer and welcomes applications from minorities, women and individuals with disabilities.

University of Chicago

Faculty Positions

The Department of Computer Science at the **University of Chicago** invites applications from exceptionally qualified candidates in all areas of Computer Science for faculty positions at the ranks of **Professor, Associate Professor, Assistant Professor**, and **Instructor**. However, strongest consideration will be given this year to candidates in the following three areas: (1) systems and networking, (2) natural language processing, and (3) theory of computing.

The University of Chicago has the highest standards for scholarship and faculty quality, and encourages collaboration across disciplines. We encourage strong connections with researchers across the campus in such areas as mathematics, natural language processing, bioinformatics, log-

ic, molecular engineering, and machine learning, to mention just a few.

The Department of Computer Science (cs.uchicago.edu) is the hub of a large, diverse computing community of two hundred researchers focused on advancing foundations of computing and driving its most advanced applications. Long distinguished in theoretical computer science and artificial intelligence, the Department is now building a strong Systems research group. This closely-knit community includes the Computation Institute, the Toyota Technological Institute, and Argonne's Mathematics and Computer Science Division.

The Chicago metropolitan area provides a diverse and exciting environment. The local economy is vigorous, with international stature in banking, trade, commerce, manufacturing, and transportation, while the cultural scene includes diverse cultures, vibrant theater, world-renowned symphony, opera, jazz, and blues. The University is located in Hyde Park, a Chicago neighborhood on the Lake Michigan shore just a few minutes from downtown on an electric commuter train.

All applicants must apply through the University's Academic Jobs website. For applicants in:

(1) systems and networking, the LINK is academiccareers.uchicago.edu/applicants/Central?quickFind=51727;
(2) natural language processing, the LINK is academiccareers.uchicago.edu/applicants/Central?quickFind=51728;
(3) the theory of computing, the LINK is academiccareers.uchicago.edu/applicants/Central?quickFind=51729;
(4) all other areas, the LINK is academiccareers.uchicago.edu/applicants/Central?quickFind=51730.

A cover letter, curriculum vitae including a list of publications, a statement describing past and current research accomplishments and outlining future research plans, and a description of teaching experience must be uploaded to be considered as an applicant. Candidates may also post a representative set of publications, as well as teaching evaluations, to this website. The reference letters can be sent by mail to:

Chair, Department of Computer Science
The University of Chicago
1100 E. 58th Street, Ryerson Hall
Chicago, IL. 60637-1581

Or by email to: Recommend@mailman.cs.uchicago.edu (letters can be in pdf, postscript or Microsoft Word).

Three reference letters are required. They need to be need to be mailed or e-mailed to the above addresses and one of them must address the candidate's teaching ability. Applicants must have completed all requirements for the PhD except the dissertation at time of application, and must have completed all requirements for the PhD at time of appointment. The PhD should be in Computer Science or a related field such as Mathematics or Statistics. To ensure fullest consideration of your application all materials, including supporting letters, should be received by November 19. However, screening will continue until all available positions are filled. The University of Chicago is an Affirmative Action/Equal Opportunity Employer.

University of Kentucky
Computer Science Department

The University of Kentucky Computer Science Department expects to hire one or more tenure-track faculty to begin employment August 15, 2012. Candidates must have earned a PhD in Computer Science or closely related field at the time employment begins. Review of credentials will begin 1 October 2011 and continue until a suitably qualified candidate is found.

The department seeks to hire energetic researcher/educators who are interested in the application of advanced computing to challenging and relevant problems. We are especially interested in researchers who can collaborate to solve multidisciplinary problems. Areas of interest include, but are not limited to:

- Optimal application of high-performance computing and database systems to problems in the sciences, medicine, engineering, business and the humanities.
- Design of scalable distributed computing, communication, and storage systems that protect the economic and privacy interests of both providers and users;
- Specification, design and engineering of robust, maintainable and evolvable software systems.

