COMMUNICATIONS OF THE ACM

CACM.ACM.ORG
07/2013 VOL.56 NO.07

Computational Epidemiology

Diffusion Curves

The Great Robotics Debate

Cake Cutting

All the Items Fit to Print

Synchronization via Procrastination

Association for Computing Machinery

UBICOMP 2013

The 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing

ISWC 2013

The 17th International Symposium on Wearable Computers

September 8-12, Zurich, Switzerland

The 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (**UBICOMP**) is the premier outlet for novel research contributions in ubiquitous and pervasive computing systems. It is the result of a merger of the two most renown conferences in the field: Pervasive and UbiComp.

The International Symposium on Wearable Computers (**ISWC**) is the premier forum for wearable computing and issues related to on-body and worn mobile technologies. It brings together researchers, product vendors, fashion designers, textile manufacturers, users, and related professionals.

UBICOMP and **ISWC** will be held as a single event, featuring a single registration rate. Attendees of either conference will have full access to all tracks of the other. The event will feature over 100 individual paper presentations, as well as 21 pre-conference workshops.

Confirmed keynote speaker:
Thad Starner *(Google X Lab & GeorgiaTech)*
Technical Lead / Manager on Google's Project Glass

Early registration closes July 12, 2013

Locally organized by **ETH** Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

www.ubicomp.org www.iswc.net

Sponsored by SIGCHI SIGMOBILE

A.M. Turing Award

Call for Nominations

http://amturing.acm.org/nominate

Association for Computing Machinery

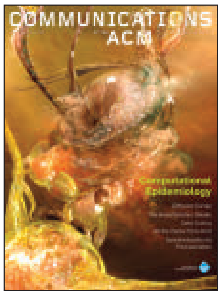
Advancing Computing as a Science & Profession

COMMUNICATIONS OF THE ACM

PHOTOGRAPH COURTESY OF OLAF DIEGEL/ODD GUITARS

Association for Computing Machinery
Advancing Computing as a Science & Profession

07/2013
VOL. 56 NO. 07



About the Cover:
At press time, global health officials were tracking a new deadly virus first noted in the Middle East and more recently spotted in parts of Europe. This month's cover story—*Computational Epidemiology*—examines how computer models are used to understand and control the diffusion of viral diseases through populations, often in real time. While this latest threat is not of epidemic strength (yet), such models may play a critical role in tracking and possibly containing its path. Cover illustration by AS&K Visual Science, www.ask-visual.com.

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.

STAFF

DIRECTOR OF GROUP PUBLISHING
Scott E. Delman
publisher@cacm.acm.org

Executive Editor
Diane Crawford
Managing Editor
Thomas E. Lambert
Senior Editor
Andrew Rosenbloom
Senior Editor/News
Larry Fisher
Web Editor
David Roman
Editorial Assistant
Zarina Strakhan
Rights and Permissions
Deborah Cotton

Art Director
Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Directors
Mia Angelica Balaquiot
Brian Greenberg
Production Manager
Lynn D'Addesio
Director of Media Sales
Jennifer Ruzicka
Public Relations Coordinator
Virginia Gold
Publications Assistant
Emily Williams

Columnists
Alok Aggarwal; Phillip G. Armour; Martin Campbell-Kelly; Michael Cusumano; Peter J. Denning; Shane Greenstein; Mark Guzdial; Peter Harsha; Leah Hoffmann; Mari Sako; Pamela Samuelson; Gene Spafford; Cameron Wilson

CONTACT POINTS

Copyright permission
permissions@cacm.acm.org
Calendar items
calendar@cacm.acm.org
Change of address
acmhelp@acm.org
Letters to the Editor
letters@cacm.acm.org

WEBSITE

http://cacm.acm.org

AUTHOR GUIDELINES

http://cacm.acm.org/guidelines

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY 10121-0701
T (212) 626-0686
F (212) 869-0481

Director of Media Sales
Jennifer Ruzicka
jen.ruzicka@hq.acm.org

Media Kit acmmediasales@acm.org

Association for Computing Machinery (ACM)
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA
T (212) 869-7440; F (212) 869-0481

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:10.1145/2483852.2483853

Moshe Y. Vardi

The Great Robotics Debate

Are robots and automation destroying more jobs than they are creating?

THE FIELD OF artificial intelligence has been accompanied by a vigorous debate essentially from its very beginnings. Alan Turing addressed the issue of machine intelligence in 1950 in what is probably his most well known paper, "Computing Machinery and Intelligence," where he proposed the "Imitation Game," now known as the "Turing Test," as an operational definition for machine intelligence. The main focus of the paper is the possibility of machine intelligence. Turing carefully analyzed and rebutted arguments against machine intelligence and stated, "I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted." This view was later strenuously contested by philosophers such as Hubert Dreyfus and John Searle, who argued against the possibility of intelligent machines.

Many of the early AI pioneers were also brimming with unbounded optimism. Marvin Minsky wrote in 1967: "Within a generation...the problem of creating 'artificial intelligence' will substantially be solved." Perhaps because of such overoptimism, AI has suffered from repeated "AI Winters": periods that were characterized by slow progress and a dearth of research funding. AI researchers refer to the "First AI Winter," 1974–1980, and "Second AI Winter," 1987–1993.

More recently a major debate has erupted among economists regarding the impact of robots and automation on jobs and the possibility of *technological unemployment*. The traditional approach by economists is skeptical of Luddism—the fear of the inevitable changes brought about by new technology. Such a position was expressed by Kenneth Rogoff, who wrote last year, "Since the dawn of the industrial age, a recurrent fear has been that technological change will spawn mass unemployment. Neoclassical economists predicted that this would not happen, because people would find other jobs, albeit possibly after a long period of painful adjustment. By and large, that prediction has proven to be correct." But in a recent working paper, "Smart Machines and Long-Term Misery," Jeffrey Sachs and Laurence Kotlikoff posed the question, "What if machines are getting so smart, thanks to their microprocessor brains, that they no longer need unskilled labor to operate?" After all, they point out, "Smart machines now collect our highway tolls, check us out at stores, take our blood pressure, massage our backs, give us directions, answer our phones, print our documents, transmit our messages, rock our babies, read our books, turn on our lights, shine our shoes, guard our homes, fly our planes, write our wills, teach our children, kill our enemies, and the list goes on."

It is in the context of the Great Recession that people started noticing that while machines have yet to exceed humans in intelligence, they are getting intelligent enough to have a major impact on the job market. In his 2009 book, *The Lights in the Tunnel: Automation, Accelerating Technology and the Economy of the Future*, Martin Ford asked: "What economic impact will technological acceleration have as we anticipate recovery from the current crisis—and in the years and decades ahead?" In their 2011 book, *Race Against The Machine: How the Digital Revolution is Accelerating Innovation, Driving Productivity, and Irreversibly Transforming Employment and the Economy*, Erik Brynjolfsson and Andrew McAfee argued that "technological progress is accelerating innovation even as it leaves many types of workers behind."

It is only this year that this debate moved from economics to computer science. In an interview with Steven Cherry earlier this year for *IEEE Spectrum*'s Techwise Conversations, I argued that "by 2045 machines will be able to do if not any work that humans can do, then a very significant fraction of the work that humans can do" (http://spectrum.ieee.org/podcast/at-work/tech-careers/the-job-market-of-2045), and wondered whether we are ready for a world in which half the adult population does not work. In a follow-up interview, Henrik Christensen, a Georgia Tech professor of robotics, disagreed and argued that automation is still creating more jobs than it destroys (http://spectrum.ieee.org/pod-cast/robotics/industrial-robots/robots-are-not-killing-jobs-says-a-roboticist).

This is a fundamental debate. Robotics is clearly going to be a key economic enabler, as declared in the recent report "Roadmap for U.S. Robotics," perhaps as transformative as the Internet, but will its fruits be distributed equitably or will it create classes of haves and have-nots? We are launching this debate in the pages of *Communications* with this editorial and a Viewpoint article by Martin Ford (p. 37). I view this discussion as one of the most important our community needs to engage in, and I hope to see a robust conversation.

Moshe Y. Vardi, EDITOR-IN-CHIEF

Your **brain** will be **delighted**.
Both sides.

Find yourself among thousands of techno-enthusiasts as you engage in a dazzling array of mind-expanding programs, informative sessions, and blockbuster events showcasing the latest in computer graphics and interactive techniques.

SIGGRAPH2013
Left Brain + **Right** Brain

The **40th** International **Conference** and **Exhibition** on **Computer Graphics** and **Interactive Techniques**

Conference 21–25 July 2013
Exhibition 23–25 July 2013
Anaheim Convention Center

Sponsored by ACM**SIGGRAPH**

www.siggraph.org/s2013

DOI:10.1145/2483852.2483854 Vinton G. Cerf

'But Officer, I Was Only Programming at 100 Lines Per Hour!'

It's summer and I thought it might be worth stirring up some thunder and lightning with another controversial topic. As many readers know, there is a category of engineer that is sometimes called "Registered" or "Professional." This is not to say that most engineers, registered or not, are not professionals, but it turns out the terms are specifically and legally meaningful. While the procedures and practices vary from state to state in the U.S. (and elsewhere), there is often an exam to pass or at least a questionnaire to fill out and a registration step of some kind that authorizes the engineer to label herself or himself as "Professional."

Among the privileges granted to professional engineers is the right to undertake certain kinds of projects, to certify designs, to oversee design and implementation, and so on. These projects are often large and potentially involved in what could be life-threatening conditions of use. Examples include designs for bridges, buildings, aircraft, roads, railways, levees, dams, and hydroelectric- and nuclear power-generation systems. Indeed, some medical devices may require some oversight and certification for use. The reason for this attention to qualification and certification is obvious: public safety is involved one way or another.

I am sure some of you will have figured out where I am going with this. I think we would all agree that software (and its underlying hardware) is increasingly a part of our daily lives in large and small measure. The programs that control the car engine, the controllers for appliances (like water heaters, heating and cooling systems, washers and dryers, ovens, and other fairly large-scale and power-consuming devices), and financial management and trading systems are potentially hazardous if they have errors that lead to excessive load, mechanical malfunction, unexpected behaviors, and so on.

One wonders whether legislative bodies and perhaps insurance companies will reach conclusions that suggest software designers and implementers should have imposed upon themselves similar regulatory requirements, authorizations, and certifications. I once made a living writing or designing software for IBM, MCI, and the USG, among others. Some of that software or at least its design was pretty widely used (for example, MCI Mail, the Internet, time-sharing services, among others). With the increasing incidence of hacking, denial-of-service attacks, penetration, malware infection, and botnet creation, one might be tempted to argue that software developers should bear more responsibility for the behavior of their software.

I think many of you would agree that a test or questionnaire is not likely to provide assurance that a "certified professional" programmer's work is free of flaws. I am not even sure what sort of test could be proposed that would provide such assurance. But it is very tempting to imagine a balance between certification and liability is worth some consideration. How can we provide motivation to a well-intended programmer to reduce the risk to users of his or her software? What incentives would companies that create software or market it have to reduce liability by appointing a "certified software engineer" to take responsibility for the safety and reliability of the software?

Plainly one would expect that any kind of licensing would have to produce some kind of safe harbor for both the employer and the employee. Perhaps reduction of insurance premiums?

In the past, I have had no problem convincing myself that the idea of professional licensing along these lines is a dumb idea. But we do have certification for users or maintainers of certain kinds of software or equipment (for example, for Microsoft software, Cisco equipment). I take pride in believing that I, and many colleagues, see themselves as professional in spirit if not in name and that we strive to deliver reliable code. I also believe that no one really knows how to write absolutely bug-free code, especially if the input and operating state space is innumerably large. So accepting liability that the code won't break or be broken is a pretty scary thought.

We are so dependent on an increasing number of programs, large and small, it is difficult to believe the software profession will escape some kind of deep accountability in the future. We many avoid this in the near term, but I am not so sure in the long term.

OK, I have my cast-iron three-piece suit in place...What do you think?

Vinton G. Cerf, ACM PRESIDENT

DOI:10.1145/2483852.2483856

Plenty More Hacker Motivations

Exploring the malicious hacker problem, Zhengchuan Xu et al.'s "Why Computer Talents Become Computer Hackers" (Apr. 2013) overlooked many motivations for computer talents becoming computer hackers, due, possibly, to cultural differences between China and the West, as suggested in the article, or just to the authors' limited findings, which were based on interviews with only six known computer hackers in Shanghai. Moreover, the article did not distinguish between hackers who ethically go to extremes of computing and malicious hackers willing to do harm to achieve their goals.

My own research at SRI International, 1971–1995, funded by the U.S. National Science Foundation, U.S. Department of Defense, and U.S. Department of Justice, involved interviews with more than 100 notorious malicious computer hackers in the U.S. and Western Europe. That work was documented in case studies in my books *Crime by Computer* (1976) and *Fighting Computer Crime* (1983), both published by Charles Scribner's Sons, Inc. Malicious hacking is still motivated in much the same way as it always has been, starting with phone phreaking in the 1960s.

One significant motivation the article clearly overlooked is that some young hackers are looking for shortcuts to a high-paying career in information technology without first undergoing a formal education. Such an irrational strategy typically concentrates on learning from manuals, the Web, personal experimentation, and experienced hackers, as noted in the article, believing that if they engage in sufficiently outrageous but brilliant conduct, with ends justifying means, as was done by some highly publicized malicious hackers before them, they will be noticed by their victims and hired at high pay to protect society from further harm.

My own interviews revealed many kinds of deviant behavior associated with ready access to the powerful, pervasive, vulnerable, fragile information technology at the heart of almost every organization's and user's operations, including negativism, delusions of grandeur, infantile ideals, grandiose, overt behavior, frequent regressions, compensatory mechanisms, peer pressure, fragile ego, irrationality, antisocial behavior, poor grooming and personal habits, unhealthy diet, use of alcohol and drugs, squalid living, disrespect for authority, deception, thievery and burglary, piracy, extortion, endangerment, misuse, abuse, negligence, sabotage, espionage, misrepresentation, intimidation, physical and mental violence, bullying as subject and object, autism, idiot-savantism, sibling rivalry, reaction to parental aberrations and broken homes, hero worship, sexual excess, attention deficit disorder, and more.

I agree with the findings in the article as far as they went, that "... hacker candidates encounter porous security, are tolerated by some academics, and are encouraged by like-minded individuals." They are also sometimes inadvertently encouraged by naive reformers at hacker conferences and by the early success of a few malicious hacker heroes (such as "Cap'n Crunch," "Fiber Optic," and Kevin Mitnick). These hacking mentors and others are described in my books, as well as in others, including *Hackers, Heroes of the Computer Revolution* by Steven Levy (Anchor Press/Doubleday, 1984), *The Hacker Crackdown* by Bruce Sterling (Bantam, 1992), and *The Fugitive Game* by Jonathan Littman (Little, Brown and Company, 1996).

Donn B. Parker, Los Altos, CA

Too Early for Verdict on MOOCs

Michael A. Cusumano's Viewpoint "Are the Costs of 'Free' Too High in Online Education?" (Apr. 2013) gave an essentially positive answer despite hedging with phrases like, "Maybe, but maybe not." It was clear from the context which side he is on, saying, for example, "The industries I follow closely are still struggling to recover from the impact of free." Comparing traditional institutions of higher learning with online courses is somewhat unfair, as it involves contrasting institutions that have undergone decades or even centuries of refinement with a new paradigm still in its infancy. Moreover, face-to-face interaction in a classroom is wonderful under ideal conditions, but universities in much of the world simply lack ideal conditions; for example, face-to-face interaction is unlikely when a disengaged professor teaches hundreds of students in a large lecture hall where those students have difficulty even seeing the professor or projected slides or reading the scribbles on the board and are often in uncomfortable seats or no seat at all in the case of overenrolled classes. Students sometimes forego courses they prefer in favor of available ones and might wait weeks before receiving feedback on homework submissions or exam papers because the professor is too busy or there are too few teaching assistants. Online courses have been successful in part because they can provide immediate feedback and facilitate peer-to-peer interaction, as well as flexibility and variety. Institutions of higher learning must adapt to this wonderful new resource, and not take a defensive stance, insisting on business as usual.

Behrooz Parhami, Santa Barbara, CA

What We Want from Computer Science

One way to address the question "Is computer science a science?" is to imagine having to translate it into another language. We would immediately confront two difficulties: "computer science" generally translates to something like "informatics," and, in other languages, the word "science" typically refers to any rigorous intellectual discipline, even in the humanities. The question then translates to "Is informatics a rigorous intellectual discipline?" where the answer is surely yes. But in his Viewpoint "The Science in Computer Science" (May 2013), Peter J. Denning clearly adopted the typical English speaker's view of science as abbreviating "natural science," something like physics or geology. The question then translates to "Is informatics like physics or geology?" and looks like nonsense. Making matters worse, Denning's focus on experimental science seemingly excluded topics like cosmology and evolutionary biology, where "reproducibility of results" is out of the question; nobody can repeat the big bang or the evolution of life on Earth. (Moreover, alchemists were fond of experimentation.) The quest to discover the science in computer science seems to rely on semantic questions. Does it really matter whether computer science is a form of engineering or instead an applied science? Does the existence of natural information processes make computer science more rigorous or significant?

I am sure the exercise involves legitimate goals that could be made clearer by asking specific questions; for example, does the subject use sound methods that deliver trustworthy results? (Economists should ask themselves this one.) Does computer science get the prestige/recognition/funding it deserves? How can we convey a clear understanding of it to the wider public? I suggest we focus on such specific, unambiguous questions and not get bogged down on the issue of what exactly counts as a science.

Lawrence C. Paulson, Cambridge, England

Author's Response:

I listed seven criteria for a field to be considered a science in the common meaning: "a discipline that employs the scientific method." Computer science meets them all. Reproducibility is one, and indeed cosmology and evolutionary biology strive for results others can reproduce. The degree to which computer science integrates science, engineering, and mathematics affects answers to fundamental questions about methodology (How do we practice computer science?), pedagogy (How do we teach it?) and dissemination (How do we communicate it?). For more, check the ACM Ubiquity symposia on science (http://ubiquity.acm.org/symposia.cfm).

Peter J. Denning, Monterey, CA

Accessibility Out of the Box

We agree with Vinton G. Cerf's President's Letter "Why Is Accessibility So Hard?" (Nov. 2012) that computer science should be one field where accessibility is a given. Indeed, there is no technical reason accessibility should be impossible, and any lack of accessibility is likely due to non-technical reasons. People in and out of computer science should appreciate that accessibility is both necessary (people growing old is justification enough) and may also be a source of profit, as well as inspiration. Though an average programmer cannot anticipate all user needs, accessibility typically boils down to structuring the application logically rather than graphically, as in the structured Web pages of Cascading Style Sheets. Accounting for accessibility from the beginning of a development project is more likely to yield better, more maintainable software, a principle all programmers should learn.

Generic screen readers involve both advantages and weaknesses but permit access even to applications not designed to be accessible. However, such access does not always work as intended, especially with graphically designed applications, though scripting engines in screen readers make it possible to tune reader behavior; advanced users can write and share such scripts.

Multiple accessibility settings may be overwhelming for some users, despite their usefulness, in light of the diversity of user needs, but coping with them could constitute a completely new profession involving discussions with users and proposing and testing solutions in the search for optimal combinations.

Moreover, developing accessible substitutes for existing applications from scratch is not effective, as they are not sustainable over time due to lack of resources; we have seen several projects starve to death this way. Accessibility should be integrated into the entire user software stack, not just alongside it; accessibility would thus be sustainable, as with internationalization and security. When an API is available, users themselves would be able to write dedicated interfaces to access shared software. We have seen such arrangements work very well, though having users write their own code depends on making computer science education itself more accessible.

Finally, accessibility should be delivered out of the box, on all computers in the wild, waiting to be triggered, including their hardware, BIOS, bootloader, and OS installer; for example, the Debian Linux installer includes a speech synthesizer that should be generalized to all operating systems.

Sébastien Hinderer, Nancy, France and
Samuel Thibault, Bordeaux, France

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.

Coming Next Month in COMMUNICATIONS

Evolutionary Robotics

Are We Free to Code the Law?

Differences in Productivity and Impact Across Computer Science Subareas

Is Over-Censorship a Fatal Mistake?

Spectral Sparsification of Graphs: Theory and Algorithms

Plus the latest news about storage breakthroughs, medical tricorders, and the ways crowdsourcing unites online workers.

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:10.1145/2483852.2483857 http://cacm.acm.org/blogs/blog-cacm

Ph.D. Students Must Break Away From Undergraduate Mentality

Jason Hong considers how students working on their doctorates in computer science must adapt and evolve to succeed.

Jason Hong
Ph.D.'s from the Faculty's Perspective
http://cacm.acm.org/blogs/blog-cacm/157012-phds-from-the-facultys-perspective/fulltext
November 5, 2012

One of Blog@CACM's bloggers, Philip Guo, has been doing a great job in discussing grad school from the Ph.D. student's perspective. I figured it would be good to offer a complementary perspective of graduate school, distilling what I have learned over the past years in advising and working with Ph.D. students.

Break Out of the Undergraduate Mentality

A common challenge for a lot of new Ph.D. students is that they still have an "undergraduate mentality," where they believe grades still matter (they do, but only marginally so), and there will always be someone there to tell you what to do.

It is natural that a lot of new Ph.D. students think in this way. The primary form of evaluation for undergrads is grades, and new Ph.D. students have had 4+ years optimizing their thinking and work processes for that system. Similarly, most undergrad courses are geared toward "demonstrate that you can build this" rather than trying to answer more important questions like "what should we build, and why?"

As such, grades in graduate school do not really matter beyond the fact that you are above the minimum bars as set by your department. Instead, the main form of evaluation for Ph.D. students is progress toward research which, for better or for worse, can be approximately boiled down to research publications.

Now, having said that, you should do more than the bare minimum amount of work needed for your courses. The instructors for your course will be on committees reviewing your progress, and they will also be your future peers. Leaving a bad impression here will not help your case. Furthermore, you might miss a lot of wonderful opportunities for learning new things if you only do the bare minimum.

Most Ph.D. students grow out of this undergrad mentality in a semester or two. The best thing you can do is realize this frame of thinking exists, and to watch out for it if you feel it creeping over you.

Own Your Research

If you are waiting around for someone to tell you what to do, you are doing it wrong. The most impressive story I have ever heard about owning your research is from Ron Azuma's retrospective "So Long, and Thanks for the Ph.D." Azuma tells the story of how one graduate student needed a piece of equipment for his research, but the shipment was delayed due to a strike. The graduate student flew out to where the hardware was, rented a truck, and drove it back, just to get his work done.

Note that, however, it is very likely you actually *will* start out by having someone telling you what to do. The vast majority of new Ph.D. students are very smart and energetic individuals. However, the main things new Ph.D. students lack are an understanding of the research process and experience with the research literature. As such, it is usually easier to start out with a directing style of research with new students, with advisors telling you to read this paper, install this software, or build this app.

Over time, as you become more familiar with research methods and what has been done in the past, you need to take more initiative and ownership in your work. As you progress in your research career, you should still listen to your advisor, but take what they say only as advice. Ultimately, you have to own your research. You are the person who has to drive it forward, and you have to do what it takes to get it done.

How do I know when a student is doing really well? It is when the best thing I can do for them is to just get out of their way.

Be Willing to Push Back

The second-worst talk I have ever witnessed in my life was actually by someone quite famous in their field. I came away from the presentation feeling quite embarrassed, pretty much the same way you feel after watching Ricky Gervais doing another one of his cringe-worthy performances. The work this researcher presented made no sense whatsoever. There was no stated problem being solved, no rationale for why he did it the way he did, no new and interesting insights, and no clear innovation. It was pretty clear that this person had not read any research papers or seen any related commercial products in the area in over a decade.

I think there were two reasons why someone with such an accomplished career could make such a major misstep. First, this individual had a very strong personality and could easily dominate any conversation. Second, I think he inadvertently surrounded himself with people who were simply unwilling or unable to push back. In this particular case, without a strong counterweight, this researcher led his team into unfruitful directions that just made no sense, either from a research perspective or from a product perspective.

I tell new students this story as a cautionary tale.

I tell them this story because I want them to be able to stand up for themselves and make a strong case for what they are doing or what they want to do. Do not forget, if you are working full-time on a project, you are spending far more time thinking about the issues than your advisor is. You have 40+ hours a week to think about the problem you are working on, while your advisor probably has 5–10 hours (on top of teaching, travel, committees, and interactions with other students—this is why all faculty seem to exhibit varying degrees of brain damage).

If something does not make sense, or if you think there is a better way to do something, you need to push back and make a solid case for what you think is the right thing to do. In situations like these, I have always found Heilmeier's Catechism to be an effective tool for thinking about problems. My students will tell you that I often have them take a step back and go through Heilmeier's standard set of questions to make sure they have thought through what they want to do and why.

Be Active in the Social Dimension of Research

A common mistake I have seen with some graduate students is overlooking the social aspects of research. Graduate school is more than just taking classes, doing research, writing papers, and going to conferences. Graduate school is also about becoming a member of a larger community. It is about becoming familiar with the methods, tools, and values of your community. It is about participating in workshops, sharing your ideas with others, and helping to grow the field. And it is also about learning from each other, building on other people's work, and communicating with others what you have learned.

This same perspective also applies to your role in your university, in that you should be a good citizen for your local community as well. This might include helping to organize student lunches, participating in reading seminars, and making bridges with folks outside of your area.

There are several reasons why Ph.D. students who can navigate the social dimension of research do better. They can absorb ideas from more areas, increasing their potential to make connections that others have not seen yet. They can rely on more people for help, whether it is getting simple questions answered or getting access to more data or resources. Perhaps most importantly, they have a stronger social support network that can help them get through tough times (of which there will be many in grad school).

Build Up Your Skills, but Get Out as Soon as You Can

Although computer science is quite diverse, the best Ph.D. students across all areas are actually quite similar. They are the ones who have solid critical thinking skills, a high degree of creativity, a strong work ethic, good writing and presentation skills, and a demonstrated ability to work independently with little supervision. Most students do not start with all of these skills, and so these are the goals you should be aiming for and critiquing yourself against as you make progress in your graduate work.

Graduate studies can be and will be tough. There will be numerous dead ends, frustrations, hardware and software that does not work the way it should, countless deadlines, and sleepless nights. On the other hand, there will be a lot of rewards — in the joy of making lifelong friends, mastering new skills, discovering new findings, and helping others. Perhaps the best way to close this column is with something I once heard attributed to Stu Card, a pioneer the field of human-computer interaction: "Grad school will be the best years of your life. Having said that, get out as soon as you can."

Comments:

One thing I would add is that, when you have passed your preliminary exams and present your research plan to your committee, consider it a binding contract. Do what you agreed to, and graduate.

—Anonymous

In addition to almost all the agreeables, I think good research starts with being 'challenged with a good problem,' accomplished with 'sustained never-say-die efforts,' 'self-evolutionally rather revolutionary approach' and 'rigorosity overall in all the stages,' and finished with 'acceptable answers and other challenging problems.' Though commonly thought to be as output, it appears mythical to aim and windup by publishing. The dissemination and sharing may be immediate/intermediate gain only, but not 'fine-grains' to be looked up to.

—K. Mustafa, JMI

Jason Hong is an associate professor at Carnegie Mellon University.

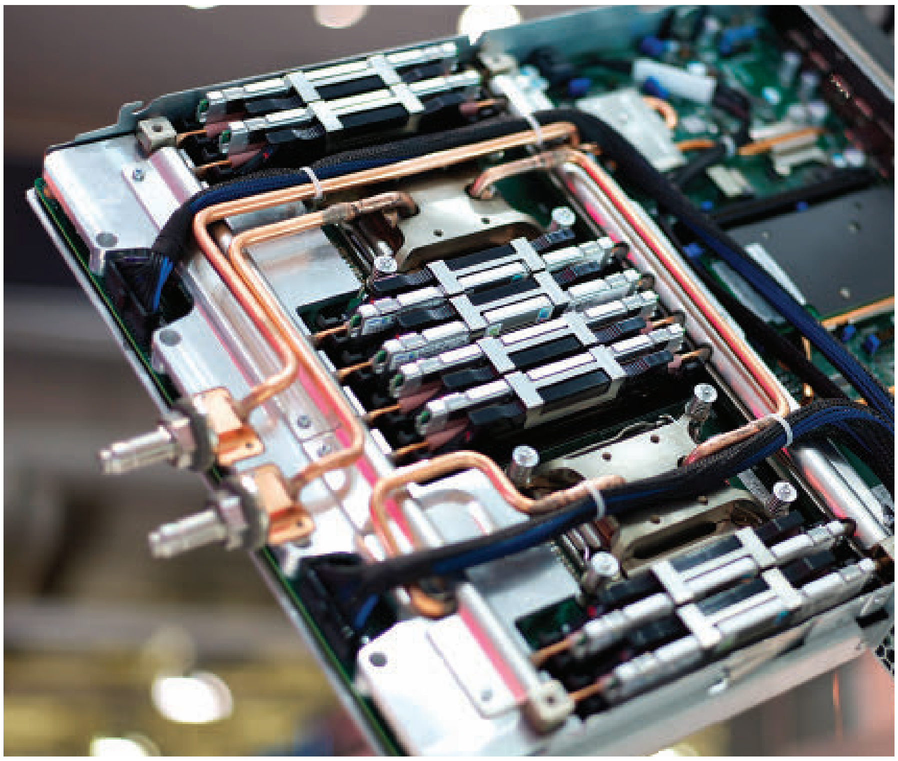
Science | DOI:10.1145/2483852.2483858 Esther Shein

Keeping Computers Cool From the Inside

New techniques could cut the power required to avoid overheating.

The issue of how to keep computers cool generally calls to mind the techniques used in enterprise data centers; that is where energy has traditionally been concentrated, so to speak. Yet, improved cooling from the inside out is crucial to reducing the carbon footprint of computers—otherwise, the need to consume increasing amounts of energy to offset the heat generated by computers will severely hamper the advancement of computing, experts in the field say.

Cooling the data center is no longer enough. In a moderate climate, for example, air-cooling a data center constitutes about 30% of the total energy it consumes and is a substantial contributor to its carbon footprint, according to a 2010 IBM report *Direct Waste Heat Utilization From Liquid-Cooled Supercomputers*. Room air conditioning is the second-largest energy component of data center operations and is highly inefficient, the report's authors state, adding, "More than 40% of the carbon footprint of an air-cooled data center is caused not by computing but by powering the cooling systems needed to keep the microelectronic components from overheating."


IBM's System x iDataPlex dx360 M4 internal server cooling system.

Advances in micro-miniaturization of electronic components required rapid growth in transistor density and a rise of clock frequency of integrated electronic circuits; this, in turn, led to increased heat generation rates in electronic chips. While chips are currently cooled primarily by forced air convection, some industry observers believe this technique will not be sufficient for next-generation electronics, which require more efficient and compact cooling solutions to maintain acceptable operating temperatures.

The computer industry has become hindered by the thermal issue, which is among the top five most pressing concerns of computer companies and users, maintains Bruno Michel, manager of Advanced Thermal Packaging at IBM Zurich Research Laboratory and one of the report's authors. "The risk is that development will stop in just a few years after the 30 years of Moore's Law,'' he said. With data centers making up about 2% of overall U.S. electrical usage and the rapid growth of data processing translating to an increase in energy consumption, the challenge for computer manufacturers is to provide more energy-efficient products.

Fortunately, progress is being made by a number of research institutes and computer technology companies.

At Purdue University, for example, the Cooling Technologies Research Center (CTRC) has a number of projects under way focused on high-conductivity thermal heat spreaders; specifically, vapor chambers and heat pipes. These devices are small, sealed chambers in the same form factor as standard metal heat spreaders, which "efficiently transport heat caused by cyclical vaporization and condensation of an internal working fluid," according to Suresh V. Garimella, Goodson Distinguished Professor of Mechanical Engineering at Purdue. The CTRC has also worked on developing physics-based models for the devices' performance, and to experimentally measure their thermal transport characteristics. The Center's work covers the entire spectrum of heat dissipation from the chip, board, and building scale, he says.

Replacing traditional Al/Cu heat spreaders with appropriate, low-cost, new materials can reduce the total thermal resistance of a cooling package, according to the CTRC. Carbon nanotubes (CNTs) can play a key role in this; CNTs are synthesized on heat spreaders to improve heat transfer at interface, says Garimella.

"One of the ubiquitous thermal management challenges is the thermal resistance inherent to contact between any system components,'' he explains. "Often, the performance of [even] highly optimized thermal packages is governed by these resistances. Additionally, the mechanical stresses [that] develop from thermal cycling at these interfaces often dictates the entire system's reliability." He says CNTs have a "high inherent thermal conductivity, which makes them a suitable material for joining components with a low thermal contact resistance." Their mechanical properties, which allow flexibility between interfaces for expansion and contraction with thermal cycling, may also improve the reliability of these interfaces, compared to other attachment methods.

Replacing traditional Al/Cu heat spreaders with appropriate, low-cost, new materials can reduce the total thermal resistance of a cooling package, according to the CTRC.

Vapor chamber technologies are also being investigated for use in portable and handheld electronic devices, says Garimella. "There is specific interest in taking advantage of the reduced thermal resistance for heat spreading provided by these devices, but their overall size and thickness have historically been unfeasibly large,'' he says. "We will study the operational limits of heat pipes both experimentally and theoretically for very thin geometries, with an ultimate goal of providing heat pipe design guidelines for these applications." The goal of the center's work is to come up with commercially viable, ultra-thin heat pipes within the next five years, he adds.

Water is considered a better coolant for microprocessors than air because of its higher heat capacity and thermal conductivity, according to the report *Experimental Investigation of a Hot Water Cooled Heat Sink for Efficient Data Center Cooling: Towards Electronic Cooling with High Exergetic Utility.* "Water cooling using compact manifold micro-channel (MMC) heat sinks is one of the most promising strategies to meet the cooling requirement of chips in next-generation data centers," write the report's authors. Using hot water as a coolant in state-of-the-art processors can eliminate the need for air cooling entirely, ensuring "a high exergetic utility in the system," along with heat sinks that are designed to be robust and scalable.

High-performance microchannel heat sinks and other liquid cooling solutions are intended to dissipate high heat fluxes at the local chip level, concurs Garimella, and are among the technologies applicable to use to augment exterior cooling options. He adds that, "Typically, ultimate heat rejection occurs to ambient air, and there are a number of low-power-consumption air cooling technologies such as piezoelectric fans, synthetic jets, and mist cooling strategies that address this need."

At giant chipmaker Intel Corp., much work has been done on not just cooling chips, but also in reducing the load of what needs to be cooled, says principal engineer Michael K. Patterson, who is focused on thermal, power, and energy efficiency. "We've done a lot to reduce the idle power on computers. In the past there was very little difference between idle power and peak power; if the computer was doing nothing, it used pretty much the same amount of power as when it was very active, and at a high utilization rate."

One area of concentration for Intel has been on changing the speed of a chip, and creating the ability to slow it down if it does not need its full clock speed, while keeping just enough performance, he says. "If it's waiting for network traffic … it will slow itself down and use less energy, which obviously helps … anywhere between idle and peak power."

When the chip is not running as fast and not generating as much heat, internal fans slow down, reducing the energy needed to run them. "As we reduce the silicon load, the support energies get reduced too, and that's where we can make a difference," Patterson says.

With each generation of Intel chips, Patterson says, the goal is to try to reduce power consumption and improve performance at the same time, thereby increasing performance per watt. The impact of this could range from more battery life for your phone, tablet, or notebook computer at the portable product level, to getting more science done for your energy investment in high-performance computing at the enterprise or supercomputer level, he says.

Commercial Viability

Besides more energy-efficient chips, other products are also becoming commercialized that are aimed at reducing the heat load of electronics.

Ionic wind cooling and piezoelectric fans are available, for example, although piezo fans are a relatively new technology that has not yet found its way into a wide variety of applications. While offering benefits to large computers/servers, the bulk of the focus for piezo fans appears to be in smaller consumer electronics, according to Nick Gilligan, production manager at Midé Technology Corp., an engineering company that makes the fans.

Piezo fans operate on the principle of piezoelectricity (an inherent material property in which electricity is created as a result of mechanical stress), and they do not suffer the same pitfalls as traditional fans, Gilligan says. Piezoelectric fans are typically configured as a cantilever beam, with a tip that can achieve "significant displacement to create airflow," he says. "Piezoelectric fans offer extremely high reliability and do not suffer from the failure modes inherent in traditional fan designs. There are no moving parts to break, they are nearly silent, and have practically infinite lifespans under known driving conditions."

GE is using "Dual Piezoelectric Cooling Jets" (DCJ) that are each 1mm thick in a prototype Core i7-powered laptop. Based on a technology GE researchers first developed for commercial jet engines, they "behave as a micro-fluidic bellows that provide high-velocity jets of air to cool electronic components," according to GE. These jets consume a fraction of the power of a traditional fan, and so the cooling technology could add an extra 30 minutes of battery life to a laptop, GE believes. Additionally, the company says, the dual piezo cooling jets can move the same amount of air as a cooling fan twice its size.