The University of Kentucky Department of Computer Science, one of the oldest CS departments in the United States, has 22 faculty members committed to excellence in education, research and service. It is one of the occupants of the brand-new, LEED Gold-certified Davis Marksbury Building. The Department awards B.S., M.S., and Ph.D. degrees.

To apply, a University of Kentucky Academic Profile must be submitted at www.uky.edu/ukjobs using job # SM537241 or SM537246. For more detailed information about these positions, go to www.cs.uky.edu/opportunities/faculty.

Questions should be directed to HR/Employment (phone 1-859-257-9555 press 2 or email ukjobs@email.uky.edu, or Diane Mier (diane@cs.uky.edu) in the Computer Science Department.

Applications will be accepted as of 19 August. The application deadline is March 1, 2012, but may be extended as needed. Upon offer of employment, successful applicants must undergo a national background check as required by University of Kentucky Human Resources.

The University of Kentucky is an equal opportunity employer and encourages applications from minorities and women.

University of Massachusetts Lowell
Computer Science Department
Nontenure-Track Faculty Position at the Rank of Lecturer

The University of Massachusetts Lowell is 25 miles northwest of Boston in the high-tech corridor of Massachusetts. The Computer Science

Department has 15 tenured and tenure-track faculty, serving 250 BS students, 110 MS students, and 55 PhD students. It also offers bioinformatics options at all levels, a robotics minor, and a PhD in computational mathematics. The department has four NSF CAREER Award recipients. More information about the department can be found at www.cs.uml.edu.

The Computer Science Department invites applications for one full-time, nontenure-track faculty position at the rank of Lecturer to start January 15, 2012. Primary responsibilities are to provide high quality teaching and service to the department. This position is renewable annually, potentially leading to an appointment as Senior Lecturer following six consecutive years of outstanding performance evaluations.

Qualifications: Applicants must hold a doctoral degree in Computer Science or a closely related discipline. Experience and demonstrated excellence teaching Computer Science at the undergraduate level is required.

How to Apply: Please submit a cover letter, a current CV, teaching statement, and at least three names and contact information of references who have agreed to write letters of recommendation through the University of Massachusetts Lowell's website at http://jobs.uml.edu under "Faculty Positions". Direct submissions to the department will not be accepted.

Review of applications will begin on November 15, 2011 and continue until the position is filled. However, the position may close when an adequate number of qualified applications are received.

The University of Massachusetts Lowell is committed to increasing diversity in its faculty, staff, and student populations, as well as curriculum and support programs, while promoting an inclusive environment. We seek candidates who can contribute to that goal and encourage you to apply and to identify your strengths in this area.

University of Oregon

Department of Computer and Information Science

Professor/Department Head

The Computer and Information Science (CIS) Department at the University of Oregon invites applications for Professor/Department Head. We seek an outstanding scholar who will be excited to head a computer science department in a strong public research university. A number of senior faculty have recently or will soon retire, creating an opportunity for regeneration through multiple hires over the next several years. The department is currently developing and implementing a strategic plan for increased research prominence in the context of a college-wide plan for excellence.

We seek an individual with strategic vision and leadership abilities. The ideal candidate will be a prominent scholar with a sustained record of publication and research funding in an area of software or intelligent systems that complements existing research strengths of the department. We are looking for an innovative thinker who is eager to advance interdisciplinary scholarship, build bridges between academia and industry, mentor faculty to achieve excellence, teach at the undergraduate and graduate levels, and communicate the intellectual excitement of computer science and its broad, evolving role in contemporary society. A PhD in Computer Science or a closely related field is required.

The University of Oregon is an AAU comprehensive research university with many nationally and internationally renowned programs. It is located in Eugene, two hours south of Portland, and within one hour's drive of both the Pacific Ocean and the snow-capped Cascade Mountains. The CIS Department is part of the College of Arts and Sciences and is housed within the integrated Lorry Lokey Science Complex. The department offers B.S., M.S. and Ph.D. degrees. More information about the department, its programs and faculty can be found at http://www.cs.uoregon.edu, or by contacting the search committee at faculty.search@cs.uoregon.edu.

Applications will be accepted electronically through the department's web site (only). Application information can be found at http://www.cs.uoregon.edu/Employment/. Review of applications will begin January 4, 2012 and continue until the position is filled. Please address questions to faculty.search@cs.uoregon.edu.

The University of Oregon is an equal opportunity/affirmative action institution committed to cultural diversity and is compliant with the Americans with Disabilities Act. We are committed to creating a more inclusive and diverse institution and seek candidates with demonstrated potential to contribute positively to its diverse community.

Yale University

Faculty Position

The Department of Computer Science at Yale is seeking to fill a faculty position at the level of Assistant Professor (tenure track). We are interested in applicants from any of the following areas: Artificial Intelligence, Economics and Computation, Human Computer Interaction, Machine Learning, Networking, Operating Systems, Programming Languages, Robotics, and Theory of Computing. Applicants whose research spans two or more of these areas are particularly encouraged to apply. We seek excellent researchers who are also committed to excellence in teaching.

Members of the Yale Computer Science faculty have many opportunities to collaborate. Interdisciplinary work is encouraged with Yale's world-class faculty in such computationally active fields as biology, chemistry, economics, engineering, geophysics, management, mathematics, medicine, psychology, physics, and statistics. Yale faculty members teach excellent students, both graduate and undergraduate, in relatively small classes.

Candidates should hold a Ph.D. in computer science or a related discipline. Review of applications will begin on December 15, 2011, and candidates are encouraged to apply early. Applications from qualified women and minority candidates are especially welcome. Yale is an affirmative-action/equal-opportunity employer. The department's home page can be found at http://www.cs.yale.edu. Applicants should submit a curriculum vita, brief statements of research and teaching, and the contact information for three references. Applications should be submitted online at https://academicjobsonline.org/ajo/Yale. Questions should be directed to faculty-recruiting@cs.yale.edu.

Association for Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: AD10

You can join ACM in several easy ways:

Online
http://www.acm.org/join

Phone
+1-800-342-6626 (US & Canada)
+1-212-626-0500 (Global)

Fax
+1-212-944-1318

Or, complete this application and return with payment via postal mail

Special rates for residents of developing countries:
http://www.acm.org/membership/L2-3/

Special rates for members of sister societies:
http://www.acm.org/membership/dues.html

Please print clearly

Name

Address

City | State/Province | Postal code/Zip

Country | E-mail address

Area code & Daytime phone | Fax | Member number, if applicable

Purposes of ACM

ACM is dedicated to:
1) advancing the art, science, engineering, and application of information technology
2) fostering the open interchange of information to serve both professionals and the public
3) promoting the highest professional and ethics standards

I agree with the Purposes of ACM:

Signature

ACM Code of Ethics:
http://www.acm.org/serving/ethics.html

choose one membership option:

PROFESSIONAL MEMBERSHIP:

- o **ACM Professional Membership: $99 USD**
- o **ACM Professional Membership plus the ACM Digital Library: $198 USD ($99 dues + $99 DL)**
- o **ACM Digital Library: $99 USD (must be an ACM member)**

STUDENT MEMBERSHIP:

- o **ACM Student Membership: $19 USD**
- o **ACM Student Membership plus the ACM Digital Library: $42 USD**
- o **ACM Student Membership PLUS Print *CACM* Magazine: $42 USD**
- o **ACM Student Membership w/Digital Library PLUS Print *CACM* Magazine: $62 USD**

All new ACM members will receive an ACM membership card.
For more information, please visit us at www.acm.org

Professional membership dues include $40 toward a subscription to *Communications of the ACM*. Student membership dues include $15 toward a subscription to *XRDS*. Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

RETURN COMPLETED APPLICATION TO:

Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Questions? E-mail us at acmhelp@acm.org
Or call +1-800-342-6626 to speak to a live representative

Satisfaction Guaranteed!

payment:

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc. in US dollars or foreign currency at current exchange rate.

o Visa/MasterCard o American Express o Check/money order

o Professional Member Dues ($99 or $198) $ _____

o ACM Digital Library ($99) $ _____

o Student Member Dues ($19, $42, or $62) $ _____

Total Amount Due $ _____

Card # Expiration date

Signature

[CONTINUED FROM P. 120] of dots and connections. Where would all this information about me, or anyone else, come from?"