Other cooling technologies under development include high-heat flux vapor chamber heat spreaders for military applications, and dielectric fluid microchannel heat sinks for automotive power electronics cooling.

To explore the use of vapor chambers (passive heat transport devices that utilize capillary forces to circulate a fluid to capture waste heat in electronic devices and transfer it to a heat sink or cold plate for dissipation) and heat pipes in military applications, a group of CTRC faculty was awarded a $2.7-million U.S. Defense Advanced Research Projects Agency (DARPA) grant in partnership with Raytheon Company from 2008–2011.

Purdue aided in the development of a "Radio Frequency Thermal Ground Plane" (RFTGP) that provides state-of-the-art commercial heat spreaders with what Garimella calls significantly improved performance. As part of this work, Raytheon was provided an experimentally validated model for heat transport in vapor chambers, which Garimella says has improved the readiness of the device, and has led to additional reliability testing. "The thermal resistance reduction benefit provided by the developed RFTGP will have direct impact by decreasing the operational temperature and increasing the reliability of current solid-state devices,'' he explains. Raytheon expects the RFTGP technology to have a significant impact on both defense and commercial electronics packaging technology, he notes.

Converting Heat Into Electricity

Thermoelectric generators (TEGs), also known as thermogenerators, are

Piezoelectric fans have "no moving parts to break, they are nearly silent, and have practically infinite lifespans under known driving conditions."

Member News

WREAKING HAVOC WITH MOORE'S LAW

The demand for transactional memory on multicore embedding systems wreaks havoc with Moore's Law, says Ruth Iris Bahar, professor of Engineering at Brown University, Providence, RI.

The laws of physics dictate that Moore's Law is maxing out, says Bahar. "It takes longer and it's more costly to figure out how to shrink things down so you can double performance on multicore embedded systems that require faster performance, more efficient power consumption, and improved heat dissipation on mobile devices and gaming applications."

Bahar began her engineering career designing microprocessors. At Brown University, she develops new approaches to power dissipation and improving reliability in high-performance multiprocessors. Her work has received funding from groups including the Semiconductor Research Consortium and Intel. "Embedded systems are more constrained in terms of memory capacity and power dissipation. Extended battery life, reliability, and product longevity are huge issues," Bahar says. "Imagine doing the actual computing on your smartphone to generate an immediate response, instead of sending it to the cloud for computation, for such things as image tracking."

Her challenge is to make transactional memory work on an embedded system in a way that does not demand enormous bandwidth. "If it requires fancy changes to the hardware or consumes too much energy, it's not viable. You need to increase throughput and minimize complexity," Bahar says.

Ironically, she says, while she develops the technology, "my kids are better at using it."

—Laura DiDio

Energy efficiency is arguably a major challenge of the 21st century, claim the authors of IBM's liquid-cooled supercomputers report.

devices that convert heat directly into electricity. Garimella says it is possible that TEGs will be able to recharge portable electronics with their own waste heat or, at least, to diminish the power consumption of office or portable electronics.

"Unlike larger-scale thermodynamic waste heat recovery cycles, such as the organic Rankine cycles currently being researched in the Center, thermoelectric devices scale favorably to small sizes," he says. That makes it feasible—from a size standpoint—to embed thermoelectric modules for waste heat recovery in device-scale systems, he says. However, he warns, the efficiency of these thermoelectrics is "still too low for economic feasibility at these scales."

Energy efficiency is arguably a major challenge of the 21st century, claim the authors of IBM's liquid-cooled supercomputers report. Computer systems can no longer be designed solely on the criteria of computational performance, they maintain, stating "The new target must be high performance and low net power consumption (and, concomitantly, low net carbon footprint)." This requires a continued focus on more energy-efficient processors which, in turn, means striving for improved transistor designs, as well as high-performance cooling capabilities that reduce power consumption. C

Further Reading

A. Jain and S. Ramanathan
"Theoretical Investigation of Sub-ambient On-Chip Microprocessor Cooling," Proceedings of the Tenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronics Systems (ITHERM), 2006, San Diego, CA, 2006

E. G. Colgan, B. Furman, M. Gaynes, N. LaBianca, J. H. Magerlein, R. Polastre, R. Bezama, K. Marston, and R. Schmidt
"High Performance and Subambient Microchannel Cooling," ASME Journal of Heat Transfer, vol. 129(8), pp. 1046-1051, Oct. 2006. .

M. Saini and R. L. Webb
"Heat Rejection Limits of Air Cooled Plane Fin Heat Sinks for Computer Cooling," IEEE Transactions on Components and Packaging Technologies, vol. 26, no. 1. Mar. 2003.

B. Agostini, M. Fabbri, J. E. Park, L. Wojtan, J. R. Thome, and B. Michel
"State-of-the-Art of High Heat Flux Cooling Technologies," Heat Transfer Eng. 28, No. 4. 2007.

R. Mongia, K. Masahiro, E. DiStefano, J. Barry, W. Chen, M. Izenson, F. Possamai, A. Zimmermann, and M. Mochizuki
"Small Scale Refrigeration System for Electronics Cooling Within a Notebook Computer," Proceedings of the IEEE ITHERM 2006, San Diego, CA.

R. Chu, R. Simons, M. Ellsworth, R. Schmidt, and V. Cozzolino
"Review of Cooling Technologies for Computer Products," IEEE Trans. Device Materials Reliability 4, No. 4. 2004.

M. L. Kimber and S. V. Garimella
"Measurement and Prediction of the Cooling Characteristics of a Generalized Vibrating Piezoelectric Fan," International Journal of Heat and Mass Transfer Vol. 52. 2009.

T. Brunschwiler, G.I. Meijer, S. Paredes, W. Escher, and B. Michel
"Direct Waste Heat Utilization From Liquid-Cooled Supercomputers," Proceedings of the 14th International Head Transfer Conference, August 2010.

P. Kasten, S. Zimmermann, M.K. Tiwari, B. Michel, and D. Poulikakos
"Experimental Investigation of a Hot Water Cooled Heat Sink for Efficient Data Center Cooling: Towards Electronic Cooling with High Exergetic Utility," Frontiers in Heat and Mass Transfer, Vol. 1, No. 2 (2010).

Esther Shein is a freelance technology and business writer based in the Boston area.

Milestones

Computer Science Honors

IEEE HONORS DENNIS WITH VON NEUMANN MEDAL

The Institute for Electrical and Electronics Engineers (IEEE) has named Jack Dennis, a principal investigator at the Computer Science and Artificial Intelligence Laboratory (CSAIL) of the Massachussets Institute of Technology (MIT), recipient of the 2013 IEEE John von Neumann Medal, "for fundamental abstractions to implement protection in operating systems and for the dataflow programming paradigm."

Dennis, a professor emeritus in the MIT Department of Electrical Engineering and Computer Science, was the original leader of the Computation Structures Group at CSAIL, where he led the development of dataflow models of computation and novel principles of computer architecture inspired by dataflow models. He currently is engaged in research on functional programming principles and related principles of computer architecture, and is applying these concepts in the design of a novel advanced multiprocessor chip for general-purpose computing.

Dennis received the 1984 Eckert-Mauchly Award for contributions to the field of computer architecture, and is a Fellow of the IEEE and of the ACM.

The IEEE John von Neumann Medal is presented annually for outstanding achievements in computer-related science and technology.

For more information about Dennis' work, see: http://www.csail.mit.edu/user/1552.

AAAS SELECTS ANITA JONES FOR PHILIP HAUGE ABELSON AWARD

The American Association for the Advancement of Science (AAAS) has selected Anita Jones, University Professor Emerita of computer science at the University of Virginia, to receive its highest honor, the 2012 Philip Hauge Abelson Award.

A specialist in computer security systems, Jones is being honored for her scientific and technical achievements in computer science; her contributions as a mentor, inspiration, and role model for other scientists and engineers; and her lifetime of public service to government, professional institutions, academia, and industry, according to the AAAS.

The Abelson Award is given annually to either a public servant in recognition of sustained exceptional contributions to advancing science, or to a scientist whose career has been distinguished both for scientific achievement and for other notable services to the scientific community.

Technology | DOI:10.1145/2483852.2483859 Samuel Greengard

All the Items Fit to Print

3D printing has come of age. It promises to revolutionize a wide range of industries and profoundly change the way people buy and consume.

The future of manufacturing may seem far removed from a typical household. If Hod Lipson has his way, however, a machine that can fabricate just about anything will land in living rooms and offices within a few years.

Lipson, a professor of engineering at Cornell University, foresees a world brimming with 3D printers that spit out an array of things. Already, the devices have been used to fabricate coffee mugs, medical devices, food items, bicycles, furniture, musical instruments, and sailing vessels. "The technology promises to revolutionize many fields and fundamentally change our lives," says Lipson, the co-author of *Fabricated: The New World of 3D Printing* (Wiley, 2013).

It is no small matter. 3D printers—which lay down plastics, metals, and almost any other material, adding layer upon layer until a design is completed—turn the traditional concept of whittling away at wood, metals, and other materials—"subtractive" manufacturing—upside down. In fact, advances in hardware, along with increasingly sophisticated software, have recently unleashed a wave of innovation that is now sweeping through research labs and into the workplace and home. "3D printing is evolving in many ways and at a very rapid pace," states Simon Leigh, an affiliated research fellow in the School of Engineering at the University of Warwick in the U.K.

Like any emerging technology, 3D printing creates opportunities, but it also raises questions and some serious concerns, particularly revolving around the illicit use of printers to fabricate drugs, weapons, keys, counterfeit parts, and numerous other items. What makes 3D printing so powerful is an ability to use open source designs to rapidly and inexpensively print objects—thereby bypassing conventional distribution channels. This makes it possible to create a limited series or print run of an item. Moreover, 3D printers can fabricate designs that are difficult and too expensive to produce using conventional manufacturing methods; for example, an individual might fabricate a star-shaped cake, or a lamp that displays a complex 3D mathematical form.

Moving Beyond Paper

The origins of 3D printing extend back to the mid-1980s, when Carl Deckard and Joseph Beaman at the University of Texas at Austin invented a process called Selective Laser Sintering (SLS). The process fuses materials—metals, plastics, ceramics, and more—into a three-dimensional mass. In 1984, inventor Charles Hull introduced what is generally considered the first genuine 3D printer. It used stereolithography—also known as optical fabrication—to introduce the idea of rapid prototyping. By 1995, researchers at the Massachusetts Institute of Technology (MIT) had built

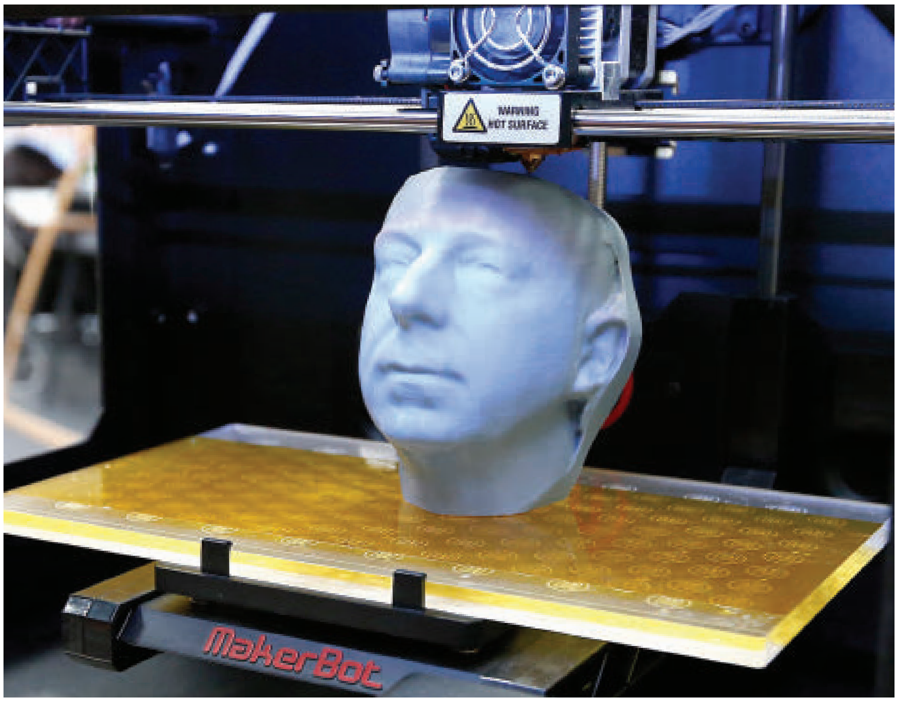
The Steampunk 3D guitar, inspired by the classic Fender Telecaster, features a 3D printed body with moving gears and piston. The entire body of the instrument, including all its moving parts, is printed as a single component, with no further assembly required.

PHOTOGRAPH COURTESY OF OLAF DIEGEL/ODD GUITARS a modified inkjet printer that could extrude a binding solution onto a bed of powder. The device spawned actual 3D printing companies.

The technology has continued to march forward; printheads on devices can now move both horizontally and vertically to create nearly unlimited design possibilities. They can fabricate multiple moving parts with a single movement, and even apply different materials to different parts of an object. This high level of flexibility allows designers to experiment with new ideas that would be cost-prohibitive using conventional manufacturing methods.

Today, commercial 3D printers are widely available at a cost of tens of thousands of dollars—while consumer versions hover just above $1,000. Designers and manufacturers are turning to these devices for rapid prototypes, models, and specialty manufacturing. These systems rely on a computer-aided design (CAD) file to tell the printer what to fabricate. In some cases it will be an entire item, and in others, it will be a product requiring some assembly that will be fabricated component by component. Depending on the complexity of the item, the production process can vary from minutes to hours.

At the University of Washington, for example, researchers have printed bow ties, a composting toilet, and a usable boat. So far, they have fabricated items from about 50 different materials, including glass, ceramics, metals, plastic, chocolate, and rice.



3D printers work with a variety of materials and can produce life-like models from 3D scans.

"It is possible to print using virtually any material," explains Mark Ganter, a professor of mechanical engineering at the University of Washington. In fact, 3D printing technology—technically referred to as "additive manufacturing"—is advancing rapidly. Some printers now support multiple materials—thus enabling far more sophisticated designs.

The ability to print actual devices using multi-depositional processes has caught Leigh's attention. He and fellow researchers have developed a new material, nicknamed "carbomorph," which lays down electronic tracks and sensors within a 3D structure. In fact, this process enables touch-sensitive areas that can be added to a basic electronic circuit board. The next step, he says, is to add cables and wires that would make the object a fully functional device, such as a television remote control, mobile phone, or a game controller.

Leigh says metals and plastics—the conventional materials used in 3D printing—are giving way to more sophisticated composites that allow far more advanced manufacturing methods, along with more sophisticated products. Lipson says that 3D software

Research

HiPEAC Debuts Roadmap for CS Research in EU

The European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC) recently introduced its newest roadmap for the future of computer systems. The blueprint presented in "HiPEAC Vision for Advanced Computing in Horizon 2020" comprises a set of analyses, challenges, and recommendations for Horizon 2020, the European Commission's new program for research and innovation, slated to begin next year.

The 48-page report issued by HiPEAC, the largest CS network in Europe representing over 1,200 researchers, identifies three strategic areas that are key to the future of CS research in Europe: mobile, embedded, and data center computing. It also points to the challenges foreseen in the areas of energy consumption, system complexity, and dependability, and presents a frank review of Europe's strengths and weaknesses in the global CS community. According to the document, the continent must balance its strong embedded ecosystem, available public funding for R&D, and its respected higher education institutions with the real challenges and threats posed by the current financial crisis, loss of competitiveness in certain domains, and fierce competition from external, inexpensive structures.

Koen De Bosschere, coordinator of the HiPEAC network, calls the report one of the most important documents steering EU research funding in advanced computing systems in Europe. "It is very important for the ACM Europe community and relevant for the global ACM community," he points out.

To view the report in its entirety, see http://www.hipeac.net/system/files/hipeac_roadmap1_0.pdf.

PHOTOGRAPH BY FABRIZIO BENSCH/REUTERS/CORBIS is maturing and a growing pool of designs is available. At the same time, machine learning is making it easier to predict how different materials will interact, and how they can be used together to create more advanced materials, shapes, and functions.

Cornell University has developed, among other things, toys, edible printed food items, and robotic insects using 3D printing technology. The latter devices—made of polyethylene film and weighing in at 3.89 grams—can hover on their own for 85 seconds. Lipson, who introduced an open source 3D printing initiative called Fab@Home in 1995 (it has emerged as a primary source for 3D printing designs), is continuing to push the technology forward. "Most things that are manufactured can be printed with 3D technology," he says.

In fact, 3D printing could turn the physics of manufacturing upside down. "Fabricating a complex object is no more expensive or time-consuming than printing a simple object," Lipson says. "This is a huge departure from the historical model of manufacturing—where more difficult designs require more skill, time, and labor." He adds that 3D printers could further redefine manufacturing, if software compilers are available to transform a concept into a technical design (like a shelf bracket or a coffee mug); a person would simply state what he or she wants, and the computer would handle the process. This would eliminate the need for programming and engineering skills to create designs, he says.

Navigating a New Frontier

Major manufacturers are adopting 3D printing technology at a rapid clip. Airbus, Ford Motor Company, Adidas, Logitech and scores of other companies are turning to industrial 3D printing for rapid prototyping, as well as for the actual production of products. For instance, airplane manufacturer Airbus uses industrial 3D printing to produce certain cabin components.

Lower-cost 3D consumer printers are appearing in homes, allowing individuals to produce plates, drinking glasses, toys, lamps, jewelry, clothing accessories, and other items on demand. Individuals now share designs for 3D printing at sites such as Thingiverse and Fab@Home.

Many describe 3D printing as highly disruptive, because it could displace a large number of jobs by eliminating much of the distribution system and business infrastructure that exists today.

The technology is not without concerns, however. Many describe 3D printing as highly disruptive, because it could displace a large number of jobs by eliminating much of the distribution system and business infrastructure that exists today. A more immediate concern is that the technology makes it relatively easy to fabricate counterfeit products rapidly, and to manufacture illegal items such as weapons and drug paraphernalia. "It's an issue that could make the controversy over illegal music downloads pale by comparison," Lipson warns.

Meanwhile, researchers are continuing to push the envelope on 3D printing. Scientists have begun to experiment with printing artificial organs, and have successfully produced a liver, bone, cartilage, and blood vessels. Within a decade or two, it will likely be possible to print usable human tissue, including a heart, from cartridges of human cells. The appeal of 3D bio-printing is the array of tissue and organs that could be constructed using a single printer.

The University of Washington's Ganter believes 3D printing ultimately will offer enormous benefits, including the ability to use common materials—dirt, salt and other widely available materials—to fabricate much-needed objects inexpensively in developing nations. 3D printers also could make it easier to provide relief at a disaster site by fabricating temporary housing and various necessities. Finally, the technology could alter the waste stream and provide substantial environmental benefits, he notes. "Imagine printing objects from recycled plastic and other materials rather than using wood and other raw materials," Ganter says.

Make no mistake, 3D printing has come of age. As the underlying software becomes increasingly sophisticated and more-complex printer designs move into the mainstream, a new era of fabrication and manufacturing will emerge, one that offers lower barriers to entry and delivers a high level of customization. Concludes Lipson: "The revolution in 3D printing is already taking place. Within a decade, the average person will have a 3D printer in their house and at their business. The technology will become a core part of our lives."

Further Reading

Campbell, T.A., Ivanova, O.S.
3D printing of multifunctional nanocomposites, Institute for Critical Technology and Applied Science (ICTAS), Virginia Tech; http://www.sciencedirect.com/science/article/pii/S1748013212001399

Willis, K., Brockmeyer, E., Hudson, S., Poupyrev, I.
Printed optics: 3D printing of embedded optical elements for interactive devices, Proceedings of the 25th annual ACM symposium on User interface software and technology, Pages 589-598, ISBN: 978-1-4503-1580-7; http://dl.acm.org/citation.cfm?id=2380190

Leigh, S.J., Bradley, R.J., Pursell, C.P., Billson, D.R., Hutchins, D.A.
A Simple, Low-Cost Conductive Composite Material for 3D Printing of Electronic Sensors, 2012, PLoS ONE 7(11): e49365. doi:10.1371/journal.pone.0049365; http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0049365

Lipson, H., Kurman, M.
***Fabricated: The New World of 3D Printing*, John Wiley & Sons, 2013.**

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

Society | DOI:10.1145/2483852.2483860 Neil Savage

Backing Creativity

Hacker spaces are spreading around the world, though some government funding is raising questions.

HACKER SPACES, COMMUNAL areas where people use shared tools to build whatever toy, craft, or gadget strikes their fancy, have been growing in popularity around the world. The hackerspaces.org wiki listed more than 1,400 such locations as of mid-May. Governments from the U.S. to China are providing funding to promote their growth, and that has provoked discussion over the evolving nature of the spaces and how they should be conceived.

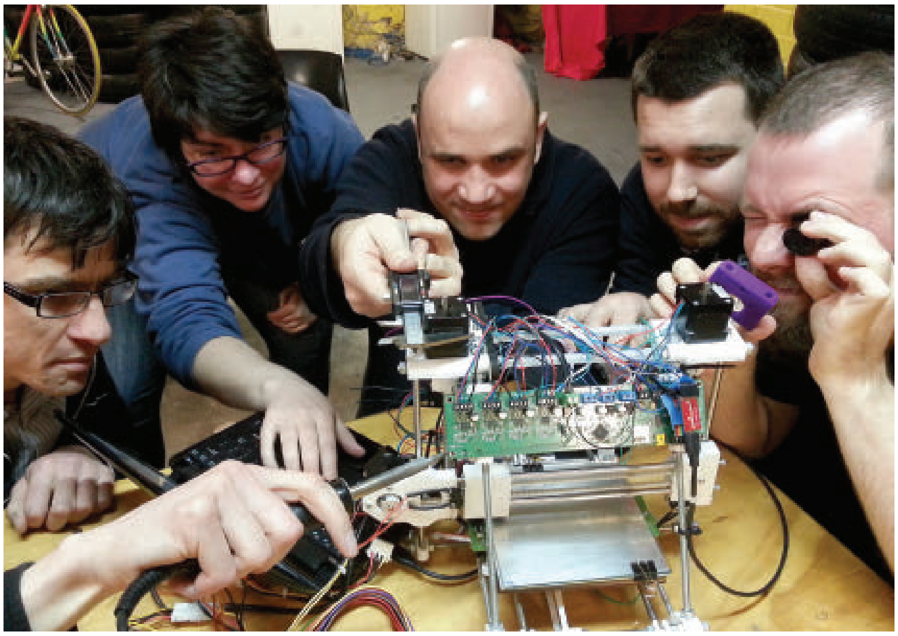
The spaces go by various names —hacker spaces, maker spaces, tech shops, fab labs—but generally consist of a collection of manufacturing equipment such as laser cutters, 3D printers, and computer numerical control (CNC) systems that perform computer-aided manufacturing, as well as more old-fashioned tools such as oscilloscopes and welding equipment, plus raw materials. They generally offer courses on how to use the equipment, and provide staff to help. Users pay a membership fee or a single-use fee to get access to the space and the equipment. Projects can range from arts and crafts to electronic gadgetry to robots, all dictated by the whims of the users.

Dale Dougherty, the founding editor and publisher of *Make* magazine and co-founder, in 2006, of the Maker Faire, which brings engineers and artists to show off their creations, makes distinctions between the different types of spaces.

Fab labs, he says, tend to be associated with universities and have academic aims; they were originally created at the Massachusetts Institute of Technology's Media Lab.

Tech shops may be more commercial, while maker spaces are open to do-it-yourselfers of varied backgrounds and ages.

Hackers—computing and engineering types who like to explore how things work and try to make them better—are only a subset of the maker movement, Dougherty says. "There's a lot broader range of participation than just hackers," he says.


Makers collaborate on a project in a hacker space.

PHOTOGRAPH COURTESY OF HACKERSPACE ADELAIDE, HACKADL.ORG

Dougherty makes the analogy with public gyms, spaces that provide shared exercise equipment. Once they were limited to bodybuilders, who organized them for their own purposes; now they attract members of the general public, who have various workout goals and fitness levels, but who all have a common aim of getting exercise.

Yet with evolution and increased interest can come controversy. Early last year, Dougherty, then with O'Reilly Media's Make Division, won a grant from the U.S. Defense Advanced Research Projects Agency (DARPA) to fund its Makerspace project, developed along with Otherlab, a research and development company in San Francisco. The goal of the program, funded by DARPA's Manufacturing Experimentation and Outreach (MENTOR) program, is to introduce high school students to various manufacturing tools they can use to build projects from mobile robots to go-karts. DARPA plans to provide $10 million to 1,000 schools over three years to promote science, technology, engineering, and mathematics (STEM) education.

O'Reilly's involvement led some hackers to boycott the Maker Faire. One of the chief critics is Mitch Altman, co-founder of the San Francisco hacker space Noisebridge, who worries that DARPA will steer work in spaces it funds toward military applications, uses he does not want his efforts supporting. "I see the military as going out and bombing and killing, not for defending our country but for other purposes," he says.

Even though the U.S. Defense Department says the DARPA funding comes with no strings, Altman does not believe that. "When a politician accepts gifts from individuals and corporations, it's very likely going to be perceived that they're taking a bribe and they're going to have to pay it back," he says. "It does change what we're willing to do and not do for the sake of the money."

Tim O'Reilly, founder of O'Reilly

Media and a supporter of the open source movement, thinks those who believe they can avoid entanglements with military funders are being naïve. After all, the Internet originated with DARPA's predecessor, the Advanced Research Projects Agency, and global positioning satellites were initially launched for military uses. O'Reilly says those who so vocally oppose contributions that are in any way Defense-related still are "perfectly happy to use the GPS in their phone for things like maps and Foursquare." He thinks having more funding, and reaching more people, will only benefit the maker movement. "If anything, the government money will help create more do-it-yourself opportunities."

Altman responds that, although the government may have created the Internet for military purposes, it was hackers who turned it into something that could benefit society. "Once it existed, hackers took it and improved upon it and shared it," he says. "That's what hackers do. That's the whole ethos."

He does not oppose *all* government funding; Altman would accept a Department of Education grant if it aligned with his values and did not come with constraints he disliked. He respects Dougherty and others who disagree with his stance, but he hopes that people take ethics into consideration when accepting funding, whatever the source. "If people look at it differently, hopefully they're looking at it consciously."

High Ideals

One area where O'Reilly and Altman agree is regarding the potential for the hacker ethos to be a force for good. O'Reilly, for instance, recently contributed to a Kickstarter campaign to start community hacker spaces in Baghdad, Iraq. Aside from helping to spur individual creativity, such spaces may help the Iraqi people see Americans as something other than invaders, he says. "I'd like to see the spirit of hackerdom improve peace in the Middle East," says O'Reilly.

Indeed, the maker movement is spreading around the globe, often with support from governments. The city of Shanghai, for instance, announced in 2011 it would provide funding for 100 spaces it called "innovation houses." A maker fair was held in April 2012 in the Chinese city of Shenzhen, and there are plans for a Beijing fair later this year, says Benjamin Koo, associate professor of industrial engineering at Tsinghua University in Beijing.

Koo ran a hackerthon at the university in January. Over the course of 80 straight hours, the 200 students participating had to react to a fictional disaster on an island, building a system that would deliver support to the island. The idea of such an "extreme learning process," he says, is to force students, who in China are used to dutifully taking in what is presented to them in class, to move out of their comfort zone and discover their capabilities. "For the first time in their lives, they became masters of their own time," Koo says. His hope is to change the way universities and students in China—and elsewhere in the world—approach education, and it is a change the government supports, hoping to modernize China's economic role, he says.

"China is obviously capable of producing goods, but is really set up in a mentality similar to the 1800s and 1900s, that humans are basically instruments making these products," Koo says. He hopes the maker model can help transform the Chinese into inventors and entrepreneurs.

Crossing Continents

International cooperation is a theme among makers. Koo is applying for grants that would allow some of Altman's colleagues to live and work at Tsinghua for two to three months in a hackers-in-residence program. He is also the founder of the Toyhouse, a hacker space that is trying to help educators, technologists, entrepreneurs, artists, and government agencies share knowledge and ideas. "In order to have hacker spaces run successfully, you need to have official recognition as well as very professional organizers to run this," Koo says.

The Santiago Makerspace in the capital city of Chile runs on corporate sponsorship from companies such as Microsoft and Nextel, membership fees, and roughly $250,000 from CORFU, the Chilean Economic Development Agency. Tiburcio de la Carcova, who runs the maker space, says part of the government funding is for the space itself, while part will go to support a maker fair later this year.

He organized Latin America's first such fair, which ran in Santiago last December and drew more than 5,000 people. Many of the participants at the fair, and many of the people who take courses and do projects at Santiago Makerspace, are children ranging from elementary school age to teenagers. The hands-on, playful quality of the maker movement is helping to educate youngsters in electronics and engineering, among other disciplines, in ways traditional schooling has not, de la Carcova says.

The people involved in Santiago Makerspace did discuss what effect the funding sources would have on their mission. "That was one of the concerns—if we get the money from the sponsors, are we going to change?" he says. However, he explains, the government does not dictate their projects, and the members of the maker space are committed to their personal values. Those include openness—the group shies away from patents as potentially stifling innovation—and sustainability, with a strong commitment to recycling.

"I don't think that funding, if you have a clear vision, can dictate what you will do," de la Carcova says. C

Further Reading

Lindtner, S. and Li, D
Created in China: the makings of China's hackerspace community, ACM Interactions 19, 6 November 2012.

O'Leary, A.
Worries Over Defense Department Money for Hackerspaces, New York Times, Oct. 5, 2012.

Tweeny, D.
DIY Freaks Flock to "Hacker Spaces" Worldwide, Wired, March 29, 2009.

Merlo, S.
China's First Maker Faire This Sunday in Shenzhen, Make, April 5, 2012.

Mota, C.
What We Can Learn from Hackerspaces, TedxTalks, http://www.youtube.com/watch?v=2c_8tsbRF8g

Santiago Mini Maker Faire Day 1 http://vimeo.com/55749953

Neil Savage is a science and technology writer based in Lowell, MA.

Association for Computing Machinery

Global Reach for Global Opportunities in Computing

Dear Colleague,

Today's computing professionals are at the forefront of the technologies that drive innovation across diverse disciplines and international boundaries with increasing speed. In this environment, ACM offers advantages to computing researchers, practitioners, educators and students who are committed to self-improvement and success in their chosen fields.

ACM members benefit from a broad spectrum of state-of-the-art resources. From Special Interest Group conferences to world-class publications and peer-reviewed journals, from online lifelong learning resources to mentoring opportunities, from recognition programs to leadership opportunities, ACM helps computing professionals stay connected with academic research, emerging trends, and the technology trailblazers who are leading the way. These benefits include:

Timely access to relevant information

- ***Communications of the ACM*** magazine
- ***ACM Queue*** website for practitioners
- Option to subscribe to the ***ACM Digital Library***
- ACM's ***50+ journals and magazines*** at member-only rates
- ***TechNews***, tri-weekly email digest
- ***ACM SIG conference*** proceedings and discounts

Resources to enhance your career

- **ACM Tech Packs**, exclusive annotated reading lists compiled by experts
- **Learning Center** books, courses, webinars and resources for lifelong learning
- Option to join **36 Special Interest Groups** (SIGs) and **hundreds of local chapters**
- **ACM Career & Job Center** for career-enhancing benefits
- ***CareerNews***, email digest
- **Recognition of achievement** through Fellows and Distinguished Member Programs

As an ACM member, you gain access to ACM's worldwide network of more than 100,000 members from nearly 200 countries. ACM's global reach includes councils in Europe, India, and China to expand high-quality member activities and initiatives. By participating in ACM's multi-faceted global resources, you have the opportunity to develop friendships and relationships with colleagues and mentors that can advance your knowledge and skills in unforeseen ways.

ACM welcomes computing professionals and students from all backgrounds, interests, and pursuits. Please take a moment to consider the value of an ACM membership for your career and for your future in the dynamic computing profession.

Sincerely,

[illegible]

Vint Cerf

President
Association for Computing Machinery

Association for Computing Machinery

Advancing Computing as a Science & Profession

Association for Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: AD13

You can join ACM in several easy ways:

Online	Phone	Fax
http://www.acm.org/join	*+1-800-342-6626 (US & Canada)* *+1-212-626-0500 (Global)*	*+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/about/code-of-ethics

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 professional 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

DOI:10.1145/2483852.2483861

Pamela Samuelson

Legally Speaking

Statutory Damages As a Threat to Innovation

Considering the negative influence of U.S. statutory damage rules on technology innovation.

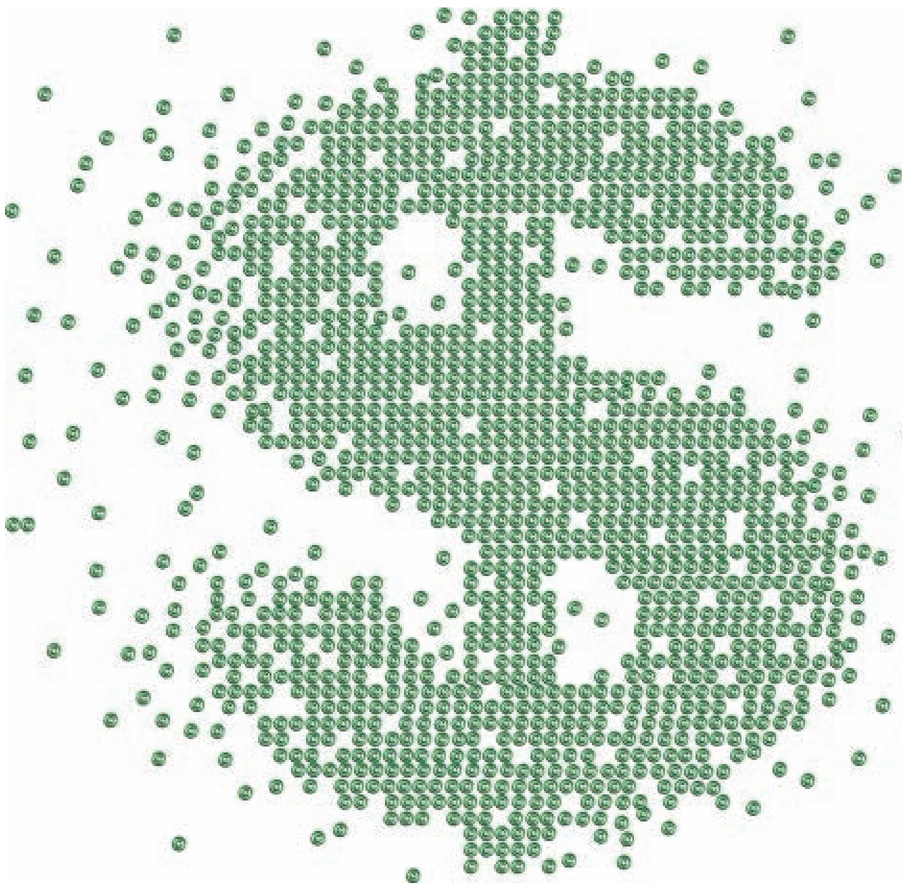


ILLUSTRATION BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

MANY *COMMUNICATIONS* READERS might have heard of the $1.92 million jury award against Jammie Thomas-Rasset for downloading 24 songs through a peer-to-peer file-sharing system. This was obviously way in excess of the actual harm to copyright markets caused by this downloading. However, U.S. law allows copyright owners to ask for statutory damages up to $150,000 per infringed work. This absurdly large award was more of a symbolic than an actual victory for the record labels who sued Thomas-Rasset, as she cannot possibly pay such a sum.

How much money should record labels be able to get against the makers of peer-to-peer file-sharing platforms who are found contributorily liable for infringements like Thomas-Rasset? Record label plaintiffs asked a federal judge for an award of $75 trillion in statutory damages against Limewire after the judge found it to be contributory infringer. (To put this number in perspective, consider that the gross domestic product of the U.S. economy is about $14 trillion.) Although the trial judge considered the record labels' demand to be "absurd," she nonetheless was contemplating an award in the low billions.

Perhaps few will sympathize with Limewire or Thomas-Rasset. But we should care about the unbridled nature of statutory damages that put technology companies at risk whenever they introduce a product or service into the U.S. market that enables new ways to access or use copyrighted works. A recent empirical study asserts that copyright statutory damages poses a real threat to technology innovation in the U.S.[2] This column will explain why.

Why Statutory Damages?

Every country with a copyright law allows rights holders to sue infringers for compensation for harms caused by infringements (for example, a lost license fee or sales diverted to the defendant). Many countries also allow copyright owners to recover that part of the defendant's profits attributable to infringement.