I started to answer, but Lydia jumped in. "That's the greatest irony of all. Nobody would let us have it, of course, but everyone, I mean everyone, just gives it to Ubatoo." It was a well-rehearsed line, every word dripping with satisfaction. "Each action, email message, trip, search, thought—all there, stored on the company's servers somewhere."

I wondered if I needed to say what should have been obvious, especially with the daily privacy fears and fiascos in the news. Ubatoo would never grant them access to the data, its most cherished possession. Only the thought of him spouting some cliché like, "Don't worry, kid. It's above your pay grade," kept me silent.

"That makes life easier," Doug said with inexplicable relief.

Instantly, Doug realized his inadvertent comment may have revealed more than he intended. Quickly changing the topic, he turned to me, saying, "So you're the one from Ubatoo who came up with all this? Does it really work?"

"Yes," I said, grateful for the chance to speak, "but not the way you..."

Lydia cut me off again. "Just take a look at this small example, Doug." A few red dots were now floating among thousands of black ones. An impenetrable jumble of thick and thin lines—connections—blanketed the screen like cartons of spilled toothpicks in a puzzle designed for Rain Man.

The "red influence," as Lydia called it, perhaps a reference to the Cold War, was now unleashed. We all watched as the red permeated the page. "Like a biological infection through water pipes," she said. "You see, if you're connected to enough bad things, at some point, your dot is going to glow bright red, too. And when it does, we'll know you're worth the resources to investigate. Just like ranking Web pages at Ubatoo, it all comes down to your connections."

Lydia was on a roll. "Let's zoom into this person's red dot. You can see the books he's read. His connections to them made them red, too. See how some are brighter than others? Those were read by others on our lists as well.

"Just like ranking Web pages at Ubatoo, it all comes down to your connections."

Even the soft drinks we've seen him drink are all redder because of him. It's all in our graph."

"Zoom in on the drinks," Doug said, amused. "Let's see which are, um, suspicious."

Lydia did as she was told, despite regretting her choice of example. On the screen were a dozen circles, each its own shade of red. She worked her way down the list until she got to the reddest of all—the most suspicious in the set. She breathed the name reverently and stepped back from the table to let the discovery sink in.

I couldn't take it anymore. "No. No. You can't just..."

This time it was Doug who cut me off, "Relax, Stephen, I get it. Of course the system doesn't work for drinks." As I began to ease back in my chair, Doug began issuing commands to Lydia. "Use this to put together a list of the most suspicious people, books, Web sites, music, whatever you can think of, and send it to me ASAP!"

I think I mumbled something about how the system was never meant to be used like this. But what was the point? Doug was no longer paying attention; he stared at a sea of red circles, desperately trying to recall his daughter's favorite soda, fearing he had just connected her to Lydia's mess. Lydia was standing proudly at the head of the table with an enormous grin. Her gaze rested triumphantly on my hand, which was now, I realized, clutching a "most suspicious" choice of soft drink. C

Ubatoo.com is a ubiquitous, and entirely fictional, Internet company providing email, search, social networking, online credit cards, and shopping services through a single Web site. It exists only in **Shumeet Baluja**'s 2011 novel *The Silicon Jungle: A Novel of Deception, Power and Internet Intrigue* published by Princeton University Press, a thriller that explores the friction at the intersection of privacy, civil rights, security, and ethics.

Future Tense, one of the revolving features on this page, presents stories and essays from the intersection of computational science and technological speculation, their boundaries limited only by our ability to imagine what will and could be.

DOI:10.1145/2001269.2001296

Shumeet Baluja

Future Tense
A Person of Influence

Inferred connections map our past and predict our future.

WITHIN MINUTES OF arriving at NSA headquarters, I had already aggravated my host, Lydia. It started the moment I offered to get her a soft drink only to realize I had no money to pay for hers—or even for my own. As she grudgingly paid for both, her irritation only grew as I explained I was so accustomed to the free food and drinks at Ubatoo I didn't realize some offices were still so antiquated in their treatment of employees.

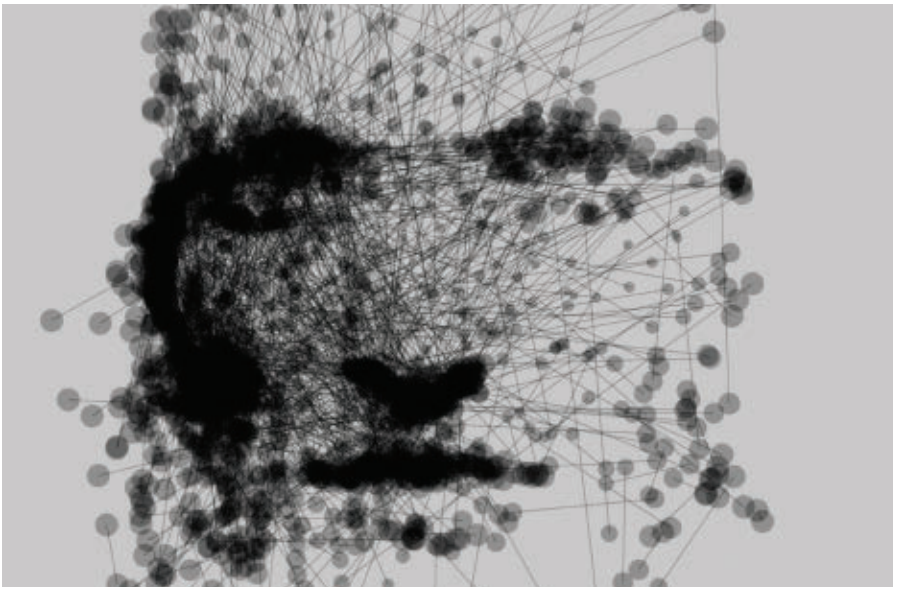
Ubatoo.com, my former employer, epitomized Silicon Valley. Early on, it set the standard for perks and eccentricities and had grown to be synonymous with the Internet itself. Write an email message, update your personal status, buy something online, watch a video, upload a picture, search a secret desire you've never shared with anyone—Ubatoo made it all possible.

Today's meeting was intended as a brainstorming session on deploying Ubatoo's Web-ranking algorithms to help NSA rank "suspects" on its ever-growing watch lists. Lydia, however, was in pure selling mode.

"We live in a connected world," she said, the moment Doug, her boss, was seated. "It's not just about the people you say are your 'friends' but about all the connections we infer."

An ode to minimalism, her first slide was a white screen with two tiny dots in the center. "One is you, Doug," she said. "The other is your daughter, Deirdre. When you make a phone call to her, we connect the dots." She tapped a key and a connecting line materialized.

"We know you write her an email message every day." The connection between them grew thicker.



"Deirdre calls and texts her boyfriend, Josh, right?" A dot for Josh emerged, already connected to Deirdre. "He's taking a course." A line between Josh's dot and his professor, Dr. Kione, appeared.

"And this Kione teaches political media by examining half a dozen Web sites, one of which is a known extremist breeding ground." Six new dots popped up, one for each site, five black, one red, indicating "under surveillance," all connected back to Kione.

"Now it's just a game of 'connect the dots'; it's unlikely he supports the extremist groups he's encountered, but he might have mentioned them in class." Like a slow drip across the screen, the red seeped outward from the lone red dot, flowing through all the connections it touched. "It's possible Josh talked about it with Deirdre." The red continued its crawl through the thick and thin connections between other dots along the way, getting fainter with each one. "It's unlikely she would talk about it with you. Unlikely, but not impossible." The red had almost disappeared by the time it reached back to Doug's no-longer pristine black dot.