In the U.S. and a distinct minority of other countries, copyright owners can ask for an award of statutory damages instead of lost profits or defendant profits from infringement. The original justification for U.S. statutory damages was to enable copyright owners to get some compensation when it was too difficult or expensive to prove harm resulting from infringement. A later justification was to enable copyright owners to vindicate their rights when litigation costs would swamp the amount that a copyright owner could recover for small-scale infringements. Statutory damages have also come to serve as a deterrent to infringements.

The normal range for U.S. statutory damage awards is between $750 and $30,000 per infringed work. U.S. courts have no discretion to award less than $750 per work unless the defendant can prove she was an innocent infringer. If the infringement is deemed willful, the upper bound is $150,000 per infringed work. Courts sometimes say a statutory damage award need not bear any relationship to injuries actual suffered.

When the U.S. Congress enacted the current copyright statute in 1976, it had no idea these rules would put technology developers at risk of outrageously large awards. Congress thought of "per work" awards would prevent excessive liability, not cause it.[3] However, digital technologies that enable users to access or use copyrighted works almost inevitably involve large numbers of works. Hence, the potential for statutory damage liability against their developers can be staggeringly large.

When the U.S. Congress enacted the current copyright statute in 1976, it had no idea these rules would put technology developers at risk of outrageously large awards.

MP3.com

Exemplifying the potential for excessive statutory damages awards against technology developers was a lawsuit that several record labels brought against MP3.com in the late 1990s. MP3.com bought 4,700 CDs of recorded music, ripped the music from the CDs, and loaded the music into a database for its new "beam-it" music service. The idea was to enable individuals who owned CDs to listen to their music through MP3.com anywhere in the world on any Internet-connected device.

The record labels concentrated their infringement case on the copies made by MP3.com in ripping the music from the CDs to create the database. MP3.com claimed the new service was lawful because it enabled its customers to make space-shift uses of their purchased music and it had only made intermediate copies of music it owned.

A trial judge ruled in favor of the labels and announced that unless MP3.com settled the case right away, he would award $25,000 per infringed CD in statutory damages, for a total of $118 million of liability. This was far in excess of the actual harm from ripping the music. Nor were there any profits to speak of, given that MP3.com had suspended the service once it was sued.

Faced with this calamitous impending award, MP3.com settled for $53.4 million. The judge made clear he intended for this very large award to deter other technology companies from succumbing to the temptation to offer similarly risky services.

Cablevision and Aereo

In 2006, Cablevision introduced a Remote Storage DVR service so that its subscribers could watch programs at a different time by accessing the programs on Cablevision servers. A consortium of television and filmmakers sued Cablevision for infringement because it had designed the system to make copies on its servers as part of the RS-DVR feature and because it transmitted the programs to subscribers, which the plaintiffs alleged were infringing public performances.

Cablevision pointed out that its RS-DVR technology enabled time-shift copying like that which the U.S. Supreme Court ruled was fair use in a 1984 case challenging the Sony Betamax video-tape recording (VTR) device. The plaintiffs tried to distinguish Cablevision's RS-DVR from the Sony VTR by pointing out that Cablevision time-shift copies were made on and transmitted through Cablevision servers.

Cablevision lost at the trial court level, but managed to win a non-infringement ruling at the appellate court level. Given the large number of time-shift copies of programs that could be made through its RS-DVR, Cablevision was at some risk of a very large statutory damage award for introducing this new service to its subscribers.

Aereo is a service that picks up broadcast signals and makes television programs available to its subscribers through Internet-connected devices. It offers day passes, but also monthly or yearly subscriptions for access to broadcast programs for viewing on the subscriber's computer or mobile devices.

ABC, among others, objected to this service. It sued Aereo for copyright infringement. Because Aereo makes a large number of television programs available, a statutory damage award against it would likely be very substantial. Although ABC was unable to get a preliminary injunction to stop Aereo from streaming broadcast content to subscribers, it has appealed this loss. A California court has ruled in favor of broadcasters in a similar lawsuit.

Google

Google is at risk of an outrageously massive statutory damage award in a case charging it with infringement for scanning in-copyright books from major research library collections. The Authors Guild's class action lawsuit seeks statutory damages for the approximately 15 million in-copyright books alleged to have been infringed.

Although Google has a very plausible (and to me convincing) fair use defense for indexing book contents and making snippets available and although authors have suffered no actual damages from Google's use of in-copyright books, Google risks, by one estimate, as much as $3.6 trillion in statutory damages if it loses its fair use defense. Even capped at the statutory damage minimum, liability could be in the billions of dollars.

Google also faces billions in possible statutory damage liability in a still-pending lawsuit brought by Viacom against YouTube for clips of Viacom programs that were uploaded by fans to YouTube's servers.

Threats to Innovation

These examples illustrate one threat that copyright statutory damages pose to innovative technology developers. Executives must be prepared to bet the company's future on a technology design that copyright industry players may find threatening. Some who have taken the bet have found themselves sued individually and at risk of losing their houses and savings. Venture capitalists are often wary of investing in these kinds of risky technologies.[2]

But consider also the innovations that are not undertaken or abandoned because of the risk of high statutory damage awards. This is not just a hypothetical problem. Based on qualitative interviews with two dozen CEOs or founders of tech companies, Michael Carrier concludes that statutory damages have a chilling effect on the development of technologies that enable access to or new uses of digital content.[2]

An International View

The risk of excessive statutory damage awards against technology developers is, oddly enough, far lower outside the U.S. Less than 14% of countries in the world have statutory damage regimes.[4]

Although the U.S. copyright statute directs courts to craft statutory damage awards that are "just," the courts have failed to develop any guidelines to make them so.

Most of these nations are developing or emerging economies, such as Azerbaijan, Belarus, Bulgaria, Costa Rica, Dominican Republic, Kazakhstan, Kyrgyzstan, Liberia, Malaysia, Moldova, Ukraine, and Vietnam. These countries typically adopted statutory damages under pressure from the U.S. as part of bilateral free trade agreements.

Most developed countries with strong copyright industries, such as France, Germany, Japan, the Netherlands, the United Kingdom, and Sweden, do not have statutory damage regimes. Copyright plaintiffs in these countries can only seek compensation for harms suffered because of infringement and sometimes an accounting for defendant profits from infringement.

Of the relatively few developed countries that do have statutory damage regimes—which include Canada, Israel, and South Korea—there are limits on such awards not found in U.S. law. Canada, for instance, limits the maximum award for noncommercial infringements (for example, individual file sharing) to $5,000. Canadian copyright law also directs courts to consider the proportionality of a statutory damage award as compared with actual damages, with no more than a 10 times multiple over actual damages being awardable.[4]

Conclusion

Although the U.S. copyright statute directs courts to craft statutory damage awards that are "just," the courts have failed to develop any guidelines to make them so.[3] Legal commentators believe the U.S. statutory damage regime is in need of significant reform.[1,3] Even the Register of Copyrights has come out in favor of some reform of statutory damage rules.

There are two ways that U.S. statutory damage rules can be reformed. One is for Congress to enact some limits on these rules such as those adopted in Canada. The other is through the courts.[3] The U.S. Supreme Court could, for instance, have taken Thomas-Rasset's appeal of the excessive statutory damage award against her for file sharing. The Court has previously struck down punitive damage awards as violating the constitutional principle of due process of law. It has directed courts to take into account the relative magnitude of the wrong committed and the relative proportionality of a punitive award as compared with actual damages suffered.

Extending this approach to strike down excessive awards in copyright cases would make U.S. law more just and make it more consistent also with the copyright laws of other nations. It would also lessen the risk of exorbitant awards against technology developers.

It is certainly true that many innovative technologies are being developed in the U.S. Obviously some entrepreneurs are still willing to take risks. But that does not mean we should not worry about the potential harm to innovation posed by statutory damage rules. There would likely be even more innovation if these rules were reformed to make them more just. C

References

1. Berg, S. Remedying the statutory damages remedy for secondary copyright infringement liability: Balancing copyright and innovation in the digital age. *Journal of the Copyright Society of the U.S.A.* 56:265 (2008–2009).
2. Carrier, M. *Copyright and innovation: The untold story.* Wisc. L. Rev. 891 (2012).
3. Samuelson, P. and Wheatland, T. Statutory damages in copyright law: A remedy in need of reform. *William & Mary L. Rev. 51:2*, 439 (2009).
4. Samuelson, P., Hill, P., and Wheatland, T. Statutory damages: A rarity in copyright laws internationally—But for how long? *Journal of the Copyright Society of the U.S.A.*, forthcoming.

Pamela Samuelson (pam@law.berkeley.edu) is the Richard M. Sherman Distinguished Professor of Law and Information at the University of California, Berkeley.

DOI:10.1145/2483852.2483862

Christopher S. Tang and Joshua Zimmerman

Computing Ethics
Information and Communication Technology for Managing Supply Chain Risks

How to encourage ethical behavior among all links in a global supply chain.

Over the last two decades, many firms implemented various supply chain initiatives to: increase revenue, for example, more product variety, more-frequent new product introductions, more sales channels/markets in new markets; reduce cost, for example, supply base reduction, online sourcing, and offshore manufacturing; and reduce assets, for example, outsourcing of manufacturing, information technology, logistics, and even product design. Supply chains today are more "efficient" but also much more complex than those in the past. As a consequence, they are more vulnerable to disruptions and delays. In some cases, unethical behavior of one supplier along the supply chain can cause serious security risks and even deaths. Here are some examples:

Communication and coordination risk. Complex global supply chains often create major communication and coordination problems. For example, Boeing outsourced the design and development of many critical sections of its 787 aircraft to tier-1 suppliers. However, these tier-1 suppliers subcontract various modules to tier-2 suppliers, who in turn outsource certain components to tier-3 suppliers. With a multi-tier supply chain that has at least 500 suppliers located in over 10 countries, the communication and coordination between Boeing and those nondirect suppliers regarding various product development activities were essentially nonexistent. This is one of the reasons why Boeing's 787 was launched 3.5 years late and the development cost was $6 billion over budget.[5] Further, the lack of communication and coordination is thought to have caused the battery problems that grounded the fleet for a period of time.

Supply chain coordination and communication problems contributed to delays and cost overruns affecting Boeing's 787 aircraft.

PHOTOGRAPH BY THOR JORGEN UDVANG

Information security. In a multi-tier supply chain, it takes only one unethical supplier to create an IT security breach. In 2011, the U.S. Senate Armed Services identified over 1,800 instances of suspected counterfeit electronics from China that were installed in military systems made by Raytheon, L-3 Communications, and

Boeing. Some counterfeit electronic parts can have malware preinstalled in the motherboards, making it extremely difficult to trace and putting U.S. troops at risk.[3] In the same vein, when firms (banks, hospitals, and insurance companies) procure their computers and servers with counterfeit electronics, they are exposed to the risk of leaking credit card data, personal data, and intellectual property information to cyber criminals.

Food safety. In Europe, the horsemeat scandal in 2013 heightened the concerns over food supply chain security. While the use of cheaper substitutes to produce food products is unethical, some suppliers would even use harmful or banned substances to produce food products to increase profits. As U.S. agribusinesses and food manufacturers import more food products from China, public concerns arose in 2012 after a series of reports about unsafe food products made in China, for example: melamine adulteration in pet food, hydrolyzed leather protein in milk, and chromium in pill capsules.[2]

Drug safety. Opaque supply chain operations in the pharmaceutical industry and online pharmacies create opportunities for unethical suppliers to produce adulterated or counterfeit drugs. According to the National Association of Boards of Pharmacy, counterfeit drugs generated over $75 billion in revenue in 2010 and caused over 100,000 deaths worldwide.[4] For example, Baxter's contract manufacturer Changzhou SPL-CZ classified itself as a chemical plant so that it was not under the scrutiny of Chinese State Food and Drug Administration. In 2007, SPL-CZ used chondroitin sulfate (a cheap and unsafe substance) to produce heparin active pharmaceutical ingredients (API) for Baxter and 14 other U.S. companies. This adulterated API was then used to produce the blood-thinning drug Heparin that resulted in several deaths and major recalls in 2008.

ICT for Mitigating Supply Chain Risks

As firms are confronting different types of supply chain risks, information and communication technology (ICT) can enable firms to access relevant and timely information so they can improve communication and coordination as well as to screen, monitor, and enforce ethical behavior of all suppliers along the supply chain. Some examples include:

Information technology for improving supply chain visibility and coordination. Information technology can enable firms to improve supply chain visibility and coordination. For example, Agilent, a premier electronics measurement company, implemented an IT system developed by Kinaxis in 2011 that would allow Agilent to "bridge" different Enterprise Resource Planning (ERP) systems of all suppliers so that Agilent has end-to-end supply chain visibility for real-time communication and coordination. By operating as a "virtually integrated" supply chain, Agilent can respond to market changes in a cost-effective and time-efficient manner.

Information technology for improving track and trace capabilities. To manage, monitor, and secure container shipments, Savi and Qualcomm co-developed an integrated service to provide real-time track and trace capability: Savi uses RFID tags, readers, software, and electronic seals placed on cargo containers, while Qualcomm uses satellites to track and trace the container movement and the seal status information conveyed by the RFID readers and the electronic seals. In Europe, the pharmaceutical industry started a new initiative in 2010 that is intended to fight the trade of counterfeit drugs. Under this initiative, pharmaceutical manufacturers add a serialization code (a unique machine readable "passport" number) to each pack of medicine before distribution, where this passport number is scanned at each link of the supply chain all the way to the pharmacist. By using direct communication links among the pharmacy, the pharmaceutical manufacturer, and the regulatory body, the pharmacist can authenticate the drug, the pharmaceutical firm can know if its drug has been counterfeited, and the regulator can take appropriate actions (for example, product recalls) immediately. In Africa, HP and the African social enterprise network mPedigree teamed up in 2012 to develop a drug authentication effort to combat counterfeit drugs in 2012. Under this effort, pharmaceutical manufacturers add a "scratch-off label" containing a verification code before distribution. When a customer buys the medicine at the pharmacy, the customer scratches the label to receive the code, and sends a Short Message Service (SMS) text message to the pharmaceutical manufacturer via mobile phone so the manufacturer can verify the drug's authenticity.

In a multi-tier supply chain, it takes only one unethical supplier to create an IT security breach.

Information technology for exposing unethical supplier behavior. Recognizing information access as an important first step to creating public exposure, voluntary groups of activists and NGOs are using the Internet to provide more timely and accurate information about food safety issues in China. In 2012, a group of Chinese volunteers developed websites (http://www.zccw.info), and smartphone apps (http://tinyurl.com/72orssq) to report the latest Chinese food safety scandals (location, food categories, safety issues, vendor identity, supplier identity, and so forth). By providing fast and accurate information about food safety issues on a single website or an app, public exposure creates incentives for vendors to be more vigilant and suppliers to think twice before committing product adulteration. Pushing this line of thinking further, manufacturers may consider partnering with NGOs to expose the identity of unethical suppliers who produce adulterated products on the Internet and mobile phone apps to create the threat of public humiliation for suppliers who produce adulterated products.[2]

Management Tools for Mitigating Supply Chain Risks

While ICT is an enabler for firms to

improve supply chain visibility, supply chain coordination, and supply chain security, many firms should consider the following management tools.

Know your suppliers and their suppliers. Many firms do not fully vet their supplier's capabilities and their ethical reputation. While ISO 26000 social responsibility standards provide guidelines for establishing ethical standards, a firm needs to conduct audits to ensure these guidelines are being followed by all partners along the supply chain. For example, to secure the government's technology supply chain, the U.S. Department of Defense (DoD) conducts audits and requires certification of all suppliers along the supply chain—not just its direct vendors. Because it is inefficient for Western firms to conduct frequent audits at various suppliers located in emerging markets, Western firms should form partnerships with various NGOs with local presence who can conduct those audits on a firm's behalf or provide information about the supplier's reputation to ensure ethical standards are maintained.

Do not focus only on cost and speed when selecting suppliers. Because most firms tend to award contracts to suppliers with the lowest bid, many suppliers may commit unethical acts in order to ensure profitability after winning a contract with the lowest bid. To break this vicious cycle, a firm should develop a more balanced view and consider different attributes than simply cost and speed when selecting suppliers and the firm should communicate this holistic measure with potential suppliers.

Provide incentives to discourage unethical behavior. To entice suppliers to maintain certain mutually agreeable ethical standards, firms should develop incentives to discourage unethical behavior. For example, the DoD would penalize their vendors for not buying components from pre-approved suppliers or for delivering computer equipment with counterfeit components. Babich and Tang[1] showed that firms can deter suppliers from product adulteration if they defer their payment as follows: pay an upfront deposit to initiate the production, withhold the contingent payment, and release this contingent payment only if no adulteration is discovered by the customers or government agencies over a pre-specified duration. The logic behind this deferred payment mechanism is simple: it allows manufacturers to have more time to learn about suppliers' product quality by withholding contingent payments.

Information technology can enable firms to improve supply chain visibility and coordination.

Conclusion

While many Western firms have ethical sourcing guidelines and their own internal ethical standards, they should not expect all partners along the global supply chain to share the same values and impose the same standards due to different regulations, different environments, and different cultures. To encourage ethical behavior of all parties along the supply chain, ICT is an enabler and the aforementioned management tools are the enhancers. Western firms should rethink the way they manage the whole supplier chain and not just focus on their direct links. C

References

1. Babich, V. and Tang, C.S. Managing opportunistic supplier product adulteration: Deferred payments, inspection, and combined mechanism. *Manufacturing, Service and Operations Management 14*, 2 (2012).
2. Babich, V. and Tang, C.S. Using Social and Economic Incentives to Discourage Chinese Suppliers from Product Adulteration. Working paper. UCLA Anderson School, March 2013.
3. Cassata, D. Lawmakers say counterfeits flood Pentagon supply. Associated Press. Nov. 8, 2011.
4. Gillette, F. Inside Pfizer's fight against counterfeit drugs. *Business Week.* (Jan. 17, 2013).
5. Tang, C.S. and Zimmerman, J. Managing new product development and supply chain risks: The Boeing 787 case. *Supply Chain Forum: An International Journal 10*, 2 (2009), 74–85.

Christopher S. Tang (chris.tang@anderson.ucla.edu) is Distinguished Professor and Edward Carter Chair in Business Administration, UCLA Anderson School, University of California at Los Angeles.

Joshua Zimmerman (joshuaz@amgen.com) is Senior Manager, Operations Risk Management, Amgen, Thousand Oaks, CA.

Calendar of Events

July 15–17
IEEE/ACM International Symposium on Nanoscale Architectures, New York, NY, Sponsored: SIGDA, Contact: Ramesh Karri, Email: rkarri@duke.poly.edu Phone: 718-260-3596

July 15–20
International Symposium on Software Testing and Analysis, Lugano, Switzerland, Sponsored: SIGSOFT, Contact: Mauro Pezze, Email: mauro.pezze@unisi.ch Phone: 39-02-6448-7882

July 19–20
High Performance Graphics Anaheim, CA, Sponsored: SIGGRAPH, Contact: Steven Molnar, Email: molnar@nvidia.com Phone: 919-408-0939

July 19–21
The ACM SIGGRAPH/ Eurographics Symposium on Computer Animation, Anaheim, CA, Sponsored: SIGGRAPH, Contact: Yizhou Yu, Email: yyz.uiuc@gmail.com

July 20
The Digital Production Symposium, Anaheim, CA, Sponsored: SIGGRAPH, Contact: Doug Epps, Email: doug.epps@comcast.net

July 20–21
Symposium on Spatial User Interaction, Los Angeles, CA, Sponsored: SIGCHI, SIGGRAPH, Contact: Evan Andrew Surna, Email: surna@ict.usc.edu

July 22–24
ACM Symposium on Principles of Distributed Computing, Montreal, Canada, Sponsored: SIACT, SIGOPS, Contact: Panagiota St. Fatourou, Email: faturu@csd.uoc.gr

July 22–26
13th ACM/IEEE-CS Joint Conference on Digital Libraries, Indianapolis, IN, Sponsored: SIGIR, SIGWEB, Contact: Stephen Downie, Email: jdownie@illiniois.edu

DOI:10.1145/2483852.2483863

Mari Sako

Technology Strategy and Management
The Business of Professionals

Expertise and service ethics are just two of the many components comprising a modern professional.

Computing professionals benefit from examining other professions with longer histories that arose from demands for new expertise. It requires more than technical excellence to translate demand into successful professional control over expertise. Professionalism is more than the promotion and certification of competence. But what is it? What are the strategies professionals have used to achieve high social respect and remuneration? What strategies should they be pursuing in the 21st century?

As a starting point, professionalism embodies expertise, a body of knowledge, and a service ethic. Fiduciary obligations of professionals extend beyond pursuit of self-interest, and involve advancing the common good through a code of ethics.

Where Did Professions Come From?

Most English accounts of professions start with lawyers and doctors, followed by a long list of others, including dentists, nurses, engineers, chemists, physicists, architects, and accountants.[2] Belonging to a profession implied a calling for talented amateurs and gentlemen seeking to employ their expertise to improve their station in society. Later, orderly careers grounded in higher education became the norm in 19th-century Victorian England. Professional associations, such as the Law Society and the Royal College of Physicians, began to require training and examinations as prerequisites before granting licenses to practice. Thus, self-regulation in matters of standards and discipline has been central to the Anglo-Saxon tradition of professions.

Continental Europe showed a different pattern, with the state playing a more prominent role.[1] The rise of the nation-state in the 16th century led to the creation of the military and civil

IMAGE FROM SHUTTERSTOCK.COM

servants in state bureaucracies. Engineering professions took off in the late 19th century in response to the needs of private enterprise in the industrial revolution and the state for public services such as railroads, gas and electricity, and telegraph and telephone. The rise of the welfare state since 1945 has given rise to professionals such as healthcare and social workers.

A Variety of Professional Types

Professions have evolved over the centuries and across geographies, factored on three dimensions: mode of training, required qualification, and standards enforcement.[6] First, some professions require practical training, as in residency for medical doctors where supervised practice is needed before they are deemed fully qualified. In such cases professional knowledge is a combination of theoretical and practical knowing, which is often difficult to codify fully. Second, some professions require a license that makes it illegal to practice without, while others merely require a certificate, which is a formal recognition of the level of competence achieved.[3] Third, ethical and performance standards are enforced in different ways. Self-regulated professionals believe that only members of their profession have the competence and appropriate ethics to enforce these standards, insisting that outsiders—lay members—cannot properly supervise their activities.

The three dimensions can be used to describe professional types (see the accompanying table). Classical professions such as physician and lawyer have all three characteristics: practical training, license to practice, and within-profession enforcement of standards. Nevertheless, they can be factored into *independent professionals* who have self-regulated associations to license and enforce standards, or *state-sponsored professionals* who are beholden to state agencies to grant licenses and enforce standards. *Organizational professionals* such as engineers and specialists in finance, marketing, or human resources, do not require practical training, do not require a license to practice, and have standards enforced by employers and clients. Knowledge professionals such as research scientists do not have formalized practical training, use certification rather than licensing, and exercise collegial enforcement of standards. These are ideal types; real professions often combine more than one type. Computing has characteristics of knowledge professionals and organizational professionals.

Professionalism is more than the promotion and certification of competence.

Professionalism is a matter of degree rather than a sharp distinction. What distinguishes between a proper profession, a quasi-profession, or an occupation? The lines can be drawn in multiple ways, and professionals who focus on traditional factors such as practical training, licensing, and self-regulation might not fare as well in the face of 21st-century challenges (as I will discuss later in this column).

Traditional Preoccupations of Professionals

Two long-standing preoccupations relevant for today's professionals are exclusive jurisdiction and professional autonomy.[8] A successful profession holds exclusive claim to a particular area of competence that goes well beyond maintenance of specific technical standards. Such exclusivity is enforced by licensing and gives professionals monopoly power that yields pay premiums.[3] Only licensed doctors can practice medicine, and only licensed lawyers are authorized to practice law. Licensing retains exclusive jurisdiction. As many as 1,100 occupations—hairdressers, florists, private detectives, and others—require license in at least one state in the U.S.

Exclusive jurisdiction rests on a claim to distinctiveness in the knowledge base. For distinctiveness, the theoretical and practical components of professional knowledge must be packaged so the knowledge is neither too narrow nor too broad. The drive toward differentiation of expertise in many professions is seen in the 25 board-certified specialties (encompassing over 125 sub-specialties) for U.S. physicians,[7] and 54 specialties for lawyers in England and Wales. At the same time, many organizational professionals, including IT professionals, are broadening their knowledge base by incorporating elements of business management. Ironically, this latter mode of pursuing distinctiveness can result in overlapping knowledge bases with other professions, so this can be self-defeating.

Some professionals have been preoccupied with defense of professional autonomy, through discretion at work and maintenance of collegiate control over performance and ethical standards. Solo practitioners, characterized by C. Wright Mills as "free professionals" amongst the old middle class, have such autonomy,

Professional types.

	Independent professionals	State-sponsored professionals	Organizational professionals	Knowledge professionals
Practical training requirements?	Yes	Yes	No	No
License to practice?	Yes	Yes	No	No
Standards enforced by:	Professional association	The state	Employers and clients	Collegial, by peers

but they are increasingly rare.[4] On the other hand, "salaried professionals" now constitute approximately one-fifth of the total labor force in many developed economies. Some of them work in professional service firms such as law firms and architectural design houses owned and managed by professionals. Many others work in complex organizations that threaten professional autonomy. Doctors in hospitals, teachers in schools, and engineers and scientists in commercial R&D labs are salaried professionals with neither exclusive nor final responsibility for their work. They accept the authority of non-professional managers, and can be overruled by managers. A particularly controversial example is medical professionals who feel patient interests are subordinated to the commercial objectives of private hospitals.

21st-Century Challenges for Professionals

The last century has seen the growth of a specific model for legitimizing professional power and status, based on an ordered career grounded in higher education. In promoting standards, professionals have had a tacit agreement with the state that amounts to a public trusteeship model. Professionals deliver expertise, a service ethic, and public protection in return for high status, reasonable compensation, and limited competition. The future will be different, for three reasons.

Consumerism Undermines the Public Trusteeship Model. Societal norms for service are changing from public trusteeship to a narrower model of service based on expertise. For example, British deregulation of legal services embodies a logic of consumer protection and market-based tests in which professionals provide clients what they want, rather than act as custodians of clients' interest. Consumerism is also evident in the public sector, where demands for greater transparency and accountability lead to more auditing and inspection of professional work in healthcare, schools, and universities.

Professional Services become Corporatized. Professional autonomy has been threatened by commercial

In promoting standards, professionals have had a tacit agreement with the state that amounts to a public trusteeship model.

organizations as more doctors become employees of large hospitals or healthcare systems, and many specialists, including IT professionals, have acquired the competences to be part of board-level decision-making in corporations. Some professionals have preserved autonomy and collegial control by forming limited liability partnerships, but growth in size and geographic coverage of professional service firms have pushed collegial control to its limit. In response, some firms have abandoned the partnership model, or introduced a managerial hierarchy that threatens the autonomy of professional work. Thus, threats to the autonomy of professional work used to come from commercial enterprises, but are coming increasingly from within professional service firms.

Professional Service Markets Globalize through Trade, Investment, and Offshoring. Talented individuals now experience greater cross-border mobility, but international mobility for professionals is hampered by qualification and license rules of national governments and state-level authorities.[5] Supra-national efforts at harmonization and mutual recognition of national qualifications are under way, but are slow to bear fruit.

These forces affect professionals. The state has become less willing to grant professional privileges and is keener to introduce market principles to improve the quality of professional services. Professionals must renegotiate the public trusteeship model, and are at risk of losing self-regulation. State-sponsored professionals can suffer in a globalizing world unless they can shift their locus of negotiation to supra-national levels. In contrast, organizational and knowledge professionals can do well by adjusting their expertise and their service ethics.

Implications for "Computing as a Profession." Professional prosperity should resonate with computing professionals. The aim of ACM is to advance "computing as a science and a profession." This requires a complex set of considerations. The main audience for the scientist is fellow scientists who are in a position to judge competence. In contrast, the main audience for the professional is clients or employer-clients who usually cannot judge competence. Yet consumerism implies that clients will demand greater transparency and accountability in their attempt to judge competence.

As organizational professionals, computing specialists face the challenge of incorporating business competences to enhance their reputation for managing the delivery of services. As knowledge professionals, computing specialists can be ahead of the game compared to other professions by having internationally recognized certifications. Some of the considerations for "computing as a profession" for the 21st century go beyond the educational mission of ACM to advance "the art, science, engineering, and application of information technology." C

References

1. Brante, T. State formations and the historical take-off of continental professional types: The case of Sweden. *Sociology of Professions: Continental and Anglo-Saxon Traditions*. L.G. Svensson and J. Evetts, Eds. Gotenberg, Daidalos, 2010.
2. Carr-Saunders, A.M. and Wilson, P.A. *The Professions*. Frank Cass & Co., London, U.K., 1933, 1964.
3. Kleiner, M.M. Occupational licensing. *Journal of Economic Perspectives 14*, 4 (2000), 189–202.
4. Mills, C.W. *White Collar: The American Middle Classes*. Oxford University Press, New York, 1951, 2002.
5. Sako, M. Globalization of knowledge-intensive professional services. *Commun. ACM 52*, 7 (2009), 31–33.
6. Sako, M. Professionals between market and hierarchy: A comparative political economy perspective. *Socio-Economic Review 11* (2013), 1–28.
7. Scott, W.R. Lords of the dance: Professionals as institutional agents. *Organization Studies 29*, 2 (2008), 219–238.
8. Wilensky, H.L. Professionalization of everyone? *American Journal of Sociology 70*, 2 (1964), 137–158.

Mari Sako (mari.sako@sbs.ox.ac.uk) is Professor of Management Studies at Saïd Business School, University of Oxford.

DOI:10.1145/2483852.2483864

Ron Eglash, Juan E. Gilbert, and Ellen Foster

Broadening Participation
Toward Culturally Responsive Computing Education

Improving academic success and social development by merging computational thinking with cultural practices.

CULTURALLY RESPONSIVE COMPUTING education is an exciting new field that has the potential to raise the achievement and interest of students from underrepresented ethnic groups. Culturally responsive education can be used to explore problems and solutions in any scientific or technical field, often using traditional knowledge or practices of the group being educated. While much of this work has been focused in the U.S. with African-American, Latino, and Native American students, it can be applied elsewhere. University of Finland's Matti Tedre, for example, found that his computing students in Tanzania did not understand programming examples that referenced European games of chance.

The benefits of culturally responsive education are not limited to raising test scores: it can help all students understand the relevance of education to issues of social justice, improve the inclusive scope of educational practice, and in many ways better serve the needs of a multicultural, democratic society. Whether the concern is the academic gap between majority and minority youth in developed nations, or the need to inspire a new generation of computing professionals in the developing world, teaching with the use of artifacts, practices, narratives, and contexts from either a particular ethnic heritage or a more general vernacular sensibility can contribute to both improved academic success and the moral and social development of youth in computing careers.

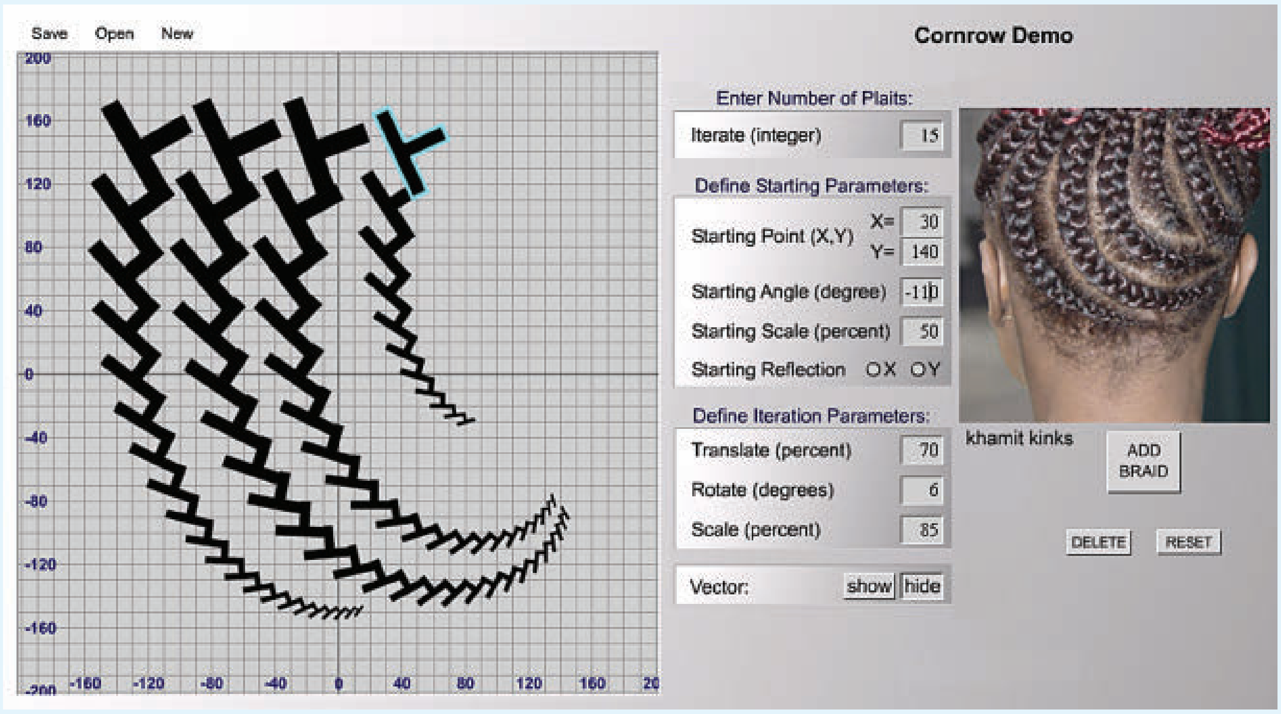
The benefits of culturally responsive education are not limited to raising test scores.

Factors in Underrepresented Youth STEM Achievement

We need to acknowledge that students vary widely in their interests and responses; no single strategy will best suit all underrepresented students. Some students are strongly affected by the direct impact of economic forces: less stable living conditions, fewer resources, attendance in underserved schools, and other factors. However, there is ample evidence that cultural factors can play a significant role. For example, several researchers have documented the ways in which high-achieving African-American students were accused of "acting white" by their peers.[9] Similar cultural barriers emerge in areas such as African-American conceptions of the "cultural ownership" of mathematics and the conflict between mainstream stereotypes of scientists and African-American cultural orientation.[3,12]

Myths of genetic determinism create another barrier. Claims about IQ averages for women and minorities, for example, are sometimes used to uphold destructive stereotypes. Yet there has been a steady, well-documented increase in IQ across several decades of testing (called the "Flynn Effect"). If IQ was a genetically fixed characteristic, we would not see these increases. But they are well explained if we think of IQ as impacted by the quality of education. Indeed James Flynn, for whom the effect is named, published a study showing the black-white IQ gap has been decreasing since the civil rights movement and school desegregation of the 1960s.[2] This is just one of many studies that has discredited these myths of genetic determinism (for details see Fischer et al[7]). However if children or teachers believe in the myth, it can have real impact. African-American students do worse on standardized testing when they are told the test may be reflecting racially determined intelligence.[13] The same "stereotype threat" can be seen on women's test performance (despite the fact the male-female IQ gap has dramatically decreased in the wake of equity efforts). In other words, while the genetic claims themselves are bogus, the myth of genetic determination of intelligence becomes a self-fulfilling prophecy. If you believe your low

Figure 1. Simulation for cornrow hairstyles.



Cornrow Demo

Enter Number of Plaits:

Iterate (integer) 15

Define Starting Parameters:

Starting Point (X,Y) X= 30 Y= 140

Starting Angle (degree) -110

Starting Scale (percent) 50

Starting Reflection OX OY

Define Iteration Parameters:

Translate (percent) 70

Rotate (degrees) 6

Scale (percent) 85

Vector: show hide

scholastic performance is genetically fixed, there is no point in trying. The myths of genetic determinism diminish motivation, excuse poor performance, and divert underrepresented students toward identities focused on sports and entertainment.

A wide variety of culturally responsive frameworks—sometimes referred to as "ethnocomputing"—have been developed to address these non-economic barriers. In this column we highlight the diversity of these approaches and some of their preliminary results.

Indigenous Knowledge in Computing Education

John Ogbu's ethnographies of African-American children documented how their sense of cultural authenticity ("keepin' it real") meant pride in non-academic subjects (such as sports or music). By demonstrating the sophisticated mathematical and computational thinking embedded in traditional cultural practices, students can discover, through their own design activities, opportunities to directly oppose primitivist stereotypes—including myths of genetic determinism—and incorporate computational thinking as a part of their cultural heritage rather than outside of it.