"You're not terribly red, so you're okay," said Lydia, "but imagine we repeated this procedure with every Web site you visited, every person you called or emailed, every product you purchased—each represented here by a tiny dot, connected by thick and thin lines back to you. Some connections will inevitably link you to persons of interest, to things and events we monitor. You'll be connected to all those little red dots we've worked so hard to identify."

Doug was skeptical. "There would be billions [CONTINUED ON P. 119]

ILLUSTRATION BY SERGIO ALBIAC

Call for Nominations

The ACM Doctoral Dissertation Competition

Rules of the Competition

ACM established the Doctoral Dissertation Award program to recognize and encourage superior research and writing by doctoral candidates in computer science and engineering. These awards are presented annually at the ACM Awards Banquet.

Submissions

Nominations are limited to one per university or college, from any country, unless more than 10 Ph.D.'s are granted in one year, in which case two may be nominated.

Eligibility

Each nominated dissertation must have been accepted (successfully defended) by the department between October 2010 and September 2011. Exceptional dissertations completed in September 2010, but too late for submission last year will be considered. Only English language versions will be accepted. Please send a copy of the thesis in PDF format to emily.eng@acm.org.

Sponsorship

Each nomination shall be forwarded by the thesis advisor and must include the endorsement of the department head. A one-page summary of the significance of the dissertation written by the advisor must accompany the transmittal.

Deadline

Submissions must be received by **October 31, 2011** to qualify for consideration.

Publication Rights

Each nomination must be accompanied by an assignment to ACM by the author of exclusive publication rights. (Copyright reverts to author if not selected for publication.)

Publication

Winning dissertations will be published by ACM in the ACM Digital Library, not by Springer as previously noted.

Selection Procedure

Dissertations will be reviewed for technical depth and significance of the research contribution, potential impact on theory and practice, and quality of presentation. A committee of five individuals serving staggered five-year terms performs an initial screening to generate a short list, followed by an in-depth evaluation to determine the winning dissertation.

The selection committee will select the winning dissertation in early 2012.

Award

The Doctoral Dissertation Award is accompanied by a prize of $20,000 and the Honorable Mention Award is accompanied by a prize of $10,000. Financial sponsorship of the award is provided by Google.

For Submission Procedure

See http://awards.acm.org/html/dda.cfm

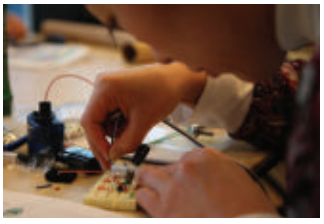
Association for Computing Machinery

Discover the Future of CG at SIGGRAPH Asia 2011

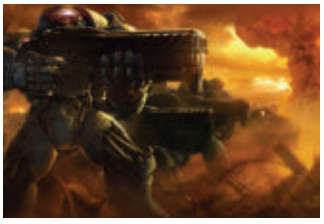
Be inspired by this year's spectacular line-up of conference programs. Establish connections with top-notch industry players on the exhibition floor and with academics at SIGGRAPH Asia 2011.

Screenings of Animation and Visual Effects

Demonstrations, Displays and Installations

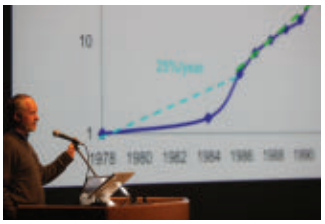


Instructional Sessions



Production Talks

Products, Services and Recruitment



Industry Insights and Forecasts

Networking Opportunities

Research and Development Presentations

REGISTER NOW AND SAVE

Network with and reach out to the industry's movers and shakers.

Choose from Full Conference or Basic Conference Passes, with admission to all or a variety of programs and events.

Join us in Hong Kong this December by registering online from August at http://www.siggraphasia2011.com

Register by 31 October to enjoy Early Bird discounts!

SIGGRAPHASIA2011 HONG KONG

The 4th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia

Conference: 12-15 December | **Exhibition:** 13-15 December Hong Kong Convention and Exhibition Centre

Sponsored by ACM SIGGRAPH

www.SIGGRAPH.org/ASIA2011