A wide variety of such simulations are freely available for educational purposes on our website at http://www.csdt.rpi.edu. These include the use of recursive geometric transforms in modeling cornrow hairstyles (see Figure 1), iterative patterns on Cartesian grids in Native American beadwork, and fractal models of traditional African arts and architecture. Several of these tools have demonstrated statistically significant improvement in pre-college student's math and/or computing understanding.[4] In Eglash et al.,[6] for example, we compared the effectiveness of two web-based curricula for teaching fractal geometry to 10th grade high school classes with a majority of African-American and Latino students. In the "experimental" class students used our culture-based fractal instruction (http://csdt.rpi.edu/african/African_Fractals); in the "control" class (taught by the same instructor) they used a popular website for teaching fractals (which also included Java applets but no cultural design activities). Pre/post differences on both achievement and attitude tests indicate statistically significant improvement for students in the experimental class.

We should caution that it is largely the teacher and students who create the learning environment; the simulations are simply tools to facilitate these connections. While evaluations have shown statistically significant results, there is one notable contradiction to the majority of the culturally responsive education literature: when offered the opportunity to use any of the tools in our suite, there is not a strong correlation between the heritage identity of the student and the cultural origin of the tool. We have seen Yupik children in a remote Alaskan village gleefully creating virtual cornrow hairstyles, and African-American children creating iterative patterns in native American bead-loom tool. Is this broader attraction because their racial/cultural identity is more "hybrid" than we expect? Or are they simply responding positively to the idea of "anti-primitive" or "anti-racist" education regardless of its ethnic origin? These remain important questions for future study.

Vernacular Culture in Computing Education

Unlike the heritage culture of indigenous knowledge, vernacular culture corresponds to domains such as rap music, "street smarts," urban graffiti, and a broad variety of other popular activities that children from underrepresented ethnic groups feel some sense of ownership or affinity toward. While the modeling approach described earlier can also be applied to vernacular culture (for example, there are culturally situated design tools modeling graffiti and breakdancing), it can also be integrated by offering computational activities in a vernacular context. For example, the African-American Distributed Multiple Learning Styles Systems (AADMLSS) began with the specific goal of developing information technology for math learning lessons that would be culturally responsive to the identities of African-American urban youth.[10] This game-like virtual environment allows cultural identity to be conveyed through a variety of signifiers: not only the ethnic identities of characters, but also a narrative of actions, contexts, and stylistic elements in sound and image that would be familiar and engaging to urban students (see Figure 2). Expansion from mathematics to computing education is currently under way. Other ongoing experiments with vernacular culture in pre-college education include Brian Magerko and Jason Freeman's Earsketch project—teaching Python coding via hip-hop music at Georgia Tech—and Christopher Emdin's use of rap in a broader STEM education program through Columbia University.

Civic Culture in Computing Education

Several researchers have developed culturally responsive STEM education based on the idea of civic responsibility. UCLA's *Mobilizing for Innovative Computer Science Teaching and Learning* at UCLA, headed by Deborah Estrin, makes use of a "Participatory Sensing" system to allow K–12 students to upload data captured by mobile phones to web servers that systematically collect and interpret data. Projects include mapping recycling bins around schools and neighborhoods, mapping travel routes to reduce carbon footprints, and an inventory of tree species to analyze the prevalence of asthma/allergy triggers.

To what extent can cultures of technology "appropriation" contribute to computing education?

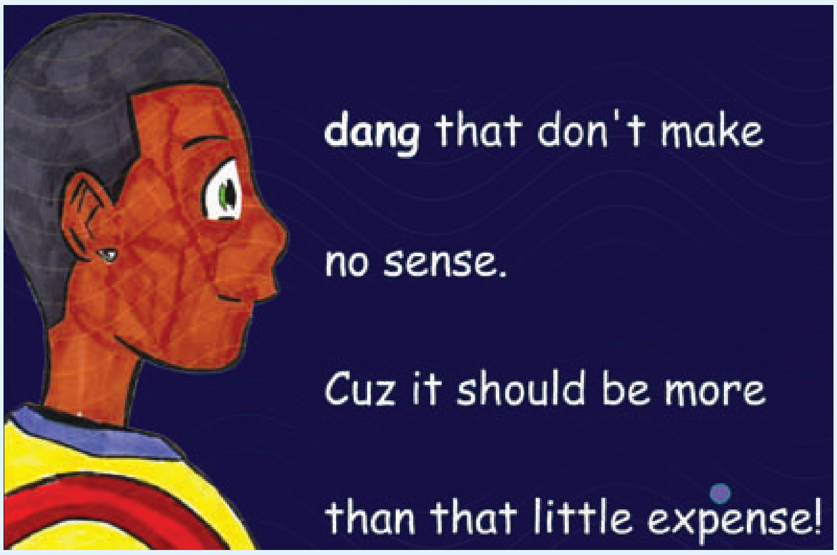
Such "participatory sensing" need not be generic. Our own "culturally situated sensing" project (http://www.3helix.rpi.edu/?p=2419), under the NSF-funded Triple Helix program, has experimented with tools and curricular materials that combine computing education with needs specific to indigenous communities. In collaboration with the Diné Environmental Institute in the Navajo Nation's community college system for example, we developed culturally specific sensing lessons, using the Cartesian structure of Navajo rugs to learn about coordinates in GIS. At Kwame Nkrumah University of Science and Technology in Ghana we have introduced Arduino-based systems to both U.S. and Ghanaian undergraduates. One unique project, led by graduate student David Banks, allows the condom vending machines he introduced to Ghana to send a cellphone text message when they need to be refilled, and helps local citizens to find their locations (http://www.3helix.rpi.edu/?p=3298).

It is possible to take the "civic culture" approach even farther, and address issues of social justice. Terry[14] for example worked with a group of African-American students to develop a statistical analysis of the changes in crime rates. He concludes that their experience fits well with the "counterstory" framework from critical race theory, in which alternative explanations challenge hegemonic claims, and open possibilities for transformation among the marginalized. Gutstein,[11] who also reports strong success with the social justice approach, cautions that computing or calculating with social data could reinforce negative stereotypes or have a depressing effect on students if it is not properly introduced.

Appropriating Technology: Hacking Culture in Computing Education

Previous work by our group (Eglash et al.[5]) analyzed how technologies can be reinterpreted, repurposed, or reinvented by low-income or other disenfranchised groups: low-rider cars, "scratch" turntables, and other user-modified gadgets are just one part of a broader culture of "hacking" that has since exploded into "maker" fairs, DIY

Figure 2. Clip from AADMLSS animation. The bold print on "dang" follows the audio intonation, and the purple spot over "expense" is following the audio as it is spoken. Rap's heavy emphasis on spoken word as an overlay to music smoothly meshed with the pedagogy.



websites (for example, instructables), and a wide variety of other activities. To what extent can cultures of technology "appropriation" contribute to computing education?

The NSF-funded "Triple Helix" project has explored this question in an after-school program based around extracting and reusing of parts from discarded machines (printers, scanners, desktop computers). This approach has several advantages: it adds a sustainability component by raising awareness of the problem of "e-waste"; it solves the problem of obtaining expensive parts under restricted school budgets; and it links to a culture of hacking that earns "cool points" with students (who are primarily from low-income African-American and Latino families). Most importantly, we have found that the participants are instantly curious about how to make use of circuitry, motors, and other components. In the long term we hope to connect this hardware to a repertoire of computing concepts and practices (perhaps via inexpensive microcontrollers) that allow the children to repurpose technologies for self-empowerment, expression, and community engagement.

Taking a very different route, Buechley and Hall[1] report on the participation of women using the "LillyPad" microcontroller kit, an Arduino variant for e-textiles developed and marketed by Leah Buechley. They found that although women constituted only 35% of LillyPad purchases, they were responsible for 65% of the online projects, a strong indicator of the increased participation among women for this case of creative computing that better fits their interests. Regarding its application to K–12 computing education, they note "instead of trying to fit people into existing engineering cultures, it may be more constructive to try to spark and support new cultures."

Conclusion

In our view, culture-based approaches to computing education offer a promising array of approaches to increasing the interest and engagement of underrepresented students. These are not merely important for the instrumental reason of raising test scores. Research indicates that "social creativity" and ethnic exploration are important means by which children from devalued or disempowered ethnic groups are able to develop a healthy self-identity.[8] Typically this is described in terms of experimentation in music, clothing, food, language, and other attributes of personal style. With a diverse array of culturally responsive learning environments, math and computing can also be part of this repertoire of healthy identity self-construction. C

References

1. Buechley, L. and Hill, B.M. LilyPad in the wild: How hardware's long tail is supporting new engineering and design communities. In *Proceedings of Designing Interactive Systems (DIS)*, Aarhus, Denmark (2010), 199–207.
2. Dickens, W.T. and Flynn, J.R. Black Americans reduce the racial IQ gap: Evidence from standardization samples. *Psychological Science 17*, 10 (Oct. 2006), 913–920.
3. Eglash, R. Race, sex and nerds: From Black geeks to Asian-American hipsters. *Social Text 20*, 2 (2002) 49–64.
4. Eglash, R., Bennett, A., O'Donnell, C., Jennings, S., and Cintorino, M. Culturally situated design tools: Ethnocomputing from field site to classroom. *American Anthropologist 108*, 2 (2006), 347–362.
5. Eglash, R., Croissant, J., DiChiro, G., and Fouché, R. *Appropriating Technology: Vernacular Science and Social Power.* University of Minnesota Press, Minneapolis, MN, 2004.
6. Eglash, R., Krishnamoorthy M., Sanchez J., Woodbridge, A. Fractal simulations of African design in pre-college computing education. *ACM Transactions on Computing Education 11*, 3, Article 17 (2011).
7. Fischer, C.S., Swidler, A., Voss, K., Lucas, S.R., Jankowski, M.S. *Inequality by Design: Cracking the Bell Curve Myth.* Princeton University Press, Princeton, NJ, 1996.
8. French S.E., Seidman E., Allen, L., Aber, J.L. The development of ethnic identity during adolescence. *Developmental Psychology 42* (2006), 1–10.
9. Fryer, R.G., Jr., and Torelli, P. *An Empirical Analysis of 'Acting White.'* Electronic document (2005), http://post.economics.harvard.edu/faculty/fryer/papers/fryer torelli.pdf.
10. Gilbert, J.E. et al. Teaching algebra using culturally relevant virtual instructors. *The International Journal of Virtual Reality 7*, 1 (2008), 21–30.
11. Gutstein, E. Teaching and learning mathematics for social justice in an urban, Latino school. *Journal for Research in Mathematics Education 34*, 1 (2003), 37–73.
12. Martin, D. *Mathematics Success and Failure among African-American Youth: The Roles of Sociohistorical Context, Community Forces, School Influence, and Individual Agency.* Lawrence Erlbaum Associates, Mahwah, NJ, 2000.
13. Steele, C., Spencer, S., and Aronson, J. Contending with group image: The psychology of stereotype and social identity threat. In *Advances in Experimental Social Psychology 37*, M. Zanna Ed. Academic Press, 2002.
14. Terry, L. Mathematical counterstory and African-American male students: Urban mathematics education from a critical race theory perspective. *Journal of Urban Mathematics Education 4*, 1 (July 2011), 23–49.

Ron Eglash (eglash@rpi.edu) is a professor in the Department of Science and Technology Studies at Rensselaer Polytechnic Institute, Troy, NY.

Juan E. Gilbert (juan@juangilbert.com) is Professor and Chair of the Human-Centered Computing Division in the School of Computing at Clemson University, where he directs the Human-Centered Computing Lab.

Ellen Foster (fostee21@rpi.edu) Ph.D. candidate in science and technology studies at Rensselaer Polytechnic Institute, Troy, NY.

DOI:10.1145/2483852.2483865

Martin Ford

Viewpoint
Could Artificial Intelligence Create an Unemployment Crisis?

Advances in artificial intelligence and robotics will have significant implications for evolving economic systems.

THERE IS AN often-told story about the libertarian economist Milton Friedman. While visiting a large-scale public works project in a developing Asian nation, Friedman asked a government official why he did not see much heavy earth-moving equipment in use; instead, there were large numbers of workers with shovels. The official explained that the project was intended as a jobs program. Friedman replied with his famous and caustic question: "So why not give the workers spoons instead of shovels?"

That story is a pretty good indication of the almost reflexive derision that is likely to arise in response to any serious speculation about the possibility that advancing technology could destroy jobs and cause long-term structural unemployment. Nonetheless, I think there are good reasons to be concerned that advances in artificial intelligence and robotics are rapidly pushing us toward an inflection point where the historical correlation between technological progress and broad-based prosperity is likely to break down—unless our economic system is adapted to the new reality.

Why should the implications of today's accelerating information technology be different from the innovations of the past? I believe the answer lies in the nature of the transition that will be required for the majority of the workforce to adapt and remain relevant.

Most of the work required by the economy is—on some level—fundamentally routine in nature. By this, I do not mean the work is rote repetitive, but rather that it can be broken down into a series of discrete tasks that are relatively predictable and tend to get repeated over some time frame. The percentage of people who are paid primarily to engage in truly creative or non-routine occupations is fairly small. This has always been the case, and the repetitive nature of most jobs has historically been a good match with the capabilities of the average worker.

Technology has, of course, often disrupted and even destroyed whole industries and employment sectors. In the U.S., the mechanization of agriculture vaporized millions of jobs and led workers to eventually move from farms to factories. Later, manufacturing automation and globalization caused the transition to a service economy. Workers repeatedly adapted by acquiring new skills and migrating to jobs in new industries—but these changes have not altered the fact that most jobs continue to be essentially routine.

In the past, disruptive innovations have tended to be relatively specialized and to impact on a sector-by-sector basis. Workers have responded by moving from routine jobs in one area to routine jobs in another. Today's information technology, in contrast, has far more broad-based implications: it is transforming and disrupting every sector of the economy. For the first time in his-

ILLUSTRATION BY ACCENT/SHUTTERSTOCK.COM

tory, computers and machines are increasingly taking on intellectual tasks that were once the exclusive province on the human brain. Information technology will continue to accelerate, and it is certain to be tightly integrated into any new industries that arise in the future.

The impact of information technology on the job market, and in particular on more routine jobs, has been well documented.[2,3] Economist David Autor of MIT, in particular, has done extensive analysis showing that the job market in the U.S. has become polarized.[1] A substantial fraction of moderate wage, routine jobs in areas like manufacturing and white-collar clerical occupations have been eliminated by technology, leaving the remaining employment opportunities clustered at the top (high-wage/high-education jobs) and at the bottom (low-wage jobs requiring little education).

While economists have noted the correlation between whether or not a job is routine and its susceptibility to automation, I do not think they have yet fully acknowledged the future impact that accelerating progress is likely to have. Our definition of what constitutes a "routine" job is by no means static. At one time, the jobs at risk from automation were largely confined to the assembly line. The triumph of IBM's Watson computer on the television game show "Jeopardy!" is a good illustration of how fast the frontier is moving. I suspect very few people would characterize playing "Jeopardy!" at a championship level as routine or repetitive work, and yet a machine was able to prevail.

Machine learning, one of the primary techniques used in the development of IBM's Watson, is in essence a way to use statistical analysis of historical data to transform seemingly non-routine tasks into routine operations that can be computerized. As progress continues, it seems certain that more and more jobs and tasks will move from the "non-routine" column to the "routine" column, and as a result, an ever-increasing share of work will become susceptible to automation.

It is important to realize technology does not have to cause immediate job destruction in order to create significant future unemployment.

This goes to the heart of why the historical record many not be predictive with regard to technological unemployment. In order to remain essential to the production process, workers will have to make a historically unprecedented transition. Rather than simply acquiring new skills and moving to another routine job, workers will have to instead migrate to an occupation that is genuinely non-routine and therefore protected from automation—and they may have to do this rapidly and repeatedly in order to remain ahead of the advancing frontier.

There are good reasons to be pessimistic about the ability of most of our workforce to accomplish this. If we assume, as seems reasonable, a normal distribution of capability among workers, then 50% of the workforce is by definition average or below average. For many of these people, a transition to creative/non-routine occupations may be especially challenging, even if we assume that an adequate number of such jobs will be available.

Both the high and low ends of our polarized job market are likely to come under attack as technology advances. Higher-wage white-collar jobs will be increasingly susceptible to software automation and machine learning. One of the biggest drivers of progress in this area is likely to be the "big data" phenomenon and the accompanying emphasis on algorithmic techniques that can leverage the enormous quantities of data being collected.

Much of the initial focus has been on how big data can be used to give organizations a competitive advantage in terms of marketing and customer relationships. However, corporations are certainly also collecting huge amounts of internal information about the work being done by employees and about their interactions with customers—potentially creating a rich dataset that future machine learning algorithms might churn through.

The impact is already being felt in a number of professions. Lawyers and paralegals have been displaced by e-discovery software that can rapidly determine which electronic documents are relevant to court cases. More routine forms of journalism—such as basic sports and business writing—have been successfully automated. Entry-level positions are especially vulnerable, and this may have something to do with the fact that wages for new college graduates have actually been declining over the past decade, while up to 50% of new graduates are forced to take jobs that do not require a college degree.[5]

The polarized nature of the job market means workers who fail to find and

retain one of the high-end jobs face a long fall. The lower-end jobs are heavily weighted toward hourly service positions with minimal wages and few benefits. These, often part-time, jobs in areas like retail, fast food, and full-service restaurants, have traditionally offered a kind of income safety net for workers with few other options.

Yet there are good reasons to expect that even these lower-range jobs may soon come under significant pressure from technology. For example, it is easy to envision increased automation taking hold in the fast food and beverage industry. From a technical standpoint, fast food is not really a service industry at all: it is, rather, a form of just-in-time manufacturing, and there is no good reason to believe it will be forever exempt from the advances that are transforming other manufacturing sectors.

Retail jobs are also likely to be impacted. Self-service checkout lanes are becoming increasingly prevalent and popular. Mobile applications offer in-store access to product information and customer service. Wal-Mart is currently testing a service that allows customers to scan barcodes and then pay for their purchases with their mobile phones—completely avoiding lines and cashiers.

Brick-and-mortar retailers will also continue to be disrupted by online competitors like Amazon, especially as Internet retailers offer faster delivery options and as customers increasingly use mobile technology to look for lower prices online. In theory, this should not destroy jobs but simply transition them from traditional retail settings to warehouses and distribution centers. However, once jobs move to a warehouse environment, they seem likely to be more susceptible to automation. Amazon's purchase of Kiva Systems—a company that focuses on warehouse robotics—is probably indicative of the trend in this area.

Many low-wage jobs have been protected from automation primarily because human beings are extremely good at tasks requiring mobility, dexterity, and hand-eye coordination, but these advantages are certain to diminish over time. Robots are rapidly advancing while becoming less expensive, safer, and more flexible, and it is reasonable to expect they will have a potentially dramatic impact on low-wage service sector employment at some point in the not too distant future.

It is important to realize technology does not have to cause immediate job destruction in order to create significant future unemployment. The U.S. economy needs to generate in excess of 100,000 new jobs per month just to keep up with population growth. As a result, anything that significantly slows the rate of ongoing job creation could have a significant impact over the long term. Because workers are also consumers, entrenched technological unemployment would be very likely to depress consumer spending and confidence—thereby spawning a wave of secondary job losses that would affect even occupations not directly susceptible to automation.[4]

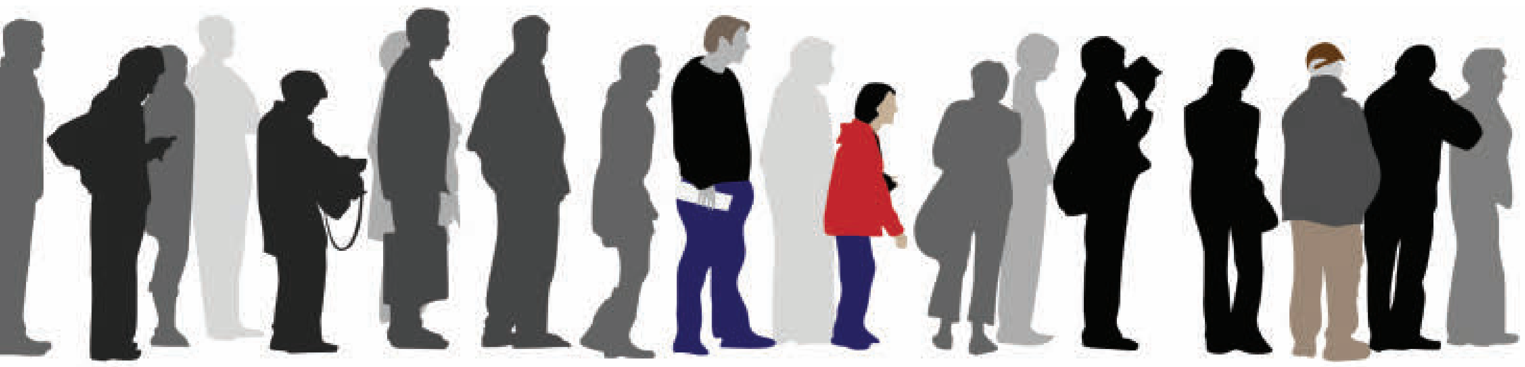
I suspect the impact of accelerating technology on the job market may ultimately represent a dramatic and vastly under-acknowledged challenge for both our economy and society. Many extremely difficult issues would arise, including finding ways for people to occupy their time and remain productive in a world where work was becoming less available and less essential. The biggest immediate challenge, however, would be one of income distribution: how will people without jobs and incomes support themselves, and how will they be able to participate in the market and help drive the broad-based consumer demand that it vital to sustained economic prosperity and innovation?

Finally, it is worth noting everything I have suggested here might be thought of as the "weak case" for technological disruption of the job market. I have presumed only that narrow, specialized forms of machine intelligence will increasing eliminate more routine jobs. None of these technologies would be generally intelligent or could pass a Turing test. Yet, the more speculative possibility of strong AI cannot be completely discounted. If, someday, machines can match or even exceed the ability of a human being to think and to conceive new ideas—while at the same time enjoying all the advantages of a computer in areas like computational speed and data access—then it becomes somewhat difficult to imagine just what jobs might be left for even the most capable human workers. C

References

1. Autor, D.H., Katz, L.F., and Kearney, M.S. The polarization of the U.S. labor market. *American Economic Review 96*, 2 (May 2006), 189–194.
2. Autor, D.H., Levy, F., and Murnane, R.J. The skill content of recent technological change: An empirical investigation. *Quarterly Journal of Economics 118*, 4 (Nov. 2003), 1279–1333.
3. Brynjolfsson, E. and McAfee, A. *Race Against the Machine: How the Digital Revolution is Accelerating Innovation, Driving Productivity, and Irreversibly Transforming Employment and the Economy.* Digital Frontier Press, 2011.
4. Ford, M. *The Lights in the Tunnel: Automation, Accelerating Technology and the Economy of the Future.* Acculant Publishing, 2009.
5. Hagerty, J.R. Young adults see their pay decline. *The Wall Street Journal* (Mar. 9, 2012); http://online.wsj.com/article/SB10001424052970204276304577265510046126438.html.

Martin Ford (lightstunnel@yahoo.com) is a software developer, entrepreneur, and author of the book *The Lights in the Tunnel: Automation, Accelerating Technology and the Economy of the Future*, which focuses on the economic impact of artificial intelligence and robotics. He has a blog at http://econfuture.wordpress.com.



DOI:10.1145/2483852.2483867

Article development led by acmqueue
queue.acm.org

We simply do not have a synchronization mechanism that can enforce mutual exclusion.

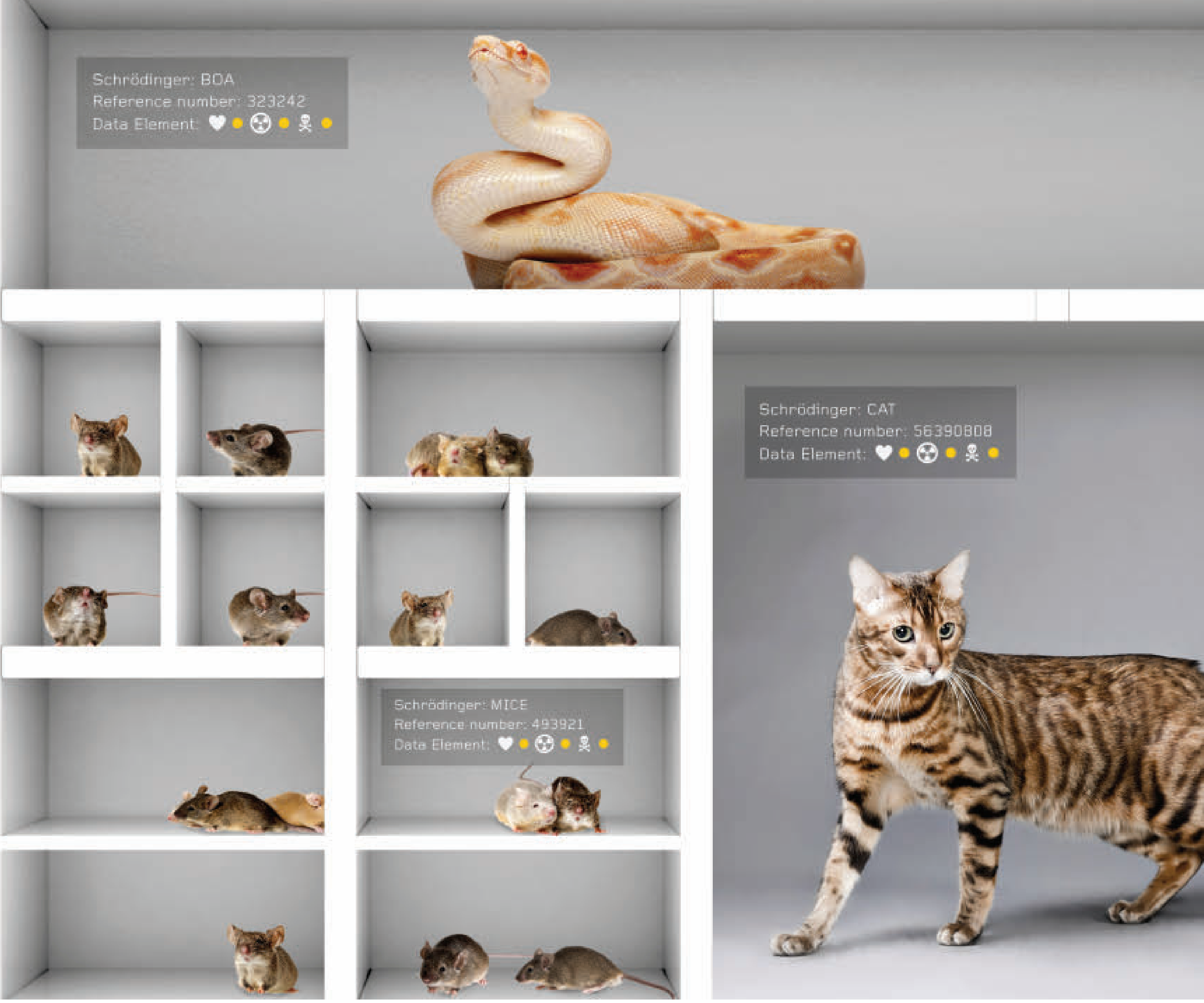
BY PAUL E. MCKENNEY

Structured Deferral: Synchronization via Procrastination

DEVELOPERS OFTEN TAKE a proactive approach to software design, especially those from cultures valuing industriousness over procrastination. Lazy approaches, however, have proven their value, with examples including reference counting, garbage collection, and lazy evaluation. This structured deferral takes the form of synchronization via procrastination, specifically reference counting, hazard pointers, and read-copy-update (RCU).

Synchronization via procrastination extends back to H.T. Kung's and Philip Lehman's 1980 paper,[12] with the general principle articulated by Henry Massalin in 1992.[13] Although these ideas have been used in production for decades,[9] they are still unfamiliar to many. This article provides an introduction to structured deferral by means of a fanciful example described here. In this example, Schrödinger would like to construct an in-memory database to keep track of the animals in his zoo. Births would of course result in insertions into this database, while deaths would result in deletions. The database is also queried by those interested in the health and welfare of Schrödinger's animals.

Schrödinger has numerous short-lived animals such as mice, resulting in high update rates. In addition, there is a surprising level of interest in the health of Schrödinger's cat,[19] so much so that Schrödinger sometimes wonders whether his mice are responsible for most of these queries. Regardless of their source, the database must handle the large volume of cat-related queries without suffering from excessive levels of contention. Both accesses and updates are typically quite short, involving



accessing or mutating an in-memory data structure, and therefore synchronization overhead cannot be ignored.

Schrödinger also understands, however, that it is impossible to determine exactly when a given animal is born or dies. For example, suppose his cat's passing is to be detected by heartbeat. Seconds or even minutes will be required to determine that the poor cat's heart has in fact stopped. The shorter the measurement interval, the less certain the measurement, so that a pair of veterinarians examining the cat might disagree on exactly when death occurred. For example, one might declare death 30 seconds after the last heartbeat, while another might insist on waiting a full minute, in which case the veterinarians disagree on the state of the cat during the second half of the minute after the last heartbeat.

Fortunately, Heisenberg[8] has taught Schrödinger how to cope with such uncertainty. Furthermore, the delay in detecting the cat's passing permits use of synchronization via procrastination. After all, given the two veterinarians' pronouncements of death were separated by a full 30 seconds, a few additional milliseconds of software procrastination is perfectly acceptable.

Here, I illustrate synchronization via procrastination with reference counting, using this well-understood mechanism to demonstrate some of the less-familiar properties of structured deferral.

Reference Counting

A simple solution for Schrödinger's zoo is to place a reference counter in each animal's data element within a hash table, with collisions handled by chaining. Readers atomically increment the reference before accessing an animal's data element and atomically decrement it afterward. This provides synchronization only between readers and updaters; updaters must synchronize among themselves using other mechanisms such as locking, non-blocking synchronization, or transactional memory.

The four-state process of removing the data element corresponding to Schrödinger's poor cat is shown in Figure 1. The initial state (1) shows one chain of the hash table, representing Schrödinger's boa, cat, and gnu. As indicated by the red color of each box, any number of readers might be referencing these data elements; therefore, updates must be carried out carefully to avoid disturbing these readers. To transition to state 2, the updater stores a pointer to the gnu's data element into the `->next` pointer of the boa's data element. This store must be atomic in the sense any concurrent reader must see either the old value or the new one, not some mashup of the two. Such a store may be carried out using a C11/C11++ relaxed atomic variable[2] or in older C/C++ compilers, a volatile cast.[3] Note that the cat's `->next` pointer continues referencing the gnu's data element to accommodate readers still referencing the cat.

From this point forward, there is no path to the cat's data element (indicated by its yellow color), so new readers cannot gain access to it. Once the cat's reference counter reaches zero, transitioning to state 3, all readers that had a reference to the cat's data structure have released their reference, indicated by the green color of the cat's box. Because there is still no path to the cat's data element, new readers still cannot gain a reference, so it is now safe to transition to state 4 by freeing the late cat's data element.

This sequence of state transitions has therefore safely removed the cat's data element from the hash table, despite the presence of concurrent readers. There remains, however, the problem of obsolete references, to say nothing of correctness and performance.

Obsolete references. While in state 2, old readers can still hold old references to the cat's data structure, but new readers cannot gain a reference. These concurrent readers can therefore disagree as to whether the cat is still alive, just as the veterinarians in the example disagreed. This disagreement is therefore not a bug, but rather a faithful reflection of external reality.

In other cases, disagreements can be detected and rejected. For example, if accesses and updates to a given data element were protected by mutual exclusion, a "deleted" field would allow readers to reject deleted data by acting as if the search had failed.[1]

Finally, suppose an algorithm uses the common idiom that makes a decision while holding a lock, and then relies on that decision after releasing the lock. This algorithm is in fact relying on obsolete information because some other CPU might acquire the lock and change the data on which the decision was based—while the first CPU is still relying on its now-obsolete decision.

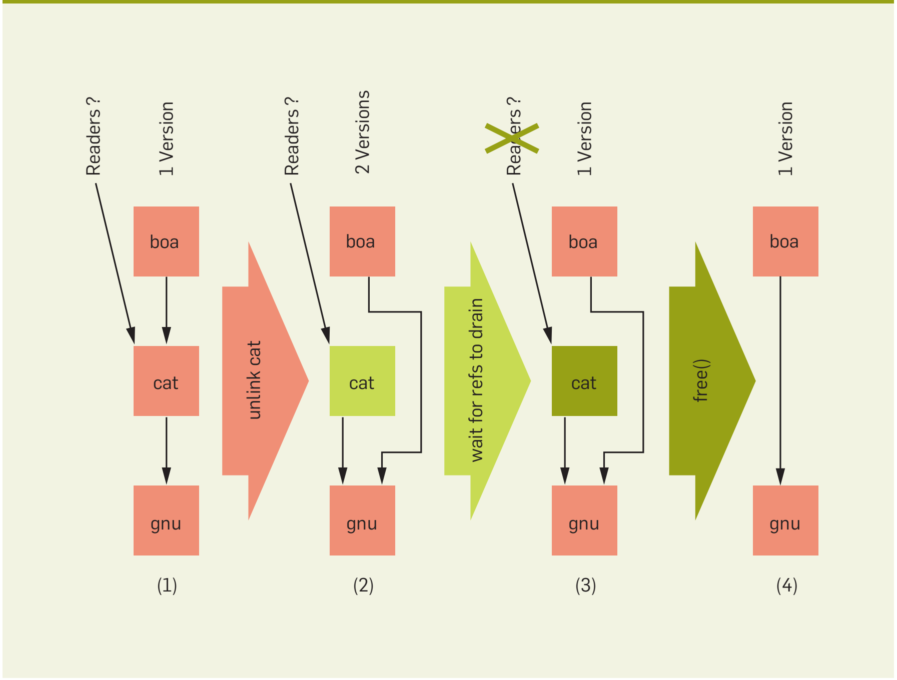
To see how this idiom works, consider a networking stack that acquires a lock, makes a routing decision, transmits a packet, releases the lock, and then updates statistics. Because the statistics need to reflect where you actually sent the packet, as opposed to where you might have sent it had you waited, using "obsolete" data is in fact correct. Furthermore, in most cases, the software does not actually transmit the packet but instead causes the hardware to queue the packet for later transmission. By the time the hardware actually transmits the packet, the routing decision might well have changed. Worse yet, it can take hundreds of milliseconds for a packet to travel from its source to a distant destination, providing even more opportunity for the routing decision to change. Therefore, it is reasonable to make the routing decision while holding the lock, and then release the lock before transmitting the packet—or to use synchronization via procrastination.

In addition, detecting hardware failure often takes significant time, resulting in a period during which it is uncertain whether or not the hardware has failed. Furthermore, many updates have a wide timing window—for example, a regulatory change might require a security-configuration update within a 90-day period. In both cases and in many similar situations, a few extra milliseconds of software procrastination during the update are quite acceptable. In fact, software procrastination can enable readers to become aware of external changes sooner[16 (Figure 17)], thus reducing response time.

In short, synchronization via procrastination is especially useful when interacting with external state.

Correctness and performance. Unfortunately, naïve use of reference counters results in several problems. The first problem is a race between deletion and read-side reference acquisition. To illustrate, suppose a reader fetches the boa's `->next` pointer while in the first state of Figure 1 and is then preempted. Before this reader resumes, an updater sequences through the rest of the states in the figure. When the reader resumes, it will atomically increment what used to be the cat's reference counter, but which might now be something else entirely, corrupting the application's state. This problem can be avoided via complex schemes based on compare-and-swap[21] (corrected by Maged Michael and Michael Scott[18]), but this results in abysmal performance.[7] Another way of avoiding this problem is to use a garbage collector so that the

Figure 1. Deferred deletion via reference counting.



cat's data structure persists for as long as it is referenced.

If there is no garbage collector, another approach is to reference-count the entire hash chain rather than the individual data elements. This works for a fixed-size hash table but falls prey to a similar race condition if the hash table can be resized. In theory, resizing can be handled using a global reference counter covering the entire hash table, but in practice this increases the probability that a continuous stream of concurrent readers would prevent the counter from ever reaching zero. If the counter never reaches zero, data elements removed from the hash table cannot be freed, eventually resulting in failure caused by memory exhaustion.

Furthermore, as shown in Figure 2, the atomic increment of a single variable simply does not scale: cross-thread references can be extremely expensive, because electrons are not infinitely fast. If a single thread attempts to increment a variable atomically, the cost on an Intel Xeon Westmere-EX system is about 10 nanoseconds for a single thread, rising to about 2,500 nanoseconds for 64 threads (the change in slope of the line is caused by hardware multithreading). This means the number of accesses drops from about 100 million for a single thread to about 400,000 on 64 threads. This is not the sort of scalability that Schrödinger requires.

Two methods for addressing these issues are hazard pointers and RCU, which are discussed in the next two sections. Another approach, called proxy collectors (http://atomic-ptr-plus.sourceforge.net/), amortizes the reference-count overhead but falls outside the scope of this article.

Hazard Pointers

The key insight behind hazard pointers is that reference counters can be implemented inside-out. Rather than incrementing an integer associated with a given data element, you instead record a pointer to that data element in a per-thread list of hazard pointers. The number of times a given data element's pointer occurs in the concatenation of these lists is that element's reference count. When a data element is to be freed, the free operation is deferred until there are no hazard pointers referencing it. Free operations are batched to minimize the number of expensive cross-thread references to the hazard pointers[17] (independently invented by others[10] and available from a number of sources, including http://concurrencykit.org/).

Because hazard pointers are thread-local, they avoid the performance and scalability problems faced by many reference-counter implementations.[7] They also avoid the race condition just described, as can be seen in `hp_acquire()` on lines 1–14 of Figure 3. The key points here are the need to allocate a hazard pointer (line 3), the need to avoid the race condition (the reload and check on line 9), and the need to defeat compiler code-motion optimizations. Compiler code motion is addressed by `ACCESS_ONCE()` on lines 6, 7, and 9, which may be implemented either as volatile casts or as C11/C++11 volatile relaxed atomic loads and stores. Compiler and CPU code motion is addressed by the memory barrier on line 8, which prevents reordering the prior store on line 7 with the subsequent load on line 9 and is thus required even on relatively strongly ordered systems such as x86. The `NULL` return on line 11 signals the caller that it must restart the hazard-pointer traversal from the beginning. All of this taken together ensures that the updater will have a correct view of the state of the hazard pointers.

Releasing a hazard pointer is straightforward: simply set it to NULL, either preceded by a memory barrier or using a C11/C++11 store-release operation; then free it so that it can be reused, as shown in `hp_release()` on lines 16-21 of Figure 3. The `hp_release()` function's sole argument is the pointer returned by `hp_acquire()`.

Most programs written using explicit reference counters can be easily converted to use `hp_acquire()` and `hp_release()`. However, algorithms that require only a fixed number of hazard pointers can allocate them statically, thus avoiding the overhead of `hp_alloc()`'s and `hp_free`'s dynamic allocation, as shown in `hp_record()` on lines 23–35 of Figure 3. Note that this approach does not release a given hazard pointer until that pointer is reused. In the worst case, this approach prevents a small fixed amount of memory from being freed, which is normally harmless. Although there are algorithms that require unbounded numbers of hazard pointers, simple searches and traversals of many data structures require at most two hazard pointers.

Hazard pointers work extremely well in many situations, but they do have some shortcomings.

Figure 2. Atomic increment does not scale.

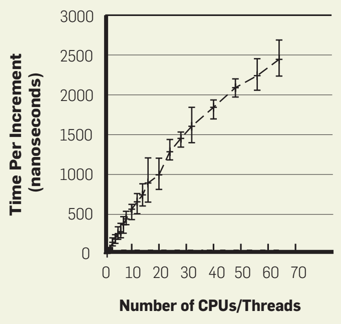


Figure 3. Hazard-pointer acquisition and release.

```
 1 void **hp_acquire(void **p)
 2 {
 3     void **hp = hp_alloc();
 4     void *tmp;
 5
 6     tmp = ACCESS_ONCE(*p);
 7     ACCESS_ONCE(*hp) = tmp;
 8     smp_mb();
 9     if (tmp != ACCESS_ONCE(*p)) {
10         hp_free(hp);
11         return NULL;
12     }
13     return hp;
14  }
15
16 void hp_release(void **hp)
17 {
18     smp_mb();
19     ACCESS_ONCE(*hp) = NULL;
20     hp_free(hp);
21 }
22
23 int hp_record(void **p, void **hp)
24 {
25     void *tmp;
26
27     tmp = ACCESS_ONCE(*p);
28     ACCESS_ONCE(*hp) = tmp;
29     smp_mb();
30     if (tmp != ACCESS_ONCE(*p)) {
31         ACCESS_ONCE(*hp) = NULL;
32         return 0;
33     }
34     return 1;
35 }
```

Figure 4. Nonpreemptible grace period.

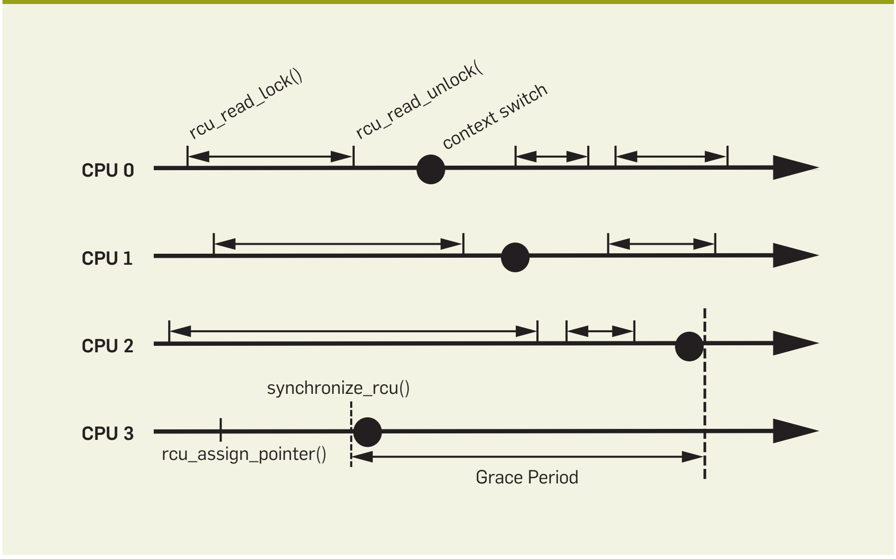


Figure 5. Textbook implementation of RCU.

```
#define rcu_read_lock()
#define rcu_read_unlock()
void synchronize_rcu(void)
{
  int cpu;

  for_each_online_cpu(cpu)
    run_on(cpu);
}
#define rcu_dereference(p)\
({\
  typeof(p) _p1 = ACCESS_ONCE(p));\
  smp_read_barrier_depends();\
  _p1;\
})
#define rcu_assign_pointer(p, v)\
({\
  smp_wmb();\
  ACCESS_ONCE(p) = (v);\
})
```

▸ Retries and memory barriers result in degraded performance.

▸ If hazard-pointer protection is required for a large group of data elements in a linked structure, then a separate hazard pointer must be acquired for each and every data element.

▸ Acquiring a hazard pointer requires memory, which must be managed. Although the static-allocation strategy used by Michael[17] works well in small programs for some types of data structures, in general allocation and freeing are required.

▸ Hazard pointers provide no provision for taking additional action when freeing memory—for example, shutting down threads or hardware associated with a given data element.

▸ Finally, in the general case, the code must keep track of the hazard pointers and explicitly release them when they are no longer needed.

Some of these shortcomings are inherent in hazard pointers' strengths, which include nonblocking updates, strong ordering properties, and reduced memory overhead.[7] Nevertheless, in operating-system kernels and large applications, hazard pointers' shortcomings can outweigh their strengths. Next, I present an alternative design that addresses these shortcomings—albeit by sacrificing some of hazard pointers' strengths. The lesson is, use the right tool for the job!

Read-Copy-Update

The goal is to produce a reference-counting scheme that has low (and preferably zero) overhead; needs no memory allocation when acquiring references; can protect arbitrary numbers of data elements with a single operation; and permits additional actions to be taken prior to freeing a given data element. One mechanism that meets these goals is RCU.[16]

To ensure the goal of low overhead is met, let us define acquiring a reference (`rcu_read_lock()`) and releasing it (`rcu_read_unlock()`) as no-ops that generate no code, thus achieving the best conceivable performance, scalability, real-time response, wait freedom, and energy efficiency. This admittedly unconventional approach has the additional benefit of dispensing with memory allocations. Given that `rcu_read_lock()` and `rcu_read_unlock()` take no arguments, there is no way for them to specify a particular data element, so they also meet the third goal of protecting all data elements.

Skeptics might argue that if `rcu_read_lock()` and `rcu_read_unlock()` generate no code, they cannot affect machine state and therefore cannot possibly act as synchronization primitives. Let us press on nonetheless: after all, only those who have gone too far can possibly know how far they can go.

Figure 1 shows how an element can be removed from a linked list despite the presence of concurrent readers, which indicates that Schrödinger does not need mutual exclusion. The transition from state 2 to state 3, however, requires waiting for all preexisting readers to complete. Therefore, the challenge is to implement an operation that waits for preexisting readers, even though `rcu_read_lock()` and `rcu_read_unlock()` generate no code.

To surmount this challenge, let us first consider the nonpreemptive software environment, where a given thread continues to execute until it voluntarily relinquishes the CPU. Examples of nonpreemptive software environments include the Linux kernel when built with `CONFIG_PREEMPT=n`, DYNIX/ptx, and numerous embedded systems. This type of environment can impose the convention that a thread is forbidden from relinquishing its CPU if it has executed `rcu_read_lock()` but has not yet reached the matching `rcu_read_unlock()`. (This same convention is required when holding pure spinlocks to avoid deadlock.) Such a thread is said to be in an *RCU read-side critical section*. Just as with reference counting and locking, if a reference to a given data element is obtained within a given RCU read-side critical section, that reference must be dropped before exiting that section, unless the data element has been handed off to some other synchronization mechanism.

Now suppose that the updater thread wishing to transition from state 2 to state 3 in Figure 1 sees CPU 0 execute a context-switch operation. Given that RCU readers must refrain from relinquishing their CPUs, this context switch implies all of CPU 0's prior RCU read-side critical sections have completed—in other words, that CPU 0 is in a *quiescent state*. In addition, subsequent readers running on CPU 0 cannot acquire a reference to Schrödinger's cat because there is no longer a path to the cat. Therefore, once the updater has observed a context switch on each CPU, there can no longer be any readers referencing the cat—in other words, a *grace period* will have elapsed.

This procrastination procedure is encapsulated in the RCU primitive `synchronize_rcu()`, the simplest implementation of which simply runs in turn on each CPU, thus ensuring each CPU has executed at least one context switch. The operation of `synchronize_rcu()` is shown in Figure 4, where each horizontal double-ended arrow represents an RCU read-side critical section beginning with `rcu_read_lock()` and ending with `rcu_read_unlock()`, and where each circle represents a context switch. The updater, running on CPU 3, first removes the cat's element using `rcu_assign_pointer()` and then invokes `synchronize_rcu()` in order to wait for each CPU to execute a context switch, thereby guaranteeing the completion of all preexisting readers that might have held a reference to the cat. Once `synchronize_rcu()` completes, the updater is free to execute any additional actions required (for example, shutting down associated threads or hardware) and then free the element. Production-quality implementations also provide an asynchronous counterpart, `call_rcu()`, that invokes a specified function at the end of a grace period. During the early part of the grace period, the CPUs might well disagree on the cat's state, which is only fitting for Schrödinger's cat.

In short, `rcu_read_lock()` and `rcu_read_unlock()` do not affect machine state, but they do not need to. Instead, they act on the developer, who is required to follow the convention that RCU readers must not release their CPUs. This form of RCU can therefore be thought of as synchronization via social engineering. (Other synchronization mechanisms also rely on social engineering; examples include prohibiting data races and a specific lock protecting all uses of a given object.)

A "textbook" RCU implementation is quite simple, as can be seen from the 20 lines of code in Figure 5. Furthermore, in an SC (sequentially consistent) environment, only `rcu_read_lock()`, `rcu_read_unlock()`, and `synchronize_rcu()` are required, consisting of only nine lines of code. Production-quality implementations are larger to meet severe performance, scalability, response-time, and energy-efficiency requirements. User-mode RCU implementations are also available,[4] as are preemptible-kernel implementations.[5] (The full Linux-kernel RCU implementation is beyond the scope of this article.[14])

Because no mainstream computing system is SC, however, `rcu_dereference()` must be used when fetching an RCU-protected pointer, and `rcu_assign_pointer()` must be used when mutating an RCU-protected pointer. In SC systems, `rcu_dereference()` and `rcu_assign_pointer()` reduce to a simple load and store, respectively. In C11/C++11, `rcu_deref-`

Figure 6. Deferred deletion via RCU.

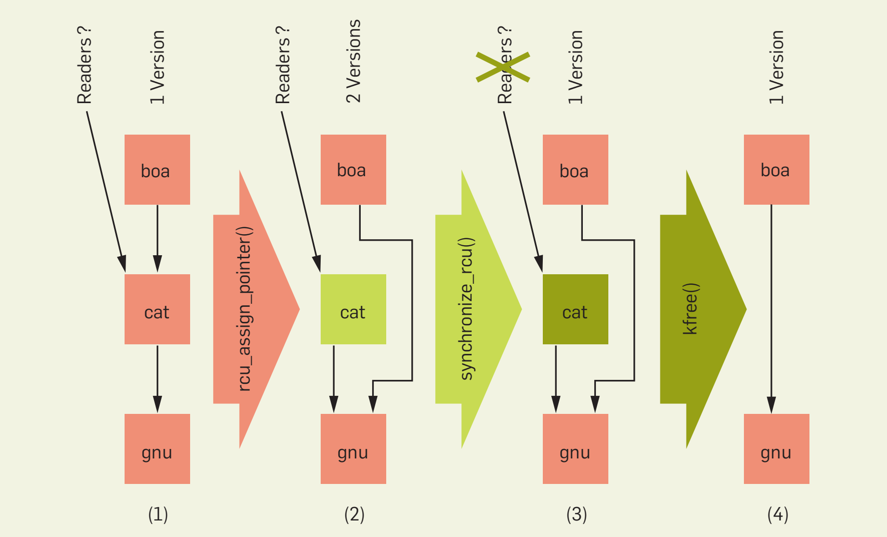


Table 1. Comparison of hazard pointers and RCU ("+" is advantage, "-" is disadvantage).

		Hazard Pointers		RCU
Memory	+	Bounded memory overhead.	-	Blocked readers can exhaust memory (avoid via RCU priority boosting).
	-	Individual hazard-pointer acquisition required for each data element.	+	Single rcu _read _lock() protects all data elements.
Blocking	+	Completely nonblocking.	-	Updaters can block (but can use non-blocking subset of RCU API[4]).
	-	Updates can force concurrent readers to retry.	+	Readers and updaters make unconditional concurrent forward progress, deterministic wait-free readers.
Linearizability	+	Linearizable.	-	Non-linearizable (but often not observable).
	-	Readers require heavyweight memory barriers, but still low overhead.	+	Lower read-side overhead: no read-side memory barriers.

erence() is a load consume, and rcu_assign pointer() is a store release. Figure 6 shows the deletion process in terms of the RCU primitives.

RCU therefore meets its goals of low overhead, read-side memory-allocation avoidance, arbitrarily large scope of protection, and support of cleanup actions. Of course, there is no free lunch: the price of these goals is giving up hazard pointers' nonblocking, bounded-memory, and ordering properties. As can be seen in Figure 7, however, RCU is nonetheless heavily used within the Linux kernel. That said, RCU is specialized, so that locking is used roughly an order of magnitude more heavily than is RCU.

Comparison

This section provides qualitative and quantitative comparisons of hazard pointers and RCU for Schrödinger's zoo. This is followed by some rules of thumb that help determine when to use synchronization via procrastination.

Qualitative comparison. Table 1 compares the properties of hazard pointers and RCU, demonstrating these two synchronization mechanisms represent different design points:

▸ Each advantage is inherently intertwined with a corresponding disadvantage. Hazard pointers' bounded memory overhead advantage over RCU requires developers to carefully acquire a hazard pointer for each data element that readers traverse. This constitutes the corresponding disadvantage versus RCU's ability to protect all data elements with a single rcu _ read _ lock().

▸ Similarly, hazard pointers' nonblocking advantage (coupled with bounded memory) requires updaters to force concurrent hazard-pointer readers to retry their hazard-pointer acquisitions. This retrying constitutes the corresponding disadvantage versus RCU.

▸ Finally, hazard pointers' linearizability advantage requires read-side memory barriers during hazard-pointer acquisition. The resulting reduced read-side performance constitutes the corresponding disadvantage versus RCU. (There is some debate as to whether linearizability is universally useful.[6,22])

Other differences appear to be implementation choices rather than inherent properties of the underlying mechanisms. For example, hazard pointers offer no way to invoke cleanup actions when a given data element is finally reclaimed. This precludes shutting down hardware, threads, or other active components that might be associated with that data element. However, the hazard-pointer's mechanism could be augmented to provide the possibility of RCU-like cleanup actions if desired. Of course, providing cleanup actions would have other consequences, including prohibiting the common usage pattern where hazard pointers are not cleaned up immediately after a traversal. This usage pattern is harmless if no cleanup actions are in place, as it simply retains a small amount of memory that could otherwise be freed. In the presence of cleanup actions, however, this usage pattern could indefinitely defer cleanup, which would have the possibly unacceptable side effect of preventing reuse of the corresponding hardware or thread data.

Restarting traversals is straightforward for simple data structures but can pose a significant software-engineering challenge for large multilinked structures with deeply nested access functions or methods.

Current implementations of hazard pointers record only the beginning of a structure when freeing it, causing difficulties when pointers reference structures nested within other structures. This nesting is quite common in some environments, including the Linux kernel. Extending hazard pointers to handle internal pointers is quite straightforward, however: when freeing a structure, you should pass its size, as well as its address, allowing hazard pointers to that structure's interior to be properly handled.

Because hazard pointers reference only specific structures, races with updates must be handled by restarting the read-side traversal from the beginning. This need to restart stems from the fact that once a structure has been removed, updates no longer change pointers emanating from that structure, so those pointers can no longer be trusted.

Restarting traversals is straightforward for simple data structures but can pose a significant software-engineering challenge for large multilinked structures with deeply nested access functions or methods. Because this article focuses on simple hash tables with chaining, it does not explore these issues. Perhaps traversals can be conveniently restarted using exceptions in

Figure 7. RCU applicability to the Linux kernel.

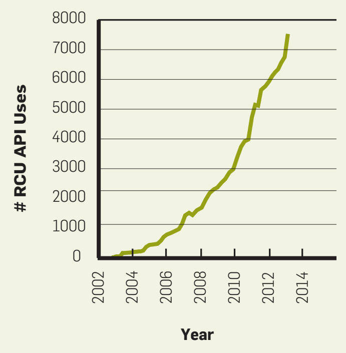


Figure 8. Read-only performance on simple hash table.

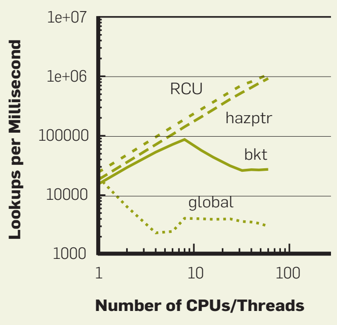


Figure 9. Read-only performance on simple hash table (linear).

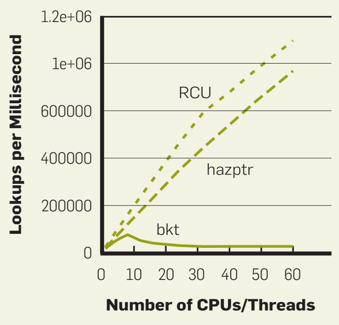


Figure 10. Read-only performance on Schrödinger's hash table.

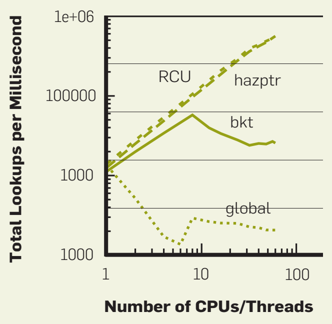


Figure 11. Read-only cat-heavy performance on Schrödinger's hash table.

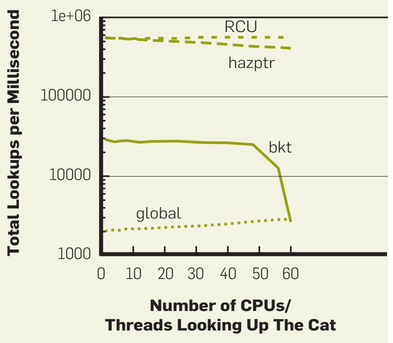
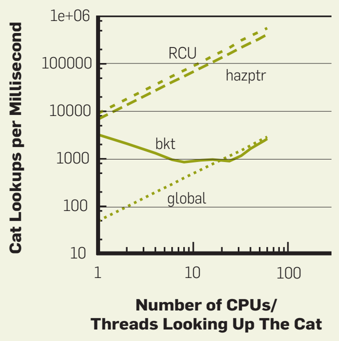


Figure 12. Read-only cat-heavy performance on Schrödinger's hash table: cat queries.



languages supporting them.

In short, the choice between hazard pointers and RCU depends on the workload's requirements, including the other synchronization mechanisms used in the program. For example, if the program were memory constrained, then hazard pointers would likely be the right choice. In contrast, if the program used large multilinked structures with deeply nested access functions or methods, then RCU might be the right choice.

Quantitative comparison. This section presents the results of benchmarks that were run on fixed-size hash tables protected by a single global lock, per-bucket locks, hazard pointers, and RCU. The tests were run on a Westmere-EX x86 system with 32 cores (64 hardware threads) running at 2GHz. To ease comparisons, hazard-pointer updates were protected using per-bucket locking. Blocking was avoided by providing each thread with its own CPU.

The tests used a signal-based RCU variant from liburcu[4] (available on a number of recent Linux distributions, including Debian, Fedora, OpenSUSE, and Ubuntu). This is slower than the zero-cost (QSBR) implementation described earlier, but the zero-cost implementation requires that every thread periodically reside in a quiescent state. Because not all applications are structured to meet this requirement, and because hazard pointers do not require any particular application structure, fairness dictates that the RCU implementation used in these tests also not require any particular structure. For purposes of comparison, the QSBR RCU implementation offers roughly 10% better performance than does signal-based RCU.

Because the hash tables are simple linked structures, hazard pointers are statically allocated and are not explicitly released, thus eliminating the overhead of allocation, free, and release operations. Schrödinger recognizes that other algorithms and data structures may require dynamically allocated hazard pointers, which would reduce their performance.

Figure 8 shows the results of a lookup-only test of a hash table with simple integers for keys. This hash table has 1,024 buckets with chaining, and it contains 1,024 elements out of a total population of 2,048. Lookup keys were randomly selected so that half of the lookups succeeded. As expected, global locking performs quite poorly. Per-bucket locking (*bkt*) scales linearly up to about eight CPUs, then drops off as a result of increasing lock and memory contention as the number of CPUs increases relative to the number of hash buckets. Increasing the number of buckets results in better scalability and performance, as expected. Both hazard pointers (*hazptr*) and RCU scale very nearly linearly with excellent performance.

Figure 9 plots the same data on a linear scale, which more clearly shows RCU's and hazard pointers' performance and scalability advantages compared with per-bucket locking. Note also RCU's inflection at 32 CPUs: hazard pointers take better advantage of this particular system's dual-threaded hardware than does RCU. (The runs using 32 or fewer CPUs run each thread on its own core, while the runs with more CPUs run two threads per core.) Nevertheless, RCU enjoys a 14% performance advantage at 60 CPUs and a 23% performance advantage at 32 CPUs.

Figure 10 shows read-only performance of a prototype of Schrödinger's in-memory database, which is a 1,024-bucket hash table with chaining that uses ASCII strings of up to 31 characters as keys. The results are similar to those for the integer-keyed hash table, except that RCU's advantage over hazard pointers decreases to about 8% at 32 CPUs and to nil at 60 CPUs because of the heavier-weight key operations.

Figure 11 shows the effect of increasing fractions of the queries accessing Schrödinger's cat for 60 CPUs. Global locking performs poorly throughout, as expected. Per-bucket locking's effectiveness is decreased by the increasing level of contention on the hash bucket containing the cat. The decrease becomes catastrophic when more than about 50 CPUs execute cat-related queries. RCU's throughput is unaffected by the fraction of cat-related queries, but hazard pointers' throughput decreases with increasing fractions of cat-related queries, ranging from parity with RCU down to about a 25% performance penalty. This decrease comes as a result of the memory barriers required by hazard pointers. Removing these memory barriers restores performance parity with RCU, but also results in an unsafe hazard-pointer implementation. This behavior suggests the hypothesis that memory barriers are more expensive when multiple CPUs access the same memory locations, which seems plausible given that CPUs accessing disjoint memory locations cannot detect each others' memory-access order.

Figure 12 shows data taken from the same runs as for Figure 11, but shows only the throughput of cat-related queries. The global-lock, hazard-pointer, and RCU results scale as expected: the more the attempted cat-related queries, the more that get done. Per-bucket locking does not fare as well because the contention on the cat's hash bucket increases as the number of attempted cat-related queries increases. Therefore, beyond about 20 CPUs, per-bucket locking performance is similar to global locking.

Table 2. Schrödinger's zoo with updates (operations per millisecond).

Mechanism	Reads	Failed Reads	Cat Reads	Adds	Deletes
Global Locking	799	80	639	77	77
Per-Bucket Locking	13,555	6,177	1,197	5,370	5,370
Hazard Pointers	41,011	6,982	27,059	4,860	4,860
RCU	85,906	13,022	59,873	2,440	2,440

Of course, Schrödinger needs to do updates. Table 2 shows the results of a test with 15 threads doing updates, 15 threads querying the cat, and 30 threads querying random animals. All numbers are events per millisecond, consistent with the earlier figures. Although the cat was always present during this test, the other animals were randomly added and deleted, so there was a 50% probability of any given animal being present at any given time. Therefore, 50% of the non-cat queries could be expected to fail, which was the case here.

Global locking performed poorly, as expected. Per-bucket locking was outperformed by hazard pointers by a factor of three, and hazard pointers were in turn outperformed by RCU by an additional factor of two. In contrast, updates (adds and deletes) slowed by 10% from per-bucket locking to hazard pointers and in turn slowed by an additional 50% from hazard pointers to RCU. This is not unexpected, given that both hazard pointers and RCU intentionally sacrifice update performance in favor of read-side performance. The increase in read-side performance is much larger than the decrease in update-side performance, compared with per-bucket locking.

This does raise the question of whether the low read-side performance of per-bucket locking and hazard pointers was in fact caused by their faster update rates. Additional testing therefore throttled per-bucket locking and hazard-pointer update rates to that of RCU. Per-bucket locking read-side throughput did increase in response to the decrease in update-side lock contention, but only to about 17,000 updates per millisecond, which is nowhere near hazard pointers' 41,011 reads per millisecond, let alone RCU's 85,906 reads per millisecond (zero-cost RCU achieves roughly 100,000 reads per millisecond). Hazard-pointer read-side throughput did not change significantly in response to the throttled update rates.

The throughput of both hazard pointers and RCU is quite a bit lower than in the read-only workload shown in Figure 10. This is because the updates result in read-side cache misses, reducing throughput.

An additional question remains: namely, why RCU's throughput is double that of hazard pointers in this benchmark, given that there was at most a 25% difference in the read-only tests. Removing hazard pointers' read-side memory barriers increased throughput to about 80,000 reads per millisecond, demonstrating that memory barriers were the culprit. This in turn suggests that memory-barrier overhead is an increasing function of cache-miss rate, so the presence of updates increases the cost of the hazard-pointer mechanism.

The data in this section clearly demonstrates the substantial read-side performance benefits of synchronization via procrastination mechanisms such as hazard pointers and RCU. RCU's update-side performance benefits have been demonstrated elsewhere.[4]

When to procrastinate. Although the read-side performance benefits of both hazard pointers and RCU can be sizable, these are specialized mechanisms that are typically used in conjunction with other mechanisms. This raises the question as to when they should be used. Extensive use of RCU in the Linux kernel[15] has led to the following rules of thumb, which may also apply to hazard pointers:

1. Procrastination works extremely well in read-mostly situations where disagreement among readers is acceptable. What constitutes "read-mostly" depends on the workload, but 90% reads to 10% updates is a good rule of thumb.
2. Procrastination works reasonably well in read-mostly situations where readers must always agree.
3. Procrastination sometimes works well in cases where the number of reads and updates are approximately equal and where readers must agree.
4. Procrastination rarely works well in update-mostly situations where readers must agree. There are currently two known exceptions to this rule: providing existence guarantees for update-friendly mechanisms; and providing low-overhead wait-free read-side access for real-time use.

Note that the traditional definition of *read* may be generalized to include writes. An example within Schrödinger's application would be if each animal's data structure included an array of per-thread cache-aligned variables that count accesses to that animal, thus allowing Schrödinger to evaluate his cat's popularity. This works because these read-side writes do not conflict. Further generalization is not only possible, but also heavily used in practice—for example, permitting conflicting writes that are protected by some other synchronization mechanism.[1] This is permissible because reference counters, hazard pointers, and RCU all permit a wide range of code in their read-side critical sections, including atomic operations, transactions, and locking.

Schrödinger's application permits the traditional definition of *read* and thus falls into the first rule of thumb. It is therefore eminently suitable for synchronization via procrastination. These rules will continue to be refined as experience accumulates. In particular, better mechanisms are needed for update-heavy situations, and there is some promising work in progress in this area.[11,20] Finally, Table 1 might be a first step toward rules of thumb for choosing between hazard pointers and RCU.

Conclusion

This article has presented an overview of synchronization via procrastination, discussing some of the consequences, such as read-side disagreements on current state, and how these consequences are actually a faithful reflection of the reality external to the computer. This is to be expected: there is simply no synchronization mechanism that can enforce mutual exclusion on any significant fraction of the physical universe. Use of the two popular procrastination mechanisms described here—hazard pointers and RCU—has increased over the past few decades and can be expected to increase further as the use of multicore systems increases.

Acknowledgments

Thanks to Steven Rostedt for the analogy between RCU and Schrödinger's cat (https://lkml.org/lkml/2013/3/29/248); to Maged Michael for discussions of hazard pointers and RCU; and to Arjan van de Ven, Peter Sewell, and Susmit Sarkar for discussions on concurrency and memory ordering. I am indebted to Samy Al Bahra, Eduardo Bacchi Kienetz, Seth Jennings, Wade Cline, Mathieu Desnoyers, Ram Pai, Kent Yoder, and Phil Estes for feedback on this article and to Jim Wasko for his support.

References

1. Arcangeli, A., Cao, M., McKenney, P.E. and Sarma, D. Using read-copy update techniques for System V IPC in the Linux 2.5 kernel. In *Proceedings of the 2003 Usenix Annual Technical Conference*, 297–310.
2. Becker, P. Working draft, standard for programming language C++; http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2011/n3242.pdf.
3. Corbet, J. `ACCESS_ONCE()`, 2012; http://lwn.net/Articles/508991/.
4. Desnoyers, M., McKenney, P.E., Stern, A., Dagenais, M.R. and Walpole, J. User-level implementations of read-copy-update. *IEEE Transactions on Parallel and Distributed Systems* 23 (2012), 375–382.
5. Guniguntala, D., McKenney, P.E., Triplett, J. and Walpole, J. The read-copy-update mechanism for supporting real-time applications on shared-memory multiprocessor systems with Linux. *IBM Systems Journal 47*, 2 (2008), 221–236.
6. Haas, A., Kirsch, C.M., Lippautz, M. and Payer, H. How FIFO is your concurrent FIFO queue? In *Proceedings of the Workshop on Relaxing Synchronization for Multicore and Manycore Scalability*, (2012).
7. Hart, T.E., McKenney, P.E., Brown, A.D. and Walpole, J. Performance of memory reclamation for lockless synchronization. *Journal of Parallel and Distributed Computing 67*, 12 (2007), 1270–1285.
8. Heisenberg, W. Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. *Zeitschrift für Physik 43*, 3–4 (1927), 172–198. English translation in *Quantum Theory and Measurement*. J.A. Wheeler and W.H. Zurek, eds.
9. Hennessy, J.P., Osisek, D.L. and Seigh II, J.W. Passive serialization in a multitasking environment. Technical Report U.S. Patent 4,809,168 (lapsed), 1989.
10. Herlihy, M., Luchangco, V. and Moir, M. The repeat offender problem: A mechanism for supporting dynamic-sized, lock-free data structures. In *Proceedings of 16th International Symposium on Distributed Computing* (2002), 339–353.
11. Jacobi, C., Slegel, T. and Greiner, D. 2012. Transactional memory: Architecture and implementation for IBM System z. *The 45th Annual IEEE/ACM International Symposium on MicroArchitecture*, (2012); http://www.microsymposia.org/micro45/talks-posters/3-jacobi-presentation.pdf.
12. Kung, H.T. and Lehman, Q. Concurrent manipulation of binary search trees. *ACM Transactions on Database Systems 5*, 3 (1980), 354–382.
13. Massalin, H. Synthesis: an efficient implementation of fundamental operating system services. Ph.D. thesis, 1992, Columbia University, New York, NY.
14. McKenney, P.E. The RCU API, 2012; http://lwn.net/Articles/418853/.
15. McKenney, P.E., Boyd-Wickizer, S., Walpole, J. RCU usage in the Linux kernel: One decade later, 2013; http://rdrop.com/users/paulmck/techreports/RCUUsage.2013.02.24a.pdf.
16. McKenney, P.E. and Slingwine, J.D. Read-copy-update: using execution history to solve concurrency problems. In *Parallel and Distributed Computing and Systems* (1998), 509–518.
17. Michael, M.M. Hazard pointers: safe memory reclamation for lock-free objects. *IEEE Transactions on Parallel and Distributed Systems 15*, 6 (2004), 491–504.
18. Michael, M.M. and Scott, M.L. Correction of a memory management method for lockfree data structures. Technical Report TR599, 1995.
19. Schrödinger, E. Die gegenwärtige Situation in der Quantenmechanik. *Naturwissenschaften* 23: 807-812; 823-828 (1935), 844–949. English translation: http://www.tuhh.de/rzt/rzt/it/QM/cat.html.
20. Sutton, A. Concurrent programming with the Disruptor, 2013; http://lca2013.linux.org.au/schedule/30168/view_talk.
21. Valois, J.D. Lock-free linked lists using compare-and-swap. In *Symposium on Principles of Distributed Computing*, (1995), 165–172.
22. Vogels, W. Eventually consistent. *Commun. ACM* 52 (2009), 40–44.

Paul E. McKenney is a Distinguished Engineer in IBM's Linux Technology Center, where he maintains the Linux kernel's RCU implementation and contributes to userspace RCU. Prior to that, he worked on Sequent's DYNIX/ptx kernel. He blogs at http://paulmck.livejournal.com/ and works on the book *Is Parallel Programming Hard, and, If So, What Can You Do About It?* at http://kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html.

DOI:10.1145/2483852.2483866

Article development led by acmqueue
queue.acm.org

Exploring an alternative to lock-based synchronization.

BY SAMY AL BAHRA

Nonblocking Algorithms and Scalable Multicore Programming

REAL-WORLD SYSTEMS WITH complicated quality-of-service guarantees may require a delicate balance between throughput and latency in order to meet operating requirements in a cost-efficient manner. The increasing availability and decreasing cost of commodity multicore and many-core systems make concurrency and parallelism increasingly necessary for meeting demanding performance requirements. Unfortunately, the design and implementation of correct, efficient, and scalable concurrent software is often a daunting task.

Nonblocking synchronization may be used in building predictable and resilient systems while avoiding the problems of lock-based synchronization. This class of synchronization is not a silver bullet, especially outside of the abstract models in which its performance properties were originally defined. Several of the performance bottlenecks and error conditions associated with lock-based synchronization remain (albeit in more obfuscated forms); therefore, ensuring correctness requires more complex verification methods, and in some cases nonblocking algorithms require the adoption of complex support systems. Stemming from these complexities, nonblocking synchronization is frequently the victim of hype, fear, uncertainty, and doubt. This article aims to equip the reader with the knowledge needed to identify situations that benefit from nonblocking synchronization.

Common Principles

Before elaborating on typical motivations for adopting nonblocking data structures, this section highlights some important principles for understanding scalability on multiprocessor systems in the traditional threading model. Both lock-based and nonblocking synchronization have performance characteristics that are a function of these principles. Practitioners who are already intimately familiar with cache coherent multiprocessors, out-of-order execution, and the design and implementation of lock-based synchronization objects may wish to skip this section. While this section is by no means exhaustive, additional references have been provided. Paul E. McKenney outlines a methodology for choosing appropriate lock-based synchronization mechanisms,[15] and other common principles have been described in previous articles.[5,13,17,]

Contention inhibits scalability. Contention on shared objects in parallel applications is a primary impediment to scalability and predictability. Regardless of the higher-level synchronization facilities being used, contention over a shared object involves some form of serialization by the underlying

ILLUSTRATION BY OLEH VOINILOVYCH

runtime environment, whether it is a language runtime or a shared-memory multiprocessor.

Shared mutable state and contention. Understanding the mechanisms that provide coherency guarantees on multiprocessors is a prerequisite to understanding contention on such systems. Cache coherency protocols are the prominent mechanisms for guaranteeing the eventual consistency of shared state across multiple cache coherent processors. These protocols implement coherency guarantees at the cache-line level, the unit that caches write to and read from main memory (at least from the point of view of the coherency mechanism).

The three prominent cache coherency protocols on commodity processors—MESI, MESIF, and MOESI—are named after the states they define for cache lines: Modified, Owned, Exclusive, Shared, Invalid, or Forward states. The cache coherency protocol manages these states to guarantee the eventual consistency of shared memory with the help of a memory-coherence mechanism. Figure 1 illustrates the state machine associated with the MOESI protocol.[1] You may interpret the probe transitions as those triggered by external memory accesses originating from other cores.

Figure 2 illustrates the life cycle of shared reads and writes in a cache coherent multiprocessor. The state machine associated with this life cycle has been simplified for brevity. This program spawns a thread that reads a modified variable.

The example assumes that the coherency mechanism being used is bus snooping, which allows any processor to monitor any memory accesses to shared locations. The initial state of the system is presented in Figure 3. There are two sockets, each with one core and

Figure 1. The MOESI state machine.[1]

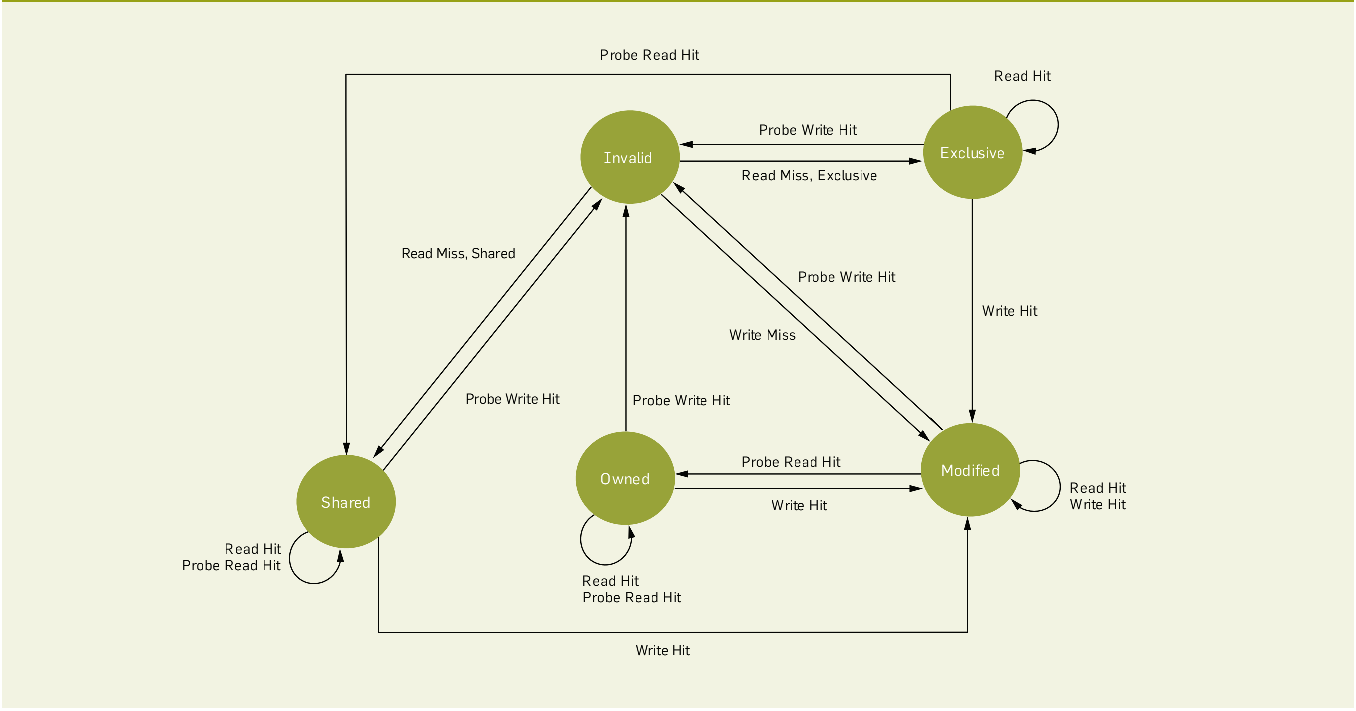


Figure 2. Reading modified variable on remote processor.

```
volatile int x = 443;

static void *
thread(void *unused)
{
      fprintf(stderr, "Read: %d\n", x);
      return NULL;
}

int
main(void)
{
      pthread_t a;

      x = x + 10010;
      if (pthread_create(&a, NULL, thread, NULL) != 0) {
             exit(EXIT_FAILURE);
      }

      pthread_join(a, NULL);
      return 0;
}
```

a 256-byte L2 direct-mapped write-back cache with 64-byte cache lines.

The process initially executes on core 0. Assume the address of variable x is 0x20c4. The increment of the value 10010 to x (x = x + 10010;) may decompose into three operations:

1. Load x from memory into a CPU register.
2. Increment the register by 10010.
3. Store value of register into the memory location of x.

The address of x is 0x20c4 and is hashed to cache line 3. Since this cache line is in a modified state and contains data from a different address (0x00c0), it must be written back to main memory. No other socket contains the cache line for 0x20c4, so a 64-byte block (starting from 0x20c0) is read into cache line 3 from main memory and set to the exclusive state (see Figure 4).

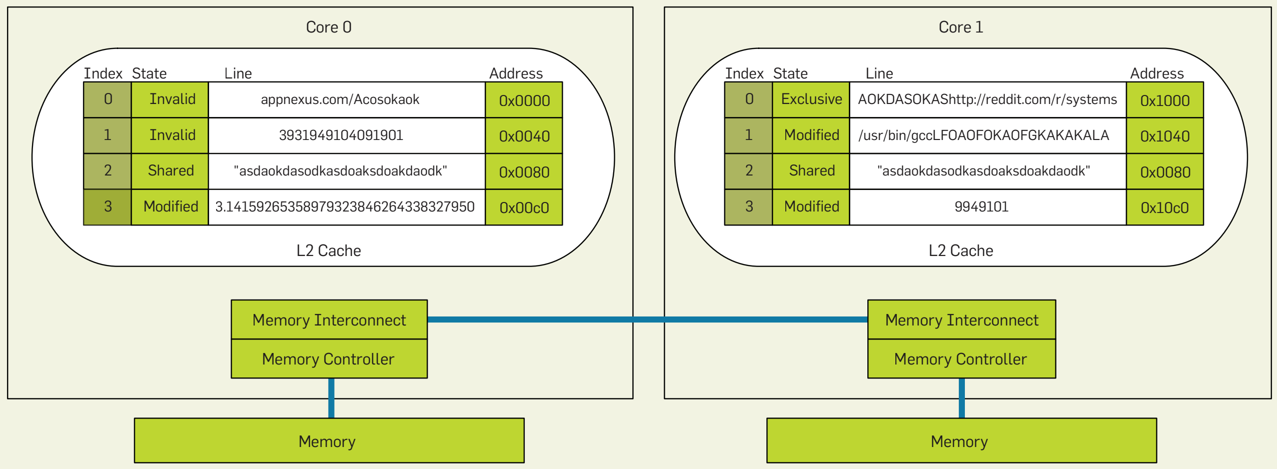
The store into x transitions the cache line from exclusive to modified state.

The new thread spawns and eventually begins executing the thread function. Assume this execution occurs on core 1 (See Figure 5). The thread function executes a load from the address of x as an input to the fprintf function call. This load issues a read probe, requiring a transition from the modified state to the owned state. MOESI allows for cache-to-cache transfer of data, an advantage if probe latency is lower than latency to main memory. In this specific configuration, this operation involves a probe on what is typically a higher-latency memory interconnect (higher latency to intrasocket cache access). The latency associated with the probe is also a function of the topology. On larger-scale machines with asymmetric interconnect topologies, substantial performance mismatch may exist on memory accesses; some sockets may require one hop for an invalidation cycle, while others may require two or more.

In Figure 6, a cache-friendly program (one able to retain shared state in cache with minimal coherence traffic) scales well on modern cache coherent multiprocessors. Mutations to actively shared state lead directly to contention as cache controllers mediate ownership of cache lines.

Coherency granularity. As mentioned earlier, the cache line is the granularity at which coherency is maintained on a cache coherent multiprocessor system. This is also the unit of contention. If logically disparate mutable objects share

Figure 3. Initial state of system.



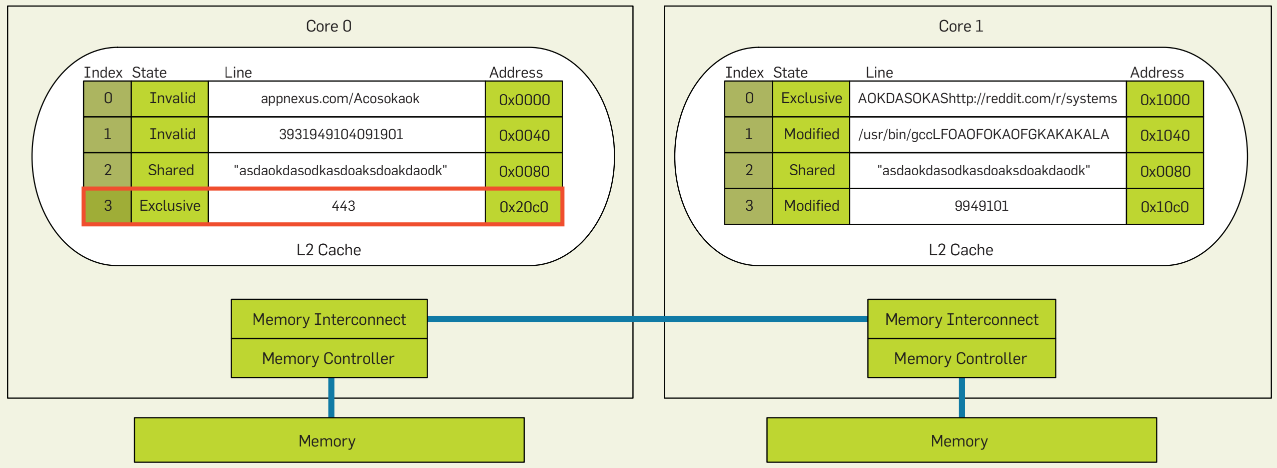
Core 0

Index	State	Line	Address
0	Invalid	appnexus.com/Acosokaok	0x0000
1	Invalid	3931949104091901	0x0040
2	Shared	"asdaokdasodkasdoaksdoakdaodk"	0x0080
3	Modified	3.14159265358979323846264338327950	0x00c0

Core 1

Index	State	Line	Address
0	Exclusive	AOKDASOKAShttp://reddit.com/r/systems	0x1000
1	Modified	/usr/bin/gccLFOAOFOKAOFGKAKAKALA	0x1040
2	Shared	"asdaokdasodkasdoaksdoakdaodk"	0x0080
3	Modified	9949101	0x10c0

Figure 4. After initial load of variable x.



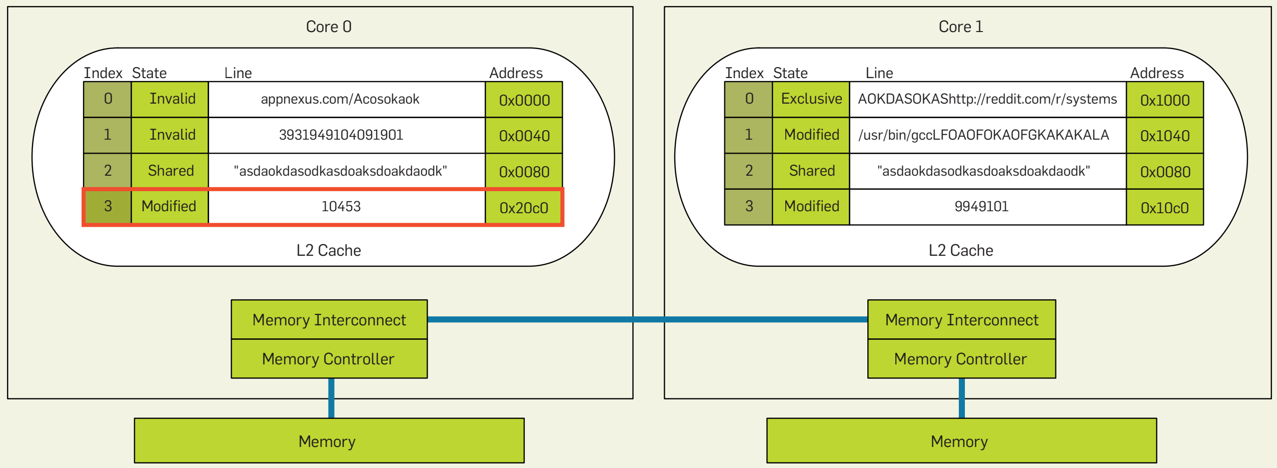
Core 0

Index	State	Line	Address
0	Invalid	appnexus.com/Acosokaok	0x0000
1	Invalid	3931949104091901	0x0040
2	Shared	"asdaokdasodkasdoaksdoakdaodk"	0x0080
3	Exclusive	443	0x20c0

Core 1

Index	State	Line	Address
0	Exclusive	AOKDASOKAShttp://reddit.com/r/systems	0x1000
1	Modified	/usr/bin/gccLFOAOFOKAOFGKAKAKALA	0x1040
2	Shared	"asdaokdasodkasdoaksdoakdaodk"	0x0080
3	Modified	9949101	0x10c0

Figure 5. After store to x.



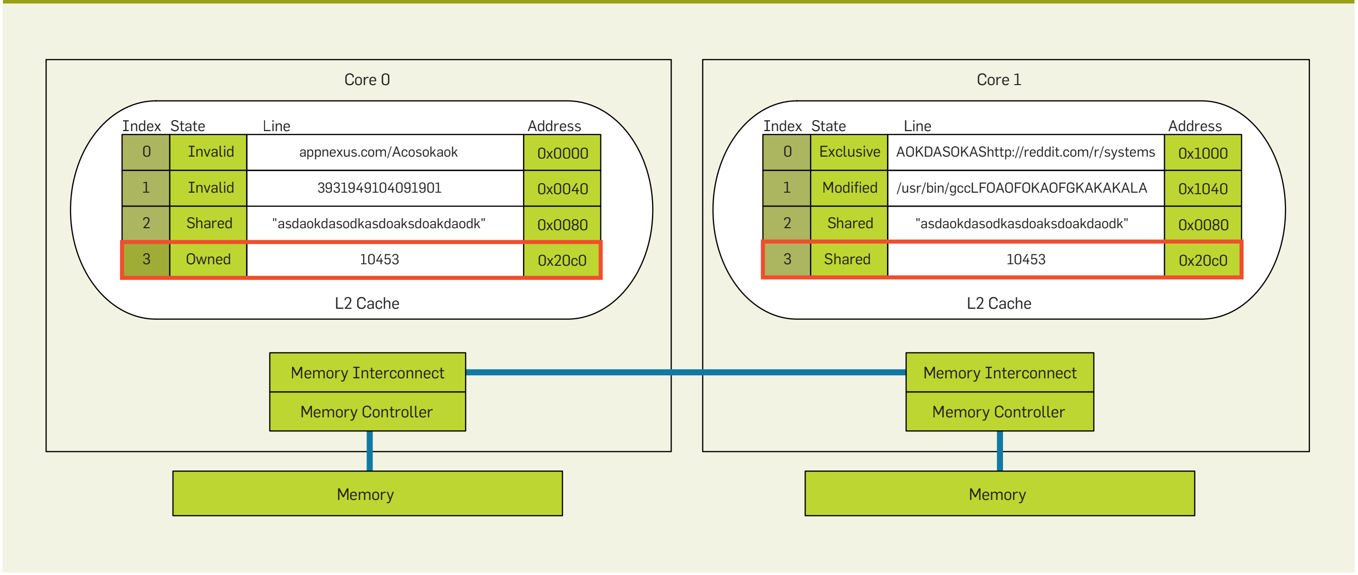
Core 0

Index	State	Line	Address
0	Invalid	appnexus.com/Acosokaok	0x0000
1	Invalid	3931949104091901	0x0040
2	Shared	"asdaokdasodkasdoaksdoakdaodk"	0x0080
3	Modified	10453	0x20c0

Core 1

Index	State	Line	Address
0	Exclusive	AOKDASOKAShttp://reddit.com/r/systems	0x1000
1	Modified	/usr/bin/gccLFOAOFOKAOFGKAKAKALA	0x1040
2	Shared	"asdaokdasodkasdoaksdoakdaodk"	0x0080
3	Modified	9949101	0x10c0

Figure 6. After remote read of *x*.


Core 0
Index State Line Address
0 Invalid appnexus.com/Acosokaok 0x0000
1 Invalid 3931949104091901 0x0040
2 Shared "asdaokdasodkasdoaksdoakdaodk" 0x0080
3 Owned 10453 0x20c0
L2 Cache
Memory Interconnect
Memory Controller
Memory
Core 1
Index State Line Address
0 Exclusive AOKDASOKAShttp://reddit.com/r/systems 0x1000
1 Modified /usr/bin/gccLFOAOFOKAOFGKAKAKALA 0x1040
2 Shared "asdaokdasodkasdoaksdoakdaodk" 0x0080
3 Shared 10453 0x20c0
L2 Cache
Memory Interconnect
Memory Controller
Memory

the same cache line, then operations on these objects exhibit contention and may become a scalability bottleneck. This is called *false sharing*.

Consider the C snippet in Figure 7. Each thread is provided a unique index (UNIQUE_THREAD_ID) into an array of counters that each thread reads from and writes to. Since these counters are laid out sequentially in memory and cache, cache-line ownership ping pongs between all processors incrementing the counter value. This vicious cycle of invalid/shared/modified/owned cache-line transitions is an example of the kind of cache-line *ping-ponging* that can occur as a result of false sharing.

One way of avoiding this problem is to guarantee, at most, one counter is in a given cache line by padding to a cache-line size. Assuming the host processor of the application has 64-byte cache lines, the following modification avoids false sharing:

```
struct counter {
        unsigned long long value;
        char pad[64 -
        sizeof(unsigned long
        long)];
};
```

In the same vein, padding should be done only when necessary. It is still important to keep access patterns in mind. For example, if two mutexes are acquired in succession on the hot path (that is, the most frequently executed segment of code), then cache-line padding does little but drive up lock-operation latency. Excessive padding may impact the overall memory footprint of the application, causing increased pressure on memory resources. Additional approaches exist to detect and minimize false sharing.[14,15]

Unavoidable costs of ordering and visibility guarantees. Certain correctness requirements call for the use of synchronization instructions that are heavier-weight than atomic loads and stores. To improve performance, many modern processors batch memory operations and/or provide some form of instruction-level parallelism. The presence of these optimizations may require the use of memory barrier (or serializing) instructions to enforce ordering of memory operations and their intended side effects. (The term *memory barrier* can be used interchangeably with *memory fence*.) The need for barrier instructions is dependent on the processor memory model, which also defines the reordering possibilities of memory operations.

Consider the source-code snippet in Figure 8. In this pseudocode, a producer thread executing on a dedicated processor creates a message and then signals a dedicated consumer thread on another processor by updating `ready`. With no memory barriers, it is possible for the processor executing `consume` to receive or process the remote store to `ready` before the remote store to `message->value` completes and/or is made visible to other processors. To guarantee that `produce` commits the store to `message->value` before the `ready = 1` operation, a memory barrier may be necessary. Then, to guarantee the reception of the message in `consume` after the consumer has observed `ready` in a nonzero state contains any memory updates before the accompanying `ready = 1` operation (in `produce`), another memory barrier may be necessary (See Figure 9).

Some processors have specialized barrier instructions that guarantee the partial ordering of only some memory operations (examples include serializing only stores with respect to each other or only loads with respect to each other). The requirement of memory barriers is dependent on the underlying memory model. Again, Paul Mc Kenney offers additional details on these memory models.[15]

To make matters worse (in terms of complexity), some languages allow their respective compilers and runtime environments to reorder operations. Until recently, even C and C++ lacked a concurrent memory model (such that the semantics of concurrent accesses were very much compiler and environment specific). Many programming languages may exclusively emit memory barriers through acquire/release semantics. Popular lock-based synchronization mechanisms hide the details of memory barriers from program designers by having implicit heavyweight memory barriers. This simplifies concurrent programs as developers are left to reason with serializability,[3] but it sometimes comes at the cost of reduced performance.

Memory barriers are also necessary for many other concurrent algorithms, especially classes of algorithms that do not rely on locks and their ordering guarantees. Ultimately, the ability to avoid heavyweight synchronization mechanisms depends on the correctness and visibility requirements of the data structure. Recently, the requirement of expensive synchronization instructions for certain correctness constraints in the presence of common access patterns has been formalized.[4]

Atomic read-modify-write operations are expensive. The cost of atomic read-modify-write operations depends on the underlying architecture and actual programming language implementation. TSO (total store ordering) is becoming an increasingly common memory model on commodity processors. Among other things, this memory model defines atomic operations as having a total ordering. This guarantee comes at a significant cost, even in the absence of contention. Table 1 illustrates non-contending throughput of atomic CAS (compare-and-swap) operations (`lock cmpxchg`) versus regular CAS operations (`cmpxchg`). The atomic CAS can provide total ordering and atomicity guarantees across multiple processors, while the regular CAS operation cannot. These measurements were made on an Intel Core i7-3615QM at 2.30 GHz and include the overhead of register spillage.

On TSO architectures, these atomic operations imply heavyweight memory-barrier semantics. Even on architectures with weaker memory models, atomic operations may be implemented in terms of complex instructions that incur the cost of long dependency chains in the pipeline. In other cases, programming languages may themselves define a total ordering to these instructions, which may lead to unnecessarily heavyweight memory fences being emitted. Atomic operations should be used only if necessary. On most architectures, atomic read-modify-write instructions are expensive (relative to other instructions)—even in the absence of contention.

Before deciding on the use of atomic operations, take into account the progress guarantees of the actual atomic implementation. For example, some architectures relying on LL/SC (load-linked/store-conditional primitives) such as ARM or Power architectures may still leave an application sensitive to external disturbances (jitter) such as preemption, even in the absence of contention.

Take advantage of fetch semantics provided by the atomic operation arsenal. If a platform provides an atomic fetch-and-increment operation or a CAS operation that returns the previous value, take full advantage of these semantics (see Figure 10).

Be wary of topology. The latency and throughput properties of memory accesses vary according to the type of memory access. For example, accessing memory in a remote cache may be cheaper than accessing local uncached memory. This performance asymmetry is an example of a non-negligible NUMA (non-uniform memory access) factor. The higher the NUMA factor, the higher this asymmetry. Optimizing the placement of objects that are shared

Table 1. The cost of atomic read-modify-write operations.

Operation	Throughput (operations/second)
Atomic CAS	147,304,564
CAS	458,940,006

Figure 7. A program that exhibits false sharing.

```
#define N_THR 8

struct counter {
        unsigned long long value;
};

static volatile struct counter counters[N_THR];

void *
thread(void *unused)
{
        while (leave == false) {
                counters[UNIQUE_THREAD_ID].value++;
        }

        return NULL;
}
```

Figure 8. A program lacking necessary memory barriers.

```
volatile int ready = 0;

void
produce(void)
{
      message = message_new();
      message->value = 5;
      message_send(message);
      ready = 1;
}

void
consume(void)
{
      while (ready == 0) {
            ; /* Wait for ready to be non-zero. */
      }

      message = message_receive();
      result = operation(&message->value);
}
```

across cores minimizes NUMA effects. For example, if shared state is accessed by a cluster of threads on a single socket, then there is no reason for that object to be contained in remote memory.

Synchronization objects that are not NUMA-aware may also be susceptible to NUMA effects. These effects manifest not only as a fast-path performance mismatch between cores but also as starvation or even livelock under load.

Fair locks guarantee starvation-freedom at the cost of increased preemption-sensitivity. Variants of these locks also allow for scalable point-to-point wake-up mechanisms. Recently, lock cohorting was developed as a generalized methodology for allowing NUMA awareness in locks at the cost of a higher fast-path latency.[8]

Predictability matters. Predictability is a desirable property in high-performance concurrent applications requiring stringent latency or throughput bounds. The progress guarantees of synchronization mechanisms define behavior in the presence of contention and unforeseen execution delays (which can include jitter). If a program is designed solely for the fast path, then minor disturbances in execution and/or workload may compound to result in unpredictably low performance.

Systems relying on blocking synchronization can be especially sensitive to execution delays. A significant delay in one thread holding a synchronization object leads to significant delays for all other threads waiting on the same synchronization object. Examples of such execution delays include process preemption and timer interrupts. The significance of these delays depends on the latency constraints of the application. If the application is required to be in the microsecond latency range, then even a network interrupt may be significant. On general-purpose processors and operating systems, isolating all external sources of delays is difficult. If there are strict latency requirements for a program relying on blocking synchronization in the fast path, consider minimizing external disturbances (real-time scheduling policies and interrupt rerouting are common). For the time scales where every disturbance counts, specialized real-time operating systems and hardware exist to address those challenges.

Figure 9. Utilizing memory barriers for visibility and ordering.

```
volatile int ready = 0;

void
produce(void)
{
        message = message_new();
        message->value = 5;
        message_send(message);

        /*
         * Make sure the above memory operations complete before
         * any following memory operations.
         */
        MEMORY_BARRIER();
        ready = 1;
}

void
consume(void)
{
        while (ready == 0) {
                ; /* Wait for ready to be non-zero */
        }

        /*
         * Make sure we have an up to date view of memory relative
         * to the update of the ready variable.
         */
        MEMORY_BARRIER();
        message = message_receive();
        result = operation(&message->value);
}
```

Nonblocking Synchronization

Before looking at reasons for adopting nonblocking data structures, this section introduces some terminology and briefly examines the complexities associated with the practical design, implementation, and application of nonblocking data structures.

In the literature, nonblocking synchronization and nonblocking operations fall into three primary classes of algorithms, each with a unique set of progress guarantees:

- *OF (Obstruction-Freedom).* The algorithm provides single-thread progress guarantees in the absence of conflicting operations.
- *LF (Lock-Freedom).* The algorithm provides systemwide progress guarantees. At least one active invocation of an operation is guaranteed to complete in a finite number of steps. There is no guarantee of starvation-freedom.
- *WF (Wait-Freedom).* The algorithm provides per-operation progress guarantees. Every active invocation of an operation completes in a finite number of steps. In the absence of overload, there is a guarantee of starvation-freedom.

There is a total ordering to these classes of algorithms such that any wait-free algorithm is also lock-free and obstruction-free; any lock-free algorithm is also obstruction-free. A nonblocking data structure implements operations that satisfy the progress guarantees in the nonblocking hierarchy (summarized earlier) for some (or an infinite) level of concurrency.[11] It is possible to implement a data structure that provides a nonblocking progress guarantee for a limited number of readers or writers. A traditional example of this is the Michael Scott's two-lock queue, which provides wait-free progress guarantees in the presence of up to one concurrent enqueue and one concurrent dequeue operation.[19]

It is also possible for a data structure to implement different progress guarantees for different sets of operations. Examples include a wait-free enqueue operation with a multiconsumer-blocking dequeue operation, as seen in the URCU (userspace read-copy-update) library (http://lttng.org/urcu), and a bounded-FIFO (first-in first-out) with a single-producer wait-free enqueue and multiconsumer lock-free dequeue, as seen in the Concurrency Kit library (http://concurrencykit.org/).

State-space explosion. Nonblocking data structures rely on atomic loads and stores or more complex atomic read-modify-write operations. The mechanism used depends on the level of concurrency the data structures intend to support and their correctness requirements.[4,11] In lock-based synchronization, it is usually sufficient to reason in terms of locking dependencies and critical sections (and read-side or write-side critical sections for asymmetric lock-based synchronization such as read-write locks).[3] When reasoning about the correctness of concurrent algorithms in the absence of critical sections, such as in nonblocking algorithms, the number of execution histories involving the interaction of shared variables can be enormous. This state space can quickly become infeasible for a mere human being to exhaust strictly through state-based reasoning. Consider the following program:

```
int x = 1;
int y = 2;
int z = 3;
void
function(void)
{
        /*
         * This returns a
         unique integer value
         * for every thread.
         */
        int r = get_thread_id();
        atomic_store(&x, r);
        atomic_store(&y, r);
        atomic_store(&z, r);
}
```

Assuming this program is executed by two threads under a memory model that implements TSO, there are approximately 20 possible execution histories, if you take only the three store operations into consideration—and consider them completely uniform and deterministic. With four threads, 369,600 distinct execution histories exist. A derivation yields that for N deterministic processes with M distinct actions, there are $(NM)! / (M!)^N$ execution histories.[21] Coupled with programming-language reordering possibilities, processor-memory reordering possibilities, and out-of-order execution, the state space will grow even more quickly in complexity.

Figure 10. Avoiding unnecessary read-write cycles.

```
int shared_flag = 0;

void
worse(void)
{
        int old_value = 0;

        /*
         * Under contention, this may involve a write invalidation
         * followed by another transition to shared state (at least
         * two probes).
         */
        while (atomic_cas(&shared_flag, old_value, 1) == false) {
                old_value = atomic_load(&shared_flag);
        }
}

void
better(void)
{
        int old_value = 0;
        int snapshot;

        while (true) {
                /*
                 * We generate a single write cycle to retrieve the
                 * value we are comparing against. This can reduce
                 * cache coherency traffic by at least 50% compared
                 * to the previous worse() usage pattern.
                 */
                snapshot = atomic_cas_value(&shared_flag, old_value, 1);

                /* Operation has completed, exit loop. */
                if (old_value == snapshot)
                        break;

                old_value = snapshot;
        }
}
```

Verifying the correctness of nonblocking algorithms, especially those of the wait-free and lock-free class, requires a good understanding of a program's underlying memory model. The importance of understanding the memory model cannot be understated, especially if the model itself is not precisely defined or allows for machine- and/or compiler-dependent behavior (as is the case for languages such as C or C++). What may seem like memory-model minutiae may be a violation in the correctness of your program.

Figure 11 takes a closer look at the C snippet. Assume that `send_to_consumer` consists solely of loads and stores and no serializing instructions. On x86 processors that implement TSO, stores to memory locations are committed in order with a few exceptions. Assuming that the `memset` function consists solely of regular store instructions, then the assertion in `consumer_thread` should not fail. The reality is that certain instructions, operations, and memory types are not guaranteed to comply with the x86 TSO model.[20] In the example, the `memset` function can be implemented with high-performance non-temporal and/or vectorized store instructions that violate the usual memory-ordering semantics. It would be necessary to determine whether the compiler or standard library implementation and target platform guarantees memory-store serialization with respect to `memset`. Fortunately, most (but not all) popular implementations of `memset` that use these instructions emit a memory barrier of some kind (by silent contract), but this is not a requirement. If you are designing lock-less concurrent programs for a low-level language, be prepared to explore similar dark recesses of implementation-defined behavior.

Correctness. The most common correctness guarantee for nonblocking data structures and operations is

Figure 11. Implemention-defined behavior of `memset`.

```
void
producer_thread(void)
{
      message_t *message;

      message = message_create();
      memset(message, 0, sizeof *message);
      send_to_consumer(message);
}

void
consumer_thread(void)
{
      message_t *message;

      message = message_receive();

      /*
       * The consumer thread may act in an erroneous
       * manner if any of the contents of the message
       * object are non-zero.
       */
      assert(message->value == 0);
}
```

linearizability. This requires that operations appear to complete atomically at some point between their invocation and completion. The *linearization point* of an operation is the instant in which the operation appears to have been completed atomically. It helps to reason in terms of linearization points with respect to the sequential specification of the data structure. For reasons described earlier, proving the linearizability of an algorithm is often difficult.

Specializing a nonblocking data structure for a fixed level of concurrency can greatly reduce the complexity of the state space. Considering the potential size of the state space of nonblocking concurrent algorithms, however, more advanced methods of testing and validation are necessary for verification.

The woes of unmanaged languages. Languages that lack built-in support for automatic memory management such as C or C++ require specialized techniques for managing dynamically allocated lock-less objects. (This section may not be relevant to those who plan on working exclusively with languages that have automatic memory management such as Java or C#.)

A popular scheme for reducing contention in lock-based synchronization involves decoupling the liveness of an object from its reachability. For example, a concurrent cache implementation using lock-based synchronization may have a mutex protecting a cache but a separate mutex protecting concurrent accesses for objects contained within the cache. This scheme is commonly implemented using in-band reference counters as illustrated in Figure 12.

This scheme allows for concurrent accesses to objects fetched from the cache while still allowing for concurrent fetches to proceed. The scheme works in the current example because the reachability of the object is managed atomically with respect to the reference counter. In other words, there is never a state in which the reference counter is 0 and the object is still in the cache. The object is never destroyed if there are still references to it. The reference counter in this case provides a mechanism for allowing the program to safely reclaim memory associated with concurrently accessed objects whose reachability is determined by cache state.

On modern processors, nonblocking data structures are implemented in terms of atomic operations that can modify only one target memory location at a time. Assume that the previous cache example was actually transformed to become lock-free. The reachability of an object in the cache is likely determined by a linearization point consisting of an atomic operation to a single memory location. For this scheme to work, the reachability of the object must be managed atomically with respect to concurrent `cache_lookup` operations. In the absence of atomic operations that operate on disparate memory locations, this common in-band reference-counting scheme is unable to provide sufficient safety guarantees.[7]

For this reason, dynamic nonblocking and lock-less data structures (structures that access dynamically allocated memory that may also be freed at runtime) must typically rely on alternative safe memory reclamation mechanisms. Besides full-fledged garbage collection, safe memory reclamation techniques include:

- *EBR (Epoch-Based Reclamation)*[9] is a passive scheme that allows for safe memory reclamation by carefully monitoring observers of a global epoch counter. It is unsuitable for protecting objects that are created and destroyed at a high frequency. Threads attempting synchronous (immediate) object destruction using EBR may block until the mechanism detects a safe destruction point. It is possible to implement variants of this scheme with minimal cost on the fast path.
- *HP (Hazard Pointers)*[18] is a scheme that works through the live detection of references to objects that require safe memory reclamation. To take advantage of it, nonblocking algorithms may require modification that is specific to this scheme. However, it is more suitable for protecting objects that are created and destroyed at a high frequency, and threads wanting to destroy protected objects are not required to block for an unbounded amount of time. This scheme can come at a high cost (full memory barrier) on the fast path and may not be suitable for traversal-heavy workloads.
- *QSBR (Quiescent-State-Based Reclamation)*[6,16] is a passive scheme that allows for safe memory reclamation through the detection of quiescent states in a program, in which no references to objects with live references could possibly exist. It is not suitable for protecting objects that are created and destroyed at a high frequency. Threads attempting synchronous (immediate) object destruction using QSBR may block until the mechanism detects a safe destruction point. It is

possible to implement variants of this scheme with zero cost on the fast path.

▸ *PC (Proxy Collection)* is an out-of-band amortized reference-counting scheme.

Additional mechanisms exist to achieve safe memory reclamation as well.[12] Thomas Hart et al. compared the performance properties of the first three schemes.[10] Note that these mechanisms may have attractive performance properties compared with reference counting for many workloads and are not required to be used exclusively with nonblocking data structures.

Resilience. Systems that require any of the following properties may benefit from the use of lock-free and wait-free algorithms:

▸ *Deadlock-Freedom.* The absence of locking means that wait-free and lock-free data structures are immune to deadlock conditions, which can be difficult to avoid in large and complex locking hierarchies.

▸ *ASYNC-Signal Safety.* Providing deadlock-freedom and coherency in the context of asynchronous interruptions of a critical section is a nontrivial problem. Wait-free and lock-free synchronization is immune to these problems.

▸ *Termination Safety.* Wait-free and lock-free operations that satisfy linearizability may be aborted at any time. This means that processors or threads that are in the middle of wait-free and lock-free operations may terminate without sacrificing the overall availability of a system.

▸ *Preemption Tolerance.* With wait-freedom, the completion of any operation is guaranteed to occur in a finite number of steps in the absence of resource overload, even in the presence of preemption and other external delays. Lock-freedom guarantees the overall progress of a system, as a delay in one thread can only incur a bounded delay in another thread (linearizability may require other threads to assist in the completion of the delayed operation, which may require additional instructions on the fast path).

▸ *Priority Inversion Avoidance.* In the absence of overload to resources such as memory, wait-freedom can provide strong bound guarantees for priority inversion, but this may at times come at a significant cost to the fast path. This additional overhead can be avoided with algorithms specialized for a fixed level of concurrency. Lock-freedom can avoid priority inversion on uniprocessor systems at a lower overhead than lock-based synchronization with the right scheduling policies.[2] On multiprocessor systems with high levels of contention, bounded priority inversion is difficult to avoid (the extent to which it can be avoided is a function of the fairness and prioritization guaran-

Figure 12. Traditional in-band reference counting scheme.

```
cache_t cache;
object_t *object;

object_t *
cache_lookup(int key)
{
      object_t *object;

      lock(cache);

      /* Returns object associated with that key. */
      object = cache_find(key);

      /* No object found associated with that key. */
      if (object == NULL) {
            unlock(cache);
            return NULL;
      }

      /*
       * Increment reference counter which was initialized to
       * 1 when inserted into cache.
       *
       * The actual reference counter is contained or is in-band
       * the actual object it is reference counting.
       */
      atomic_increment(&object->ref);
      unlock(cache);
      return object;
}

/* Remove an object from cache. */
void
object_delete(object_t *object)
{
      lock(cache);
      cache_remove(object);
      unlock(cache);

      /* Remove the cache reference. */
      object_delref(object);
}

void
object_delref(object_t *object)
{
      int new_ref;

      /*
       * Atomically decrements the integer pointed to
       * by the argument and returns the newly decremented
       * value.
       */
      new_ref = atomic_decrement(&object->ref);
      if (new_ref == 0) {
            /*
             * If the reference count is 0 then the object
             * is no longer reachable and is not currently
             * being used by other threads, so it is safe to
             * destroy it.
             */
            free(object);
      }
}
```

tees provided by the underlying coherence mechanism, which usually provides little to no guarantee on either). Lock-freedom remains vulnerable to unbounded priority inversion even in the absence of overload to memory resources, as a lower-priority thread may still cause any number of delays of a higher-priority thread with interfering operations. Contention avoidance may be used to prevent this situation.

Bridging the gap from abstract models. It is important to frame the progress guarantees of nonblocking synchronization in the context of real hardware. The wait-freedom progress guarantee can break down in real-world systems under severely high levels of contention over memory and coherency resources. At these levels of contention, it is possible to starve processors from acceptable rates of operation completion. The progress guarantees of your program can be only as good as those of the concurrency primitives it is built on and of the underlying coherency mechanism. Fortunately, as interconnect bandwidth continues to increase and interconnect latency continues to decrease, the severity of this problem dwindles as well. Lock-based synchronization is, of course, also susceptible to this problem. In the absence of overloading of memory resources and in the presence of jitter, wait-freedom can provide stronger latency-bound guarantees than lock-based synchronization (because of the increased tolerance to external delays). This latency bound is a function of the underlying microarchitecture, memory resources, and contention.

The fast-path latency of write operations on a nonblocking data structure is usually higher than the fast-path latency of a lock-based variant. This trade-off is especially common for nonblocking objects designed to work for any level of concurrency because they usually require one or more heavier-weight atomic read-modify-write operations on the fast path. This trade-off in fast-path latency is a function of the cost of an underlying platform's lock implementations and the complexity of the nonblocking algorithm.

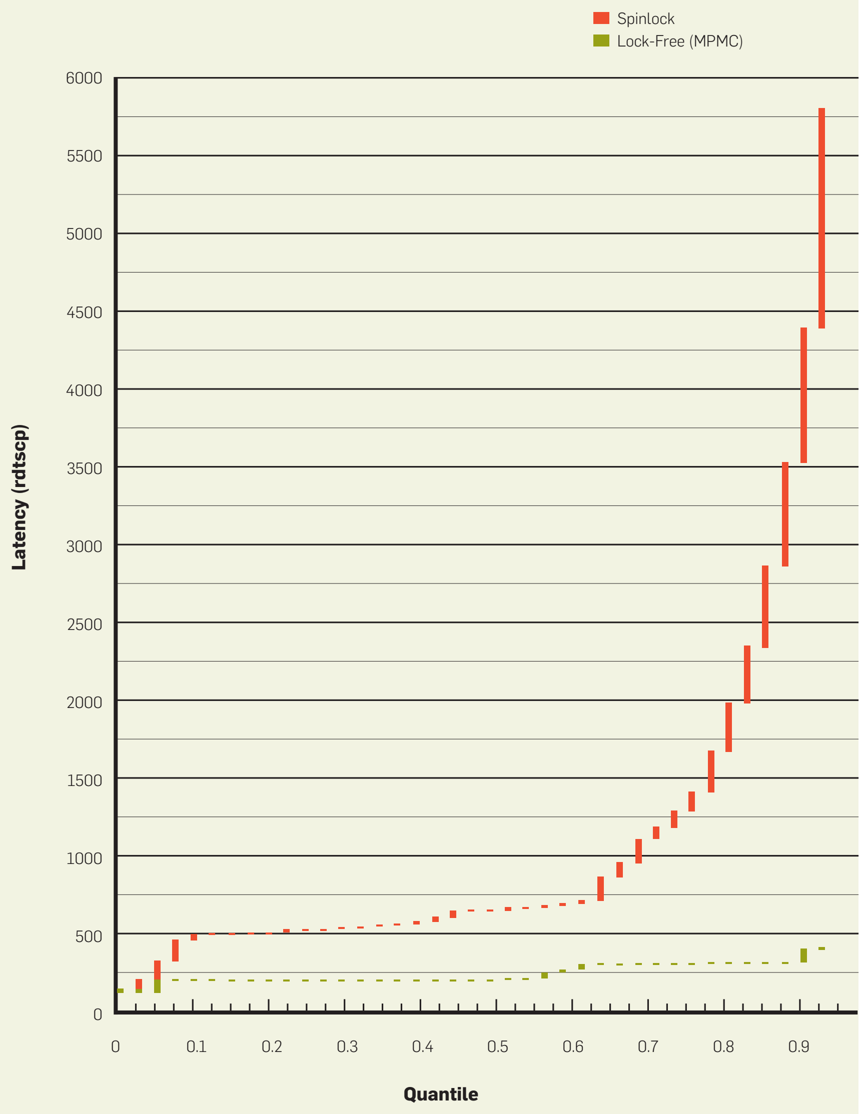
This concept can be illustrated by comparing the performance of a spinlock-protected stack with a lock-free stack (the implementation used was `ck_stack.h` from concurrencykit.org). The lock-free stack contains a single `compare_and_swap` operation for both the push and pop operations. On the x86 architecture, the fast-path latency of the spinlock-backed stack implementation is significantly lower than a lock-free stack implementation. On the Power 7 architecture, this is not the case. Table 2 displays uncontested latency (in ticks) of these various operations.

On the x86 it is possible to implement a spinlock using significantly cheaper atomic operations (in this case, the `xchg` instruction) than the ones used in the lock-free stack. The TSO of the architecture does not require the use of explicit memory barriers for this algorithm. On the other hand, the lock-free stack implementation requires the

Table 2. Uncontested latency of various operations.

Operation	Intel Core i7-3615QM	IBM Power 730 Express
spinlock_push	17	29
lockfree_push	25	12
spinlock_pop	18	29
lockfree_pop	27	12

Figure 13. Latency distribution of a push-only workload.



more complex cmpxchg (and cmpxchg16b) instruction, which exhibits higher baseline latency.

On the Power architecture, both the spinlock and lock-free implementations rely on the same underlying LL/SC primitives. The primary difference is the spinlock implementation requires a heavyweight memory barrier on invocation of every stack operation, while the nonblocking stack requires lighter-weight load/store memory barriers.

Regardless of the architecture-imposed trade-off in latency, the desirable properties of nonblocking data structures are revealed under contention. Figure 13 depicts the latency distribution of a push-only workload on a single stack across four threads on an x86 server (Intel Xeon L5640). Active measures were taken to avoid jitter, including use of the SCHED _ FIFO scheduling class, core affinity, and IRQ (interrupt request) affinity (Linux 2.6.32-100.0.19.el5).

The latency distribution illustrated in Figure 13 is *not* a result of stronger latency bound on individual stack operations, but a side effect of stronger *systemwide* progress guarantees provided by lock-free algorithms. Specifically, lock-free algorithms are able to guarantee systemwide progress in the presence of preemption. In the blocking data structure, a preempted or blocked thread inside the stack-operation critical section prevents systemwide progress. On the other hand, the lock-free stack only forces a stack operation retry if another thread has made progress by updating the stack. If all sources of jitter were removed from the test system, then the latency profile of the spinlock-based stack would prevail.

Asymmetric workloads involving a combination of write- and read-only operations can also benefit from nonblocking synchronization. Often, in combination with the safe memory reclamation techniques described earlier, it is possible to provide strong guarantees of progress at minimal to no cost of heavyweight synchronization instructions on the fast path. This is a desirable property: it avoids many of the write-side/read-side fairness problems exhibited in popular single-preference read-write locks, while also avoiding fast-path degradation associated with heavier-weight fair read-write locks.

Conclusion

Having examined some of the common principles of parallel programming on multiprocessor systems and the real-world implications of nonblocking synchronization, this article has aimed to equip readers with the background needed to explore alternatives to lock-based synchronization. Even though the design, implementation, and verification of nonblocking algorithms are all difficult, these algorithms are becoming more prevalent among standard libraries and open-source software. The information presented here helps identify situations that call for the resiliency and performance characteristics of nonblocking synchronization.

Acknowledgments

Thanks to Mathieu Desnoyers and Paul McKenney for their contributions in lock-less synchronization to the open source community. Thanks to Devon H. O'Dell and Gabriel Parmer for feedback on this article. Thanks to Theo Schlossnagle for his support in the early stages of this process. Thanks to Silvio Cesare, Wez Furlong, Maxime Henrion, David Joseph, Paul Khuong, Abel Mathew, Shreyas Prasad, Brendon Scheinman, Andrew Sweeney, and John Wittrock for their helpful feedback. Thanks to Jim Maurer, the *ACM Queue* Editorial Board, Matt Slaybaugh, and Susie Holly.

Related articles on queue.acm.org

Real-World Concurrency
Bryan Cantrill and Jeff Bonwick
http://queue.acm.org/detail.cfm?id=1454462

Unlocking Concurrency
Ali-Reza Adl-Tabatabai, Christos Kozyrakis and Bratin Saha
http://queue.acm.org/detail.cfm?id=1189288

Software and the Concurrency Revolution
Herb Sutter and James Larus
http://queue.acm.org/detail.cfm?id=1095421

References

1. AMD. *AMD64 Architecture Programmer's Manual, 2: System Programming.* Publication 24593, 2007.
2. Anderson, J.H., Ramamurthy, S. and Jeffay, K. Real-time computing with lock-free shared objects. *ACM Transactions on Computer Systems 15*, 2 (1997), 134–165; http://doi.acm.org/10.1145/253145.253159.
3. Attiya, H., Ramalingam, G. and Rinetzky, N. Sequential verification of serializability. *SIGPLAN Notices 45*, 1 (2010): 31–42; http://doi.acm.org/10.1145/1707801.1706305.
4. Attiya, H., Guerraoui, R., Hendler, D., Kuznetsov, P., Michael, M.M. and Vechev, M. Laws of order: Expensive synchronization in concurrent algorithms cannot be eliminated. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (2011), 487–498; http://doi.acm.org/10.1145/1926385.1926442.
5. Cantrill, B. and Bonwick, J. Real-world concurrency. *Queue 6*, 5 (2008), 16–25; http://doi.acm.org/10.1145/1454456.1454462.
6. Desnoyers, M., McKenney, P.E., Stern, A.S., Dagenais, M.R. and Walpole, J. User-level implementations of read-copy update. *IEEE Transactions on Parallel and Distributed Systems 23*, 2 (2012); 375–382.
7. Detlefs, D.L., Martin, P.A., Moir, M., Steele Jr., G.L. Lock-free reference counting. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing* (2001), 190–199; http://doi.acm.org/10.1145/383962.384016.
8. Dice, D., Marathe, V.J. and Shavit, N. Lock cohorting: A general technique for designing NUMA locks. In *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*: (2012), 247–256; http://doi.acm.org/10.1145/2145816.2145848.
9. Fraser, K. Practical lock freedom. Ph.D. thesis. Computer Laboratory, University of Cambridge (2003). Also available as Technical Report UCAM-CL-TR-639, Cambridge University.
10. Hart, T.E., McKenney, P.E., Demke Brown, A. and Walpole, J. Performance of memory reclamation for lockless synchronization. *Journal of Parallel and Distributed Computing 67*, 12 (2007), 1270–1285; http://dx.doi.org/10.1016/j.jpdc.2007.04.010.
11. Herlihy, M. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems 13*, 1 (1991), 124–149; http://doi.acm.org/10.1145/114005.102808.
12. Herlihy, M., Luchangco, V. and Moir, M. The repeat offender problem: a mechanism for supporting dynamic-sized lock-free data structures. Technical Report. Sun Microsystems Inc, 2002.
13. Herlihy, M. and Shavit, N. *The Art of Multiprocessor Programming.* Morgan Kaufmann Publishers Inc., San Francisco, CA, 2008.
14. Kandemir, M., Choudhary, A., Banerjee, P. and Ramanujam, J. On reducing false sharing while improving locality on shared memory multiprocessors. In *Proceedings of the 1999 International Conference on Parallel Architectures and Compilation Techniques*, 203–211. IEEE Computer Society; http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=807529&contentType=Conference+Publications.
15. McKenney, P.E. Selecting locking primitives for parallel programming. *Commun. ACM 39*, 10 (1996), 75–82; http://doi.acm.org/10.1145/236156.236174.
16. McKenney, P.E. Exploiting deferred destruction: An analysis of read-copy-update techniques in operating system kernels. Ph.D. dissertation. Oregon Health and Science University. AAI3139819, 2004.
17. McKenney, P.E. Is parallel programming hard, and, if so, what can you do about it? kernel.org (2011); https://www.kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html.
18. Michael, M.M. Hazard pointers: Safe memory reclamation for lock-free objects. *IEEE Transactions on Parallel and Distributed Systems 15*, 6 (2004), 491–504; http://dx.doi.org/10.1109/TPDS.2004.8.
19. Michael, M.M., Scott, M.L. Simple, fast, and practical nonblocking and blocking concurrent queue algorithms. Technical Report, 1995. University of Rochester, NY.
20. Owens, S., Sarkar, S., Sewell, P. A better x86 memory model: x86-TSO. *Theorem Proving in Higher Order Logics.* S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, eds, 2009, 391–407. Springer, Berlin Heidelberg; http://dx.doi.org/10.1007/978-3-642-03359-9_27.
21. Schneider, F.B. 1997. *On Concurrent Programming.* Springer-Verlag, New York, 1997.

Samy Al Bahra is an engineering team lead at AppNexus, playing a key role in the development of a leading real-time online advertising platform. Previously, he was involved with Message Systems in the development of a high-performance messaging server and was the lead developer of the George Washington University High Performance Computing Laboratory UPC I/O library reference implementation. He is also the maintainer of Concurrency Kit (http://concurrencykit.org).

DOI:10.1145/2483852.2483868

Article development led by acmqueue
queue.acm.org

Nonblocking synchronization can yield astonishing results in terms of scalability and real-time response, but at the expense of verification state space.

BY MATHIEU DESNOYERS

Proving the Correctness of Nonblocking Data Structures

SO YOU HAVE decided to use a nonblocking data structure, and now you need to be certain of its correctness. How can it be achieved?

When a multithreaded program is too slow because of a frequently acquired mutex, the programmer's typical reaction is to question whether this mutual exclusion is indeed required. This doubt becomes even more pronounced if the mutex protects accesses to only a single variable performed using a single instruction at every site. Removing synchronization improves performance, but can it be done without impairing program correctness?

Whether this feat can be achieved—and whether it can be extended to algorithms involving more complex data structures—depends on the relationship of the variable to the rest of the program. It also depends on the compiler, architecture, and operating system details, as well as other interesting aspects discussed throughout this article.

Nonblocking data structures[27] can be used to communicate between threads without using mutual exclusion or other synchronization that would otherwise make a thread block awaiting another thread. This article looks at what makes nonblocking data-structure design and implementation tricky, and it surveys modeling techniques and verification tools that can provide valuable feedback on the correctness of those algorithms.

What Makes Nonblocking Data-Structure Programming Tricky?

There are many aspects to consider when programming nonblocking data structures, including the language and architecture memory models, atomicity, ordering, linearizability, and performance.

Memory models. Unless the programmer provides explicit key words or synchronization hints, programming languages such as C, C++, and Java presume that a single thread performs variable accesses, leaving behavior of nonsynchronized concurrent data accesses on multiprocessor systems either undefined or not well understood by programmers. With multicore and multiprocessor computers becoming pervasive, however, it is important to allow concurrent execution to take place. One of the usual ways for providing consistency in concurrent systems is through the use of critical sections and mutual exclusion to ensure serializability,[5] which ends up creating regions of sequential code by excluding other execution threads from accessing critical sections concurrently. Unfortunately, this approach does not result in the best performance in many cases, especially when scalability to many cores is considered. Relaxing sequential execution by shrinking the duration of critical sections, however, increases complexity.

People have commonly used the `volatile` keyword in C and C++[24] to indicate that a variable can be modified outside of its current scope to disable optimizations that may interfere with the correctness of the program. This keyword, however, tells the compiler only to assume the variable could be modified outside of the local thread and that order among `volatile` accesses within a single thread needs to be preserved; it does nothing to prevent reordering from the processor. The ordering guarantees of the `volatile` keyword greatly vary from language to language, and even between language versions: for example, the `volatile` keyword in Java has a much stricter memory-ordering semantic starting from JDK 1.5 than it had in JDK 1.4.[15]

Atomicity. Another possible problem with nonblocking data-structure programming is that an instruction executing atomically on a processor is not sufficient to ensure its effect is made visible to other processors atomically.

Unaligned word-sized memory accesses are a good example: many architectures will allow those to be performed by a single instruction, but there is no guarantee the in-memory result will be updated atomically.

Another example is a nonatomic read-modify-write operation. Although some architectures might end up turning the C `i++` statement into a single instruction, the compiler can very well choose to perform this in three separate instructions: load from memory to register; increment register; store register to memory. The compiler may choose to do so either because it is required by the instruction set (for example, RISC) or simply because register pressure is too high. Moreover, with the Intel x86 instruction set, for example, variables meant to be read, modified, and written atomically by many processors running concurrently need to have a LOCK prefix.[23] Unless special double compare-and-swap or transactional memory instructions are being used, if supported by the architecture, memory accesses that need to touch more than one word-aligned word-sized data structure must be performed in many instructions, and thus nonatomically.

Finally, the compiler is allowed to refetch variables from memory. Therefore, what someone might think will always be performed in a single load operation might not be.

Reordering can be performed at many levels for performance reasons. First, reordering of loads and stores can be performed by many processors. In addition, processors can reorder execution of instructions that do not have interdependencies. Finally, compilers can reorder expression evaluation, statements, and instructions as long as program order is preserved. Unfortunately, these reorderings do not take into account that threads executing concurrently may assume operations performed by other threads will appear in program order from their own points of view. This is why processors provide memory-barrier instructions and compilers provide compiler barriers. These operations limit reordering across the barriers in the instruction and code flows.

It is important to understand that atomicity of a memory access does not necessarily provide ordering. In some architectures, such as x86, the LOCK prefix, used to specify atomic operations, implies a memory barrier. A great many other architectures, however, such as PowerPC,[22] ARM,[2] and MIPS,[28] perform their atomic operations with LL/SC (load-linked/store-conditional) instructions and usually require explicit memory-barrier instructions to provide ordering.

Linearizability. Atomicity and ordering are not necessarily enough to ensure an entire nonblocking data structure will behave in the same way as one that is always accessed sequentially. Nonblocking operations normally contain a linearization point[20] to guarantee correctness with respect to the sequential definition of that operation. Linearization points are the atomic operations that will perform the mutations

PHOTOGRAPH BY BORYA GALPERIN

Figure 1. Empty wait-free concurrent queue.

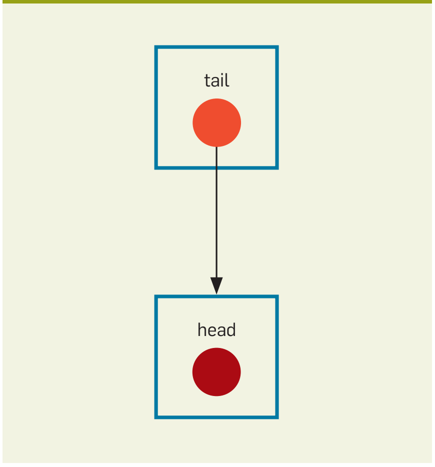


Figure 2. Enqueue first node transient state.

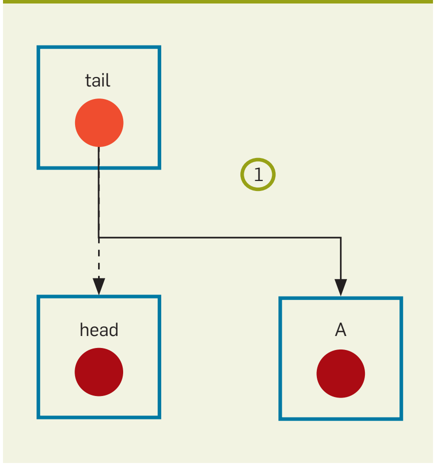


Figure 3. First node enqueue completed.

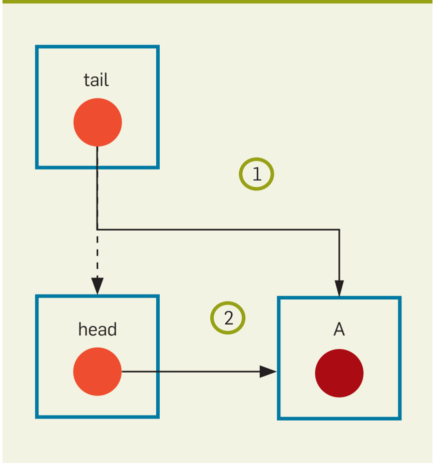
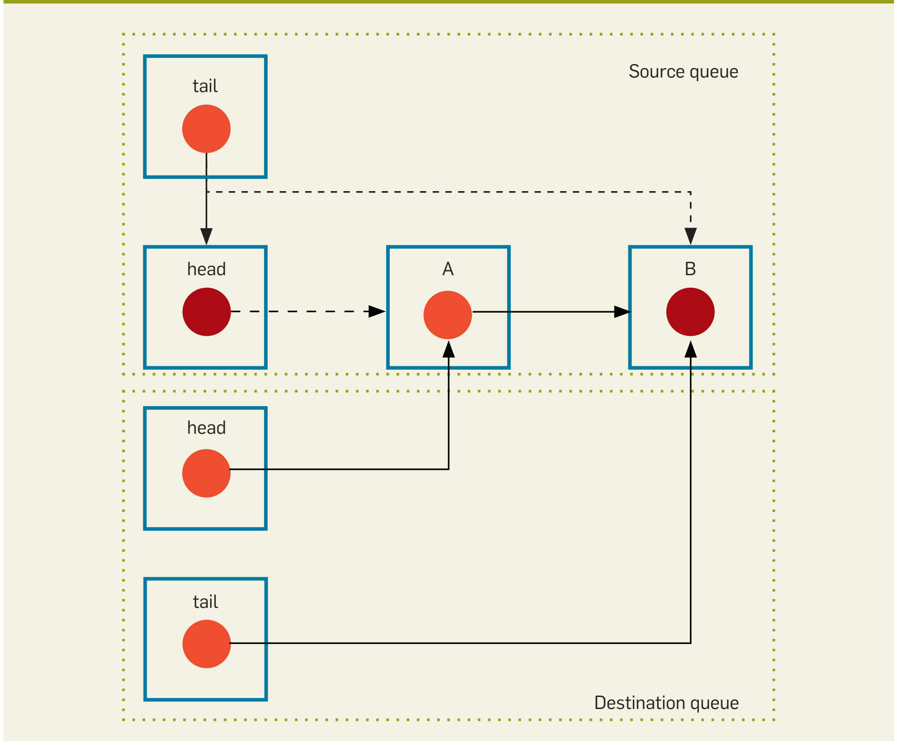


Figure 4. Splice completed.



queue nodes as boxes. Within these boxes, circles represent pointers to the next node. A gray circle is a NULL pointer. Solid arrows represent the target of a pointer, and dashed arrows represent the previous pointer target. Circles containing a number represent the order in which updates are stored in memory.

The empty queue is shown in Figure 1. An enqueue operation is performed in two steps, shown in figures 2 and 3. The first step moves the tail pointer to the next node using an atomic exchange operation. It leads to a transient state during which threads concurrently performing a dequeue, splice, or iteration need to busy-wait. Enqueue is completed by storing the new node's address into the last node's next pointer.

Performing one more enqueue operation adds a new node *B* at the tail, so you end up with a queue that has two nodes, *A* and *B*, leading to the initial state of the source queue presented in Figure 4. The splice operation shown in this figure moves all nodes from the source queue into the destination queue.

Note that an arbitrary number of operations can occur while the queue is in a transient state. This would allow, for example, the enqueue of a second node to complete while the first node is not yet completely enqueued, as shown in Figure 5. When encountering this queue structure, both splice and dequeue operations would need to busy-wait until the first enqueue completes.

Queues in a transient state, as illustrated in Figure 6, raise the possibility of an interesting optimization to the splice operation: instead of busy-waiting when encountering the first node with a NULL next pointer in a non-empty queue, one

necessary to provide the correct externally observable effects with respect to the sequential specification of the data structure simultaneously with validation of operation success or failure. This ensures no globally visible inconsistent operation state lingers when an update operation aborts. This also ensures behavior of the data structure as a whole matches the behavior expected from using the data structure sequentially. It should be understood that reasoning in terms of linearization points has some limitations. For example, it does not consider delay between invocation of a method and execution of its linearization point.[18,25]

The following is a good example of a linearization issue: suppose you have a queue such as the concurrent queue with wait-free enqueue/blocking dequeue (http://lttng.org/urcu). This enqueues nodes at the tail and dequeues from the head of the queue. It provides nonblocking (wait-free) enqueue by requiring threads performing dequeue, splice, and iterations to busy-wait if they find a NULL next pointer in a node that is not the tail of the queue. The dequeue operation dequeues one node at a time, whereas the splice operation moves all nodes from a source queue into a destination queue. To illustrate enqueue and splice operations, figures 1–6 represent could simply assume the queue is empty. After all, if the splice operation is called again after a short time, then it would eventually get the queue content. Unfortunately, even though this approach might seem appropriate at first glance, it would break the linearizability of the queue. Consider the following scenario: one thread *X* is enqueuing node *A*, and another thread *Y* is first enqueuing *B*, then performing a splice operation to grab the entire queue. Given a program that has only those two threads and no other dequeuer thread, thread *Y* should be allowed to expect that if it performs an enqueue operation followed by a splice, the splice would never encounter an empty queue because it *must* contain the element added by the previous enqueue. If thread *X* is preempted while adding the first element into the queue, however, thread *Y*'s splice operation could encounter a queue that would appear empty, even though its sequential specification should allow it to assume the queue contains at least one node.

Performance considerations. Other elements that make nonblocking data-structure programming difficult are the considerations of throughput and scalability associated with atomic accesses and memory barriers. Depending on the design and use of the data structure, it might be cheaper, performance-wise, to hold a lock and perform the operation using a sequence of regular non-atomic instructions than to perform a sequence of more expensive atomic operations with memory barriers of their own. Therefore, performance considerations are largely driving the careful choice of atomic instructions and barriers to implement higher-level operations, thus increasing complexity.

Nonblocking data structures have many interesting properties such as reentrancy and mutual-exclusion deadlock immunity, as well as, in some cases, good scalability and throughput. Implementing them efficiently, however, involves understanding interactions with interruptions and traps (at kernel level), signals (at userspace level), multithreading, scheduler preemption, and thread migration, in addition to the low-level compiler and processor aspects.

Models

As noted in the previous discussion about linearizability, providing counterexamples is a great way of illustrating problems in nonblocking data-structure design and implementation. Designers of nonblocking algorithms should try to find representations that will provide a deeper understanding of the algorithm, helping them to consider as many race scenarios as possible. These representations will not only enhance the designers' understanding of detailed interactions between threads, but will also help in expressing their ideas to others.

Which brings us to code review: encouraging many people to think of different ways in which an algorithm could misbehave, each with his or her own focus and expertise, will likely shed more light on the problem than having just one person look at it from a single point of view. This is a useful trick even for an individual reviewing a nonblocking algorithm: looking at an algorithm over and over, at different moments in the day, in all sorts of contexts, with a plethora of models to represent the algorithm, will help achieve a thorough study from various viewpoints.

This calls for representing algorithms in various ways. Diagrams are a great way of showing the relationship between various objects in a data structure, dependency between memory operations, and the various state transitions that an object can go through.

The concurrent queue with wait-free enqueue/blocking dequeue presented in the previous section illustrates how to use diagrams to represent nonblocking data structures. Others have provided further examples of the states of a data structure represented as diagrams: in the read-copy-update (RCU) linked list and grace period explanation;[12] showing the various states in which a hash table can be found in Cliff Click's hash table explanation;[8] and showing instruction dependencies at the processor level.[10,11]

Many optimizing compilers internally use models to represent dependen-

Figure 5. Enqueue second node over first-node-enqueue transient state.

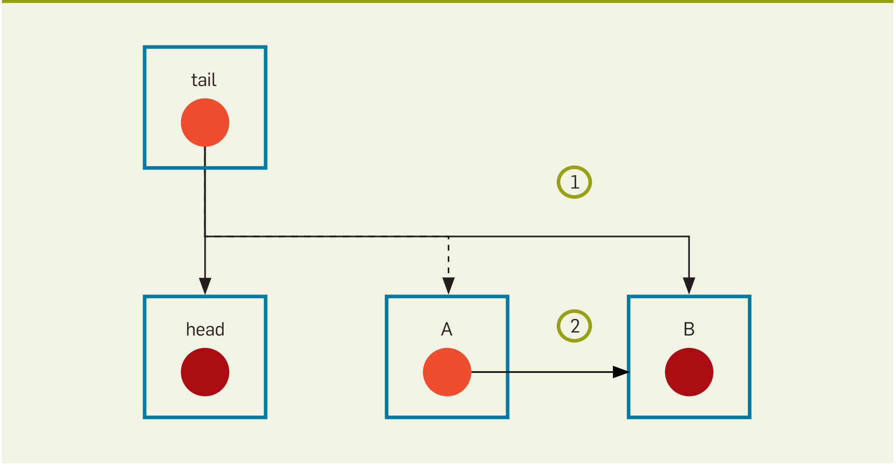
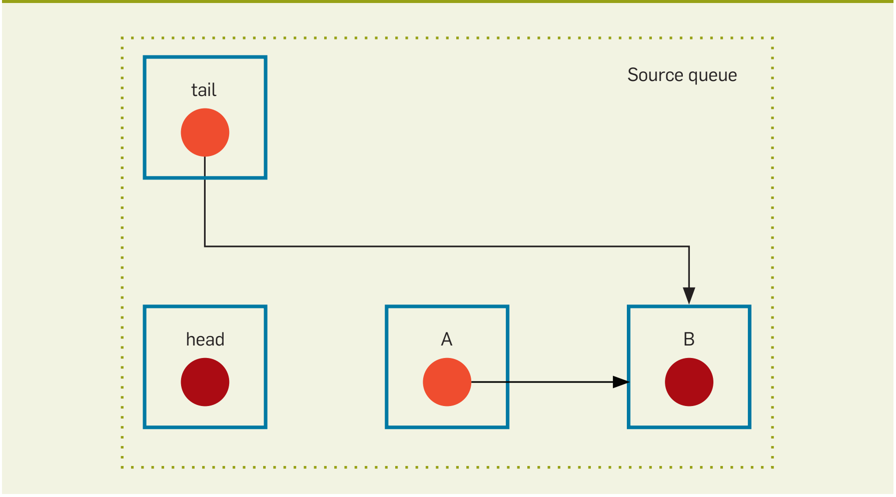


Figure 6. Splice waiting for first-node-enqueue completion.



cies between statements. It helps them move statements around and carry optimizations without changing the behavior as seen from program order. Some of these representations, to name a few, are data flow graph (DFG), which represents the data dependencies between statements (for example, memory accesses or register accesses); control flow graph (CFG), which represents the control dependencies between statements (for example, branches or loops); and a convenient combination of those two, program dependence graph (PDG).[14] Modeling algorithms at the PDG level can be useful for verification, as will be shown in the Testing section.

Representing algorithms as sequences of statements in a programming language is another representation that allows thinking about the code in a more sequential manner. However, it should not be assumed the code executes sequentially. This representation is merely a starting point for considering all possible reorderings that could be performed. As silly as it might sound, writing the code on a computer, with pencil and paper, with or without thorough commenting on the possible reordering at each line, as an initial draft, or recopying from memory are all different ways of interacting with the code that can lead to a better understanding of the reasons why the code is written a certain way.

A convenient way of presenting counterexamples is to demonstrate race scenarios by side-by-side execution sequences of two or more processors or threads. Initial variable states, valid within the specifications of the algorithm, are first detailed, and then portions of the algorithm are examined. This can be achieved by detailing, in program order, execution sequences for each processor or thread involved. Then the order of one or two statements can be altered within the constraints allowed by the compiler and processor. For each modified execution scenario, all invariants imposed by algorithm correctness should be respected. If it is possible to find a scenario that breaks those invariants, then it should be treated as a bug. It might be caused by an algorithm design issue, by missing memory barriers, or by incorrect assumptions about the atomicity of a sequence of operations.

For smaller code snippets, moving to the instruction-level scope to represent key pieces of the algorithm can be worthwhile. This is especially useful when considering aspects regarding memory reordering performed by the processor. It gets the compiler out of the way and lets the reviewer focus on instruction and processor semantics—at the expense of a less compact model.

The issue with nonblocking algorithms is that it is not sufficient to consider the sequentially equivalent high-level algorithm operations as happening one after another: between every instruction within the algorithm, one needs to consider what happens if other processors execute any concurrent operation of the algorithm a possibly infinite number of times. Moreover, when there are no compiler or processor-level memory barriers in place, every reordering allowed within the specification of the architecture needs to be taken into account. This quickly increases the number of concurrent execution flows to consider, which explains why making sure no execution flow can misbehave is hard. When faced with a large number of states to validate, model checkers, presented in the Testing section, can be very helpful in automating tedious and error-prone verification.

Every assumption made in a concurrent piece of code should be revisited with prejudice. It is important to assure these assumptions will hold across hardware memory models, programming-language memory models, and higher-level correctness constraints (such as linearizability). For each assumption made, many attempts should be made to come up with a race, as far-fetched as it may be, that can make the algorithm misbehave.

Model Accuracy

To come up with an accurate model of a nonblocking algorithm, the memory models of all targeted architectures need to be taken into account. At first glance, it might appear that nonblocking algorithms should be specifically tied to a single architecture, but there are methods to model nonblocking algorithms in a way that allows reasoning about their correctness across a large set of architectures.

To model an algorithm in a way that is portable across multiple architectures, you can think of the algorithm as running on a model consisting of all the worst reordering possibilities that can be performed by the set of architectures. Memory models targeting a set of architectures can be found in the Linux kernel (http://www.kernel.org), the Userspace RCU project (http://lttng.org/urcu), and within the Concurrency Kit (http://concurrencykit.org).

The algorithm models should therefore consist of the worst-case reordering that could happen by combining the characteristics of all architectures targeted. Of course, this increases the state space to validate, but the benefit is in having a single model of an algorithm that works on a wide range of architectures. Proving that a single targeted architecture can fail is sufficient to identify an error.

When targeting many architectures, stress testing under all supported architectures is mandatory, because some specific reordering characteristics that would cause the algorithm to misbehave could be specific to only one architecture within the set.

Targeting architectures with weaker memory models will increase the size of the model's state space, because many more reorderings can occur. On the other hand, if architectures with weak memory consistency models are covered, then other architectures with stricter memory consistency models—in every way equally or more strict than previously validated models—will be a given.

This demonstrates a limitation of models: their completeness is unfortunately limited by how accurately they represent architecture behavior. Techniques such as litmus tests can help improve the accuracy of models. Those will be discussed later.

Testing

Notwithstanding the amount of care taken while designing and implementing a nonblocking algorithm, there is always the chance that some characteristics of the processor, compiler, or operating system have gone unnoticed. Therefore, no modeling or review can replace the good old testing approach. Moreover, testing is very effective in discovering issues when porting to a yet unforeseen architecture.

When nonblocking algorithms are being tested, the testing coverage needs to be slightly adapted, compared with its usual definition. Indeed, when a simple sequential algorithm is being tested, covering a large percentage of lines and branches can be a good indicator that tests were thoroughly performed. Unfortunately, this is not sufficient when testing algorithms that can execute in parallel. Testing coverage must check not only that every line of code and branch has been executed, but also that each one was executed in every context of other threads.

This means that it might take a long time for an error to trigger in production if it is caused by a small race window in which two sequences of instructions misbehave when executed concurrently. Therefore, moderate testing can be sufficient for making sure that the common cases work fine, but making sure that corner cases do not misbehave can be quite challenging.

One way of tackling this issue is stress testing. When an error condition takes time to reproduce, focusing on triggering the race over and over for a long period of time can increase the chance of reproducing it more quickly. The main aspect that can be controlled is the frequency at which the race window is executed. Therefore, as the runtime length of the algorithm to stress test increases, the time ratio spent executing each individual race window diminishes.

To accelerate this process, you can use what could be called "oriented stress testing." This entails changing the configuration of the test bench while doing the stress test, based on an understanding of the design, so corner cases are more likely to be hit frequently. For example, to stress test a hash table, keeping a number of buckets purposefully low and testing with thousands of nodes with the same key are both likely to generate long hash chains. This corner case might be difficult to trigger with a large hash table, even after weeks of testing, without oriented stress testing. Besides configuration changes, some tools can help automate oriented testing: CONCURRIT can help specify a deterministic execution order that should be enforced by testing runs;[6] the Relacy tool (http://www.1024cores.net/home/relacy-race-detector) allows controlling the interleaving of atomic operations between threads.

To model an algorithm in a way that is portable across multiple architectures, you can think of the algorithm as running on a model consisting of all the worst reordering possibilities that can be performed by the set of architectures.

Another trick for stress testing is to add random delays within the algorithm, so different execution timings are tested. This can make race windows slightly larger, which can help hit issues more quickly.

There has been a fair amount of research on deterministic multithreading,[4,29] which consists of deterministically fixing the order of critical sections with a scheduler. Regarding correctness, the objective is to obtain repeatable results across runs, thus making bugs easier to reproduce in testing. Even though this leads to acceptable results in some cases, it remains to be seen whether this approach will scale to fine-grained locking and larger multiprocessor systems, while having a low overhead. Limitations of this approach include its requirement for concurrent accesses to use locks, which are more expensive than RCU and nonblocking synchronization; the increase of state-space size needed to track the patterns with the frequency of lock acquisition; the extra communication overhead required at lock acquisition; and the fact that small changes to the source code can significantly change lock access patterns.

When scalability, performance, and real-time response matter, stress testing can yield better results than deterministic multithreading, by using a statistical approach to testing coverage rather than fixing the layout of lock acquisition at runtime.

Model Checking

Wouldn't it be nice if the time-consuming exercise of validating a nonblocking algorithm by manually coming up with counterexamples based on all states that can be reached by concurrent threads for any given state of a thread, and statistically through testing, could be automated? Fortunately, there are approaches that allow this, to some extent.

Formal methods. Model checking undertakes a full state-space search to verify specific assertions of a given model.[3,7] While very powerful, this approach has important limitations. For one, the amount of memory and time required to perform the full state-space search grows very quickly with the size of the state space. Therefore, this approach applies only to relatively small models. This is why keeping nonblocking algorithms very compact in terms of state space is important: it allows easier thorough state-space search. In order to ensure this, nonblocking algorithms should be designed in ways that will keep state space as small as possible.

One way to leverage model checking for nonblocking algorithms is to design algorithms as small building blocks, none of which exceeds the state-space search capacity of current computers. Each algorithm can then provide memory-ordering guarantees through an API, which can then be assumed correct by another nonblocking algorithm that would use it. Therefore, it is possible to compose nonblocking algorithms into quite complex algorithms by cutting the state-space search at their interfaces.

Another limitation of model checking is that the verification is only as accurate as the model and assertions. If the model is too simple compared with the reordering that can be performed by the processor, some errors will not be caught by the checker. Moreover, if assertions do not represent what is intended to be verified, the checker may never find a bug that would be assumed to be caught. This is because model checkers work a little bit like oracles: when everything is fine, they just report that everything went fine. It is only when an assertion fails that they provide a detailed sequence of events that led to the issue.

One way of limiting the risk of modeling errors is error injection into the model. If it is assumed that a given model and assertion should catch one type of error, a slightly altered model that triggers the assertion should be created, just to make sure it catches the error if it is added purposefully.

Model checkers that can be used for nonblocking algorithms include Spin,[21] which allows describing a model in Promela and verifying linear temporal logic (LTL) assertions on that model. The properties to validate can be as simple as an assertion in the code, but the real power of LTL is that it provides temporal operators. It is then possible to validate that a certain state is never reached until another state is reached, for all possible executions.

Other model checkers focus on reproducing architecture behavior at the instruction level. Previous articles have reported on the results of litmus tests on a wide range of architectures to characterize their respective behavior and to formalize their memory model.[1] The authors then created a tool that allows running ARM and Power litmus tests in a verifier (http://www.cl.cam.ac.uk/~pes20/ppcmem/).[30]

The important question regarding model checking of nonblocking algorithms is: What should be modeled?

The important question regarding model checking of nonblocking algorithms is: What should be modeled? In previous research, I proposed a model for compiler, memory, and processor reordering.[10] Its purpose was to allow expressing algorithms in a model that takes into account PDG dependencies at the code and instruction levels. Depending on the accuracy level needed, the complexity of the algorithm, and the resources available to run the model checker, a different degree of detail can be either included or omitted from the models.

Execution-based model checking. Driving the model checker by executing the program to verify is another model-checking approach, known as EMC (execution-based model checking).[13]

Rather than traversing all states required to prove formally that a condition is never false, driving the verification with the states reached by execution of the verified system can significantly limit the number of states to explore, at the expense of completeness of the proof. Therefore, if the execution of the program causes the verified condition to fail, it will be reported. Absence of error does not guarantee validity of the program, however, since it may contain errors in states not reached by its execution.

A good EMC example is found in the lockdep verifier[9] implemented within the Linux kernel. Perhaps its most important verification is performed by constructing a graph of encountered lock dependency chains, and reporting errors if lock usage could trigger a deadlock. It is worth mentioning that lockdep can be adapted to validate locking within arbitrary applications and is therefore at the grasp of any programmer.

EMC can be performed either directly alongside execution of the program or based on a trace of the program execution. This execution trace can act as a container storing the events driving an off-line model checker.

Formal Verification Through Mathematical Proof

Formal mathematical proof is another way of verifying that an algorithm does not contain unwanted characteristics. The idea is to start with a theorem to prove. This typically assumes that some invariant is always true. Then the proof is obtained through induction based on ordering constraints enforced by the compiler and processor memory models, as well as the algorithm. Examples of formal mathematical proofs can be found in the literature,[12,17] and other authors have also presented interesting proofs on algorithm progress and correctness.[16,19,26]

Conclusion

When it comes to nonblocking algorithms and data structures, we are in an arms race. On one side, compiler writers and chip designers want to exploit all the freedom allowed by any unspecified behavior. On the other side, designers and developers of parallel algorithms want to have precisely defined behavior when it comes to dependency ordering.

Nonblocking synchronization can yield astonishing results in terms of scalability and real-time response, but it comes at the expense of verification state space. Verifying algorithms on multiple architectures, each with its own memory model, also increases state-space size. Once the weakest ordering constraints are modeled, however, new architectures that fit within those constraints can be added with no extra verification cost, other than testing to ensure the memory model is well defined.

There are various ways to increase confidence in a nonblocking algorithm, including oriented stress testing, formal and execution-based model checking, mathematical proofs, and code review. Given that the state space to explore increases quickly with the complexity of a nonblocking algorithm, designing small algorithms with full state-space verification in mind helps full state-space verification. Those can then be composed into more complex algorithms.

As programming languages improve their awareness of concurrency, it will be interesting to see the continuing advance in modeling and verification of concurrency at the compiler level and by static-code analyzers.

Acknowledgments

Thanks to Paul E. McKenney for his work on RCU and nonblocking synchronization, David Miller for his work on Linux kernel atomic operations, David Howells for his work on Linux kernel atomic operations and memory barriers, and Hugh Dickins for his work on type-safe memory. Thanks also to the Linux kernel community for their work and feedback on nonblocking synchronization, memory models, and lockdep, in particular Lai Jiangshan, Ingo Molnar, Peter Zijlstra, and Arjan van de Ven. I am indebted to Paul E. McKenney, Samy Bahra, Christian Babeux, Jeremie Galarneau, and Michel R. Dagenais for reviewing this article. C

Related articles on queue.acm.org

Revisiting Network I/O APIs: The netmap Framework
Luigi Rizzo
http://queue.acm.org/detail.cfm?id=2103536

Software and the Concurrency Revolution
Herb Sutter abd James Larus
http://queue.acm.org/detail.cfm?id=1095421

Real-World Concurrency
Bryan Cantrill and Jeff Bonwick
http://queue.acm.org/detail.cfm?id=1454462

References

1. Alglave, J., Maranget, L., Sarkar, S. and Sewell, P. Litmus: Running tests against hardware. In *Proceedings of the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, (2011); http://dl.acm.org/citation.cfm? id=1987389.1987395.
2. ARM. *ARM Architecture Reference Manual*, 2010; http://infocenter.arm.com/help/ index.jsp?topic=/com.arm.doc.set.architecture/index.html.
3. Baier, C. and Katoen, J. *Principles of Model Checking.* MIT Press, Cambridge, MA, 2008; http://books.google.ca/books?id=nDQiAQAAIAAJ.
4. Bergan, T., Anderson, O., Devietti, J., Luis, C. and Grossman, D. Coredet: A compiler and runtime system for deterministic multithreaded execution. *ACM SIGARCH Computer Architecture News 38*, 1 (2010), 53–64; http://doi.acm.org/10.1145/ 1735970.1736029.
5. Bernstein, P., Shipman, D. and Wong, W. Formal aspects of serializability in database concurrency control. *IEEE Transactions on Software Engineering 5*, 3 (1979), 203–216.
6. Burnim, J., Elmas, T., Necula, G. and Sen, K. CONCURRIT: testing concurrent programs with programmable state-space exploration. In *Proceedings of the 4th Usenix Conference on Hot Topics in Parallelism*, (2012), 16-16; http://dl.acm.org/citation.cfm?id=2342788.2342804.
7. Clarke, E., Grumberg, O. and Peled, D. 1*Model checking.* MIT Press, Cambridge, MA, 1999; http: //books.google.ca/books?id=Nmc4wEaLXFEC.
8. Click, C. Lock-free hash table. JavaOne Conference, 2007.
9. Corbet, J. The kernel lock validator. LWN, 2006; http://lwn.net/Articles/185666/.
10. Desnoyers, M. Low-impact operating system tracing. Ph.D. dissertation. Ecole Polytechnique de Montreal; http://www.lttng. org/pub/thesis/desnoyers-dissertation-2009-12.pdf.
11. Desnoyers, M., McKenney, P.E. and Dagenais, M.R. Forthcoming. Multicore systems modeling for formal verification of parallel algorithms. *Operating Systems Review.*
12. Desnoyers, M., McKenney, P.E., Stern, A.S., Dagenais, M.R. and Walpole, J. User-level implementations of read-copy-update. *IEEE Transactions on Parallel and Distributed Systems 23*, 2 (2012), 375–382.
13. Drusinsky, D. *Modeling and Verification Using UML Statecharts.* Elsevier Science, 2011; http://books.google.ca/books?id=JMz-SWTfgiAC.
14. Ferrante, J., Ottenstein, K.J. and Warren, J.D. The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems 9*, 3 (1987); 319-349; http://doi.acm.org/10.1145/24039.24041.
15. Gosling, J., Joy, B., Steele, G., Bracha, G. and Buckley, A. *The Java Language Specification, Java SE 7 Edition.* Pearson Education, 2013; http://books.google.ca/books? id=2RYN9exiTnYC.
16. Gotsman, A., Cook, B., Parkinson, M. and Vafeiadis, V. Proving that nonblocking algorithms don't block. In *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, (2009), 16–28; http: //doi.acm.org/10.1145/1480881.1480886.
17. Gotsman, A., Rinetzky, N. and Yang, H. Verifying concurrent memory reclamation algorithms with grace. In *European Symposium on Programming.* (Rome, Italy, 2013) Springer.
18. Haas, A., Kirsch, C. M., Lippautz, M. and Payer, H. How FIFO is your concurrent FIFO queue? In *Proceedings of the Workshop on Relaxing Synchronization for Multicore and Manycore Scalability*, (2012).
19. Herlihy, M. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems 13*, 1 (1991), 124–149; http://doi.acm.org/10.1145/ 114005.102808.
20. Herlihy, M.P. and Wing, J.M. Linearizability: a correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems 12*, 3 (1990), 463–492; http://doi.acm. org/10.1145/78969.78972.
21. Holzmann, G.J. The model checker Spin. *IEEE Transactions on Software Engineering 23*, 5 (1997): 279–295.
22. IBM. Power ISA Version 2.06 Revision B, 2010; http://www.power.org/ resources/reading/.
23. Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual: Instruction Set Reference, A-Z*; http://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-2a-2b-instruction-set- html.
24. International Organization for Standards. 2011. Programming languages - C++, ISO/IEC 14882:2011.
25. Kirsch, C.M., Lippautz, M. and Payer, H. Fast and scalable k-fifo queues. University of Salzburg, Salzburg, Austria. Technical Report 2012–04.
26. Michael, M.M. Hazard pointers: safe memory reclamation for lock-free objects. *IEEE Transactions on Parallel and Distributed Systems 15*, 6 (2004), 491–504; http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=1291819.
27. Michael, M.M. and Scott, M.L. Simple, fast, and practical nonblocking and blocking concurrent queue algorithms. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing* (1996), 267–275; http: //doi.acm.org/10.1145/248052.248106.
28. MIPS Technologies Inc. *MIPS Architecture for Programmers, Volume II-A: The MIPS64 Instruction Set*, 2012
29. Olszewski, M., Ansel, J. and Amarasinghe, S. Kendo: efficient deterministic multithreading in software. *ACM SIGPLAN Notices 44*, 3 (2009), 97–108; http://dl.acm.org/citation.cfm?id=1508256.
30. Sarkar, S., Sewell, P., Alglave, J., Maranget, L. and Williams, D. Understanding power multiprocessors. *ACM SIGPLAN Notices 46*, 6 (2011), 175–186; http://doi.acm.org/10.1145/1993316. 1993520.

Mathieu Desnoyers is president and founder of EfficiOS. He maintains the LTTng project and the Userspace RCU library. His research interests are in performance analysis tools, operating systems, scalability, and real-time concerns.

DOI:10.1145/2483852.2483869

The RSVP voice-recognition search engine improves speech recognition and translation accuracy in question answering.

BY YANG TANG, DI WANG, JING BAI, XIAOYAN ZHU, AND MING LI

Information Distance Between What I Said and What It Heard

VOICE INPUT IS a major requirement for practical question answering (QA) systems designed for smartphones. Speech-recognition technologies are not fully practical, however, due to fundamental problems (such as a noisy environment, speaker diversity, and errors in speech). Here, we define the information distance between a speech-recognition result and a meaningful query from which we can reconstruct the intended query, implementing this framework in our RSVP system.

In 12 test cases covering male, female, child, adult, native, and non-native English speakers, each with 57 to 300 questions from an independent test set of 300 questions, RSVP on average reduced the number of errors by 16% for native speakers and by 30% for non-native speakers over the best-known speech-recognition software. The idea was then extended to translation in the QA domain.

In our project, which is supported by Canada's International Development Research Centre (http://www.idrc.ca/), we built a voice-enabled cross-language QA search engine for cellphone users in the developing world. Using voice input, a QA system would be a convenient tool for people who do not write, for people with impaired vision, and for children who might wish their Talking Tom or R2-D2 really could talk.

The quality of today's speech-recognition technologies, exemplified by systems from Google, Microsoft, and Nuance does not fully meet such needs for several reasons:

- Noisy environments in common audio situations;[1]
- Speech variations, as in, say, adults vs. children, native speakers vs. non-native speakers, and female vs. male, especially when individual voice-input training is not possible, as in our case; and
- Incorrect and incomplete sentences; even customized speech-recognition systems would fail due to coughing, breaks, corrections, and the inability to distinguish between, say, "sailfish" and "sale fish."

Speech-recognition systems can be trained for a "fixed command set" of up to 10,000 items, a paradigm that

» key insights

- **Focusing on an infinite but highly structured domain (such as QA), we significantly improve general-purpose speech recognition results and general-purpose translation results.**
- **Assembling a large amount of Internet data is key to helping us achieve these goals; in the highly structured QA domain, we collected millions of human-asked questions covering 99% of question types.**
- **RSVP development is guided by a theory involving information distance.**

does not work for general speech recognition. We consider a new paradigm: speech recognition limited to the QA domain covering an unlimited number of questions; hence it cannot be trained as a fixed command set domain. However, our QA domain is highly structured and reflects clear patterns.

We use information on the Internet in the QA domain to find all possible question patterns, then use it to correct the queries that are partially recognized by speech-recognition software. We collected more than 35 million questions from the Internet, aiming to use them to infer templates or patterns to reconstruct the original intended question from the speech-recognition input. Despite this, we must still address several questions:

▸ How do we know if an input question from the speech-recognition system is indeed the original user's question?;

▸ How do we know if a question in our database is the user's intended question?;

▸ Should we trust the input or the database?; and

▸ Often, neither the database nor the input is always exactly right, so can we reconstruct the original question?

We provide a mathematical framework to address these questions and implement the related RSVP system. Our experiments show RSVP significantly improves current speech-recognition systems in the QA domain.

Related Work

Speech recognition[1] has made significant progress over the past 30 years since the introduction of statistical methods and hidden Markov models. Many effective algorithms have been developed, including the EM algorithm, the Baum-Welch algorithm, Viterbi N-best search, and N-gram language models trained on large data corpora. However, as explored by Baker et al.,[1] automatic speech recognition is still an unsolved problem.

Unlike traditional speech-recognition research, we propose a different paradigm in the QA domain. In it, we are able to collect a very large pure text corpus (no voice) that differs from the fixed command set domain where it is possible to train up to, say, 10,000 commands. The QA domain is unbounded, and the number of existing questions on QA websites involves more than 100 million questions, yet with very low coverage of all possible questions. These texts can be "clustered" into patterns. Here, we demonstrate that these patterns have 99% coverage of all possible question types, suggesting we can use them to improve speech-recognition software in this domain.

Previous research suggested that context information,[21] the knowledge-base,[16,20] and conceptual relationships[15] all can help address this.

Information Distance

Here, we develop the mathematical framework on which we designed our system. To define Kolmogorov complexity (invented in the 1960s), we start by fixing a universal Turing machine U. The Kolmogorov complexity of a binary string x, given another binary string y, $K_U(x|y)$, is the length of the shortest (prefix-free) program for U that outputs x with input y. Since it can be shown that for a different universal Turing machine U', the metric differs by only a constant, we write $K(x|y)$ instead of $K_U(x|y)$. We write $K(x|\varepsilon)$, where ε is the empty string, as $K(x)$. We call a string x random if $K(x) \geq |x|$. See Li and Vitányi[14] for more on Kolmogorov complexity and its rich applications.

Note $K(x)$ defines the amount of information in x. What would be a good departure point for defining an "information distance" between two objects? In the 1990s, Bennett et al.[2] studied the energy cost of conversion between two strings, x and y. John von Neumann hypothesized that performing 1 bit of information processing costs 1KT of energy, where K is Boltzmann's constant and T is the room temperature. In the 1960s, observing that reversible computations could be done at no cost, Rolf Landauer revised von Neumann's proposal to hold only for irreversible computations. Starting from this von Neumann-Landauer principle, Bennett et al.[2] proposed using the minimum number of bits needed to convert x to y and vice versa to define their distance. Formally, with respect to a universal Turing machine U, the cost of conversion between x and y is defined as

$$E(x,y) = \min\{|p| : U(x,p) = y, U(y,p) = x\} \quad (1)$$

It is clear that $E(x,y) \leq K(x|y) + K(y|x)$. Bennett et al.[2] obtained the following optimal result, modulo $\log(|x| + |y|)$:

Theorem 1. $E(x,y) = \max\{K(x|y), K(y|x)\}$.

Thus, we define information distance between two sequences, x and y, as $D_{\max}(x,y) = \max\{K(x|y),K(y|x)\}$.

This distance $D_{\max}$ was shown to satisfy the basic distance requirements (such as positivity, symmetricity, and triangle inequality). Furthermore, $D_{\max}$ is "universal" in the sense that it always minorizes any other reasonable computable distance metric.

We use information on the Internet in the QA domain to find all possible question patterns, then use it to correct the queries that are partially recognized by speech-recognition software.

This concept, and its normalized versions, were applied to whole-genome phylogeny,[12] chain-letter-evolution history,[3] plagiarism detection,[4] other phylogeny studies,[13] music classification,[5] and parameter-free data mining,[10] and has been followed by many other applications (for topics mentioned here that do not have references, see Li and Vitányi[14]) including protein-structure comparison, heart-rhythm data analysis, QA systems, clustering, multiword expression linguistic analysis, software evolution and engineering, software metrics and obfuscation, webpage authorship, topic, and domain identification, phylogenetic reconstruction, SVM kernel for string classification, ortholog detection,[19] analyzing worms and network traffic, image similarity, Internet knowledge discovery,[6] multidocument summarization, network structure, and dynamic behavior,[17] and gene expression dynamics in macrophase.[18]

Despite its useful properties and applications, the max distance $D_{\max}(x,y)$ involves several problems when only partial matches are considered[8,22] where the triangle inequality fails to hold and irrelevant information must be removed. Thus, Li et al.[11] introduced a complementary information-distance metric to resolve these problems. In Equation 1 we determine the smallest number of bits that must be used to reversibly convert between x and y. To remove the irrelevant information from x or y, we thus define, with respect to a universal Turing machine U, the cost of conversion between x and y as

$$E_{\min}(x,y) = \min\{|p| : U(x,p,r) = y, U(y,p,q) = x, |p|+|q|+|r| \leq E(x,y)\}, \quad (2)$$

This definition separates r from x and q from y. Modulo an $O(\log(|x| + |y|))$ additive term, the following theorem was proved in Li[11]:

Theorem 2. $D_{\min}(x,y) = \min\{K(x|y), K(y|x)\}$.

We can thus define $D_{\min}(x,y) = E_{\min}(x,y)$ as a complementary information-distance metric that disregards irrelevant information. $D_{\min}$ is obviously symmetric but does not satisfy triangle inequality. Note that $D_{\min}$ was used in the QUANTA QA system to deal with concepts that are more popular than others.[23,24]

Min-Max Distance

Now we formulate our problem in the frame of information distance, given a question database Q and k input questions from a speech-recognition system, as in $I = \{q_1,\ldots,q_k\}$ ($k \leq 3$ for the Google speech-recognition server in our experiments and $k = 1$ in our translation application. The goal is to compute the user's intended question q. It could be one of the q_is; it could be a combination of all k of them; and it could also be one of the questions in Q that is close to some parts of the q_is.

We wish to find the most plausible question q, such that q fits one of the question patterns in Q, and q is "close" to I. We assume Q contains almost all question patterns; later, we provide an experimental justification for this claim.

We can thus formulate our problem as: Given Q and I, find q such that it minimizes the sum of "distances" from Q to q and q to I, as in

$$I \longleftrightarrow q \longleftrightarrow Q$$

Here, Q is a huge database of 35 million questions asked by users. We assume q is "similar" to one of them; for example, a QA user might ask "Who is the mayor of Waterloo, Ontario?," but Q might include such questions as "Who is the mayor of Toronto, Ontario?" and "Who is the mayor of Washington, D.C.?" I sometimes contains questions like "Hole is the mayor?" and "Who mayor off Waterloo" from the speech-recognition software. Since Q is so large, the $D_{\max}$ measure does not make sense here, as most of the information in Q is irrelevant. It is natural to use $D_{\min}(q,Q)$ here. For the distance between q and I, we use $d_{\max}(q, I)$ to measure it. Given I, Q, we wish to find q that minimizes the function

$$\delta D_{\min}(q,Q) + D_{\max}(I,q), \qquad (3)$$

where $D_{\min}$ measures information distance between q and Q with irrelevant information removed; $D_{\max}$ is the information distance between q and I. We know

$$D_{\min}(x,y) = \min\{K(x|y), K(y|x)\},$$
$$D_{\max}(x,y) = \max\{K(x|y), K(y|x)\}.$$

Thus, $D_{\min}(q,Q) = K(q|Q)$, because Q is very large and q is a single question. Note that d is a coefficient that determines how much weight we wish to give to a correct template or pattern in Q.

Equation 3 thus becomes

$$\delta K(q|Q) + \max\{K(q|I),K(I,q)\}. \qquad (4)$$

Obervations: We need $\delta > 1$, so $q = I$ does not minimize Equation 4. If δ is too large, then $q = \varepsilon$ might minimize Equation 4. There is a trade-off: Sometimes a less-popular pattern (taking more bits in the $D_{\min}$ term) might fit I better (taking fewer bits in the $D_{\max}$ item), and a more popular pattern (taking fewer bits in the $D_{\min}$ item) might miss one or two key words in I, taking more bits to encode in the $D_{\max}$ item. Note that δ is optimized for the trade-off.

Encoding Issues

To minimize Equation 4, we solve three problems:

- Encode q using Q in the first term. It is a problem to encode an item with respect to a big set;
- Encode q using I or encode I using q, and take whichever is larger in the second term; and
- Find all possible candidates q and a q_0 that minimizes Equation 4.

We see that Q is very large and contains different "types" of questions. For each such type, we extract one or more question templates. In this way, Q can be viewed as a set of templates, with each template, denoted as p, covering a subset of questions from Q. When encoding q, we need not encode q from Q directly. Instead, we encode q with respect to the patterns or templates of Q; for example, if a pattern p in Q appears N times in Q, then we use $\log_2(|Q|{=}N)$ bits to encode the index for this pattern. Given pattern p, we encode q with p by encoding their word mismatches. There is a trade-off between the encoding of p and the encoding of q, given p. A common pattern may be encoded with a few bits but also may require more bits to encode a specific question using the pattern; for example, the template "who is the mayor of City Name" requires more bits to encode than the template "who is the mayor of Noun" because the former is a smaller class than the latter. However, the first template requires fewer bits to generate the question "who is the mayor of Waterloo" since it requires fewer bits to encode Waterloo from the class "City Name" than from the class "Noun."

The patterns could be extracted by pre-processing or dynamically according to the input. In practice, we extract patterns only from questions relevant to I, denoted as Q'. We organize Q' hierarchically. Similar questions are mapped to a cluster, and similar clusters are mapped to a bigger cluster. We extract one pattern from each cluster using a multiple alignment algorithm. This pattern should be as specific as possible while at the same time cover all questions in the cluster. Note that the higher the cluster in the hierarchical structure, the more general the pattern. Our hierarchical clustering algorithm thus assures we can extract all possible patterns from relevant questions. We make use of numerous semantic and syntactic information sources during the process, including POS tagger, Name Entity Recognition, WordNet, and Wikipedia. For example, given a cluster of three questions:

- Who is the mayor of Toronto?;
- Who is the president of the United States?; and
- Who is a senator from New York?

We could extract one pattern (such as Who is the Leader of Location? "Mayor," "president," and "senator") are all mapped to the Leader class, while "Toronto," "United States," and "New York" all belong to the Location class.

If we treat pattern p as a sentence, the problem of item-to-set encoding becomes item-to-item encoding, as in the computation of $K(q|I)$ and $K(I|q)$. To convert a sentence from another sentence, we need encode only the word mismatches and the missing words. The best alignment between two sentences is found through a standard dynamic programming algorithm. We encode a missing word by the negative logarithm of their probabilities to appear at the given locations and encode the mismatches by calculating their semantic and morphology similarities. It requires fewer bits to encode between synonyms than antonyms.

Equation 4 must select the candidate question q, for which we use two strategies:

Offline. We cluster questions in Q and generate patterns offline, finding the most likely pattern, then generate q that is close to the input and to one of the patterns; and

Online. We consider only the question candidates relevant to the input and that could be matched by at least one of our templates generated from a few questions and that share some keywords with the input in Q.

Finally, we choose the best q that minimizes Equation 4. Furthermore, we apply a bigram language model to filter questions with low trustworthiness. The language model is trained in our background question set Q. The value δ is trained as a part of our experiments. In our system the δ value is a function of the lengths of the input questions.

We have thus implemented RSVP, which, given speech recognition input $\{q_1, q_2, q_3\}$, finds q such that Equation 4 is minimized.

Implementation Details

The following are some RSVP implementation details:

Step 1. Analyze input:

▸ Split the questions into words by Stanford Pos Tagger; at the same time, name entities are extracted from the input questions using Stanford NER, Linepipe NER, and YAGO; and

▸ Find the best alignments among the input questions through dynamic programming. Words or name entities with similar pronunciations are mapped together; for example, given three questions—whole is the mayor of Waterloo, hole is the mayor of Water, and whole was the mayor of Water—the best word alignment would look like this:

```
Whole is the mayor of Waterloo
Hole is the mayor of Water
Whole was the mayor of Water
```

Step 2. Improve input questions:

▸ Build a question based on the word-alignment results from Step 1; for each aligned word block, we choose one word to appear in the result;

▸ We assume that a good format question should contain a "wh"-word, including what, who, which, whose, whom, when, where, why, how, or what, or an auxiliary verb, including be, have, do, shall, will (would), shall (should), may (might), must, need, dare, or ought. If the inputs do not contain any such word, we add proper words into the question candidates; and

▸ Since some correct words may not appear in the input, we further expand the question candidates with homonym dictionaries and metaphone dictionaries.

Step 3. Analyze relevant database patterns:

▸ Find relevant database questions, sorting them based on their semantic and syntactic similarity to the improved input questions from Step 2;

▸ If a question is almost the same as one of the input questions, we return that input question directly, and no further steps are done in this case;

▸ The database questions involve many forms and patterns. We group similar questions together through a hierarchical clustering algorithm. The distance between two clusters is calculated based on the syntactic similarity between questions. The algorithm stops when the minimum distance between clusters reach a predefined threshold;

▸ When the relevant questions are grouped into clusters, we are able to extract patterns from each cluster. Following the algorithm outlined in Step 1, we align questions in each group, using their semantic similarities to encode the word distance. Then a group of questions is converted into a single list with multiple word blocks, with each block containing several alternative words from different questions; for example, given questions "Who is the mayor of New York," "Who is the president of United States," and "Which person is the leader of Toronto," we obtain a list of word blocks after alignment:

{who, who, which person}, {is}, {the}, { mayor, leader, president } of { New York, United States, Toronto}; and

▸ For each aligned word block, we further extract tags that would best describe the slot; here, YAGO[9] is used to describe the meaning of each word or phrase. We extract several most-common facts as the description of each word block. We then obtain one or several semantic patterns composed of words and facts from YAGO.

Step 4. Generate the candidate questions:

▸ Map the original input questions into the patterns we extracted from the database and replace the words in the patterns with the words from the input. Many candidate questions could be generated by considering the various combinations of word replacements; and

▸ To reduce complexity, we train a bigram language model from our question set, removing candidate questions with low probability.

Step 5. Rank candidate questions using information distance:

▸ Calculate the distance between the candidate questions and the input questions $K(q|I)$ and $K(I|q)$. We align the candidate and input questions and encode the word mismatches and missing words, encoding a missing word through minus logarithm of their probability to appear at the said locations and calculating word mismatches through their semantic, morphology, and metaphone similarities;

- Calculate the distance between the candidate questions and the patterns $K(q|p)$. A method similar to the previous step is used to calculate the distances between questions and patterns; and
- RSVP Ranks all the candidates using Equation 4.

Step 6. Return the candidate with minimum information distance score as the final result: In order to improve speed, the last three items of step 3 may be performed offline on the complete database Q.

Completeness of the Database *Q*

We tested the hypothesis that Q contains almost all common question types. The test set T contained 300 questions, selected (with the criteria of no more than 11 words or 65 letters, one question in a sentence, and no non-English letters) from an independent Microsoft QA set at http://research.microsoft.com/en-us/downloads/88c0021c-328a-4148-a158-a42d7331c6cf. We found that all but three have corresponding patterns in Q. Only three questions lacked strictly similar patterns in Q: Why is some sand white, some brown, and some black? Do flying squirrels fly or do they just glide? And was there ever a movement to abolish the electoral college? We will provide the data set T upon request.

Experiments

Our experiments aimed to test RSVP's ability to correct speech-recognition errors in the QA domain, focusing on non-native speakers, as there are three non-native English speakers for each native English speaker in the world. Here, we further test and justify our proposed methodology by extending it to translation in the QA domain.

Experiment setup. We initially (in 2011) used the Nuance speech-recognition server and later switched to Google speech recognition (http://google.com), because the Google server has no daily quota and responds quicker. The RSVP system is implemented in a client-server architecture. The experiments were performed at a computer terminal with a microphone. The experimenter would read a question, and Google speech recognition would return three options. RSVP uses the three questions as input and computes the most likely question.

Dataset. We use the set T described in the previous section. T contains 300 questions. T was chosen independently, and $T \cap Q = \Phi$. Not all questions in T were used by each speaker in the experiments; non-native speakers and children skipped sentences that contain difficult-to-pronounce words, and less-proficient English speakers tend to skip more questions.

Time complexity. On a server with four cores, 2.8GHz per core, and 4G memory, RSVP typically uses approximately 500ms to correct one question; that is, the speaker reads a question into a microphone, Google voice recognition returns three questions, and RSVP uses the questions as input, taking approximately half a second to output one final question.

Human speaker volunteers. Such experiments are complex and time consuming. We tried our best to remove the individual speaker variance by having different people perform the experiments independently. We recruited 14 human volunteers, including native and non-native English speakers, adults and children, females and males (see Table 1) during the period 2011 to 2012.

We performed 12 sets of experiments involving 14 different speakers, all using the same test set T or a subset of T. Due to children's naturally short attention spans, the three native English-speaking children (two males, one female) completed one set of experiment (experiment 7), each responsible for 100 questions. A non-native-speaking female child, age 12, performed the test (experiment 10) independently but was able to finish only 57 questions.

In the following paragraphs, "CC" signifies that the speech-recognition software (from Google) returned the correct answer as the first option and RSVP agrees with it; "WC" signifies that the speech-recognition software returned the wrong answer as the first option and RSVP returned the correct answer; "CW" signifies that the speech-recognition software returned the correct answer as the first option and RSVP returned the wrong answer; and "WW" signifies that the speech-recognition software returned the wrong answer as the first option and RSVP also returned the wrong answer. All experiments were performed in quiet environments; in each, the speaker tried again if neither the speech recognition nor RSVP was correct (see Table 2).

Table 1. Individuals used in our experiments.

	Native Speaker		Non-Native Speaker	
	Adult	Child	Adult	Child
Female	0	1	4	1
Male	3	2	3	0

Table 2. Experimental results for speech correction.

Experiment	# questions	CC	WC	CW	WW	Base Translator Accuracy	RSVP Accuracy
Google as base translator	428	112	211	6	99	27.5%	75.6%
Microsoft as base translator	428	116	207	11	94	29.6%	75.6%
Google as base translator	52	21	15	0	16	40%	69%
Google as base translator	114	44	49	1	20	38%	81.6%

Table 3. Experimental results for speech correction

Experiment	Total No. of questions	CC	WC	CW	WW
1	164	105	39	5	15
2	300	219	25	6	50
3	300	222	15	5	58
4	257	141	41	7	68
5	181	100	26	4	51
6	214	125	29	10	50
7	206	145	19	8	34
8	298	180	12	4	102
9	131	77	14	0	40
10	57	28	4	1	24
11	63	35	9	1	18
12	107	62	9	2	34

Translation examples.

```
从多伦多到北京坐飞机多长时间
  Google translation: Fly from Toronto to Beijing long?
  Our translation: How long does it take to fly from Toronto to Beijing?
人身上有多少块骨头?
  Google translation: People who have a few bones?
  Our translation: How many bones do people have?
台湾的人口多少?
  Google translation: Taiwan’ s population size?
  Our translation: What is the population size of Taiwan?
恐龙上什么时候灭绝的?
  Google translation: When did the dinosaurs extinct?
  Our translation: When did the dinosaurs become extinct?
```

Experiment 1. Non-native speaker, male. Here, the speaker chose only easy-to-read questions from the 300-question Microsoft test set; the following are examples where RSVP corrected Google's errors:

Original question: How many types of birds are found in North America? Google speech-recognition output: How many pints of birds are formed in North America? How many pints of birds are found in North America? How many pints of birds performed in North America? RSVP output: How many types of birds are found in North America?

Original question: How does the atom bomb work? Google speech-recognition output: call does the atom bomb work? All does the atom bomb work? aha does the atom bomb work? RSVP output: How does the atom bomb work?

Original question: How holidays are celebrated around the world? Google speech-recognition output: call holidays are celebrated around in the wall? call holidays are celebrated around in the world? how all holidays are celebrated around in the wall? RSVP output: how holidays are celebrated around in the world?

Original question: Are there any known aliens? Google speech-recognition output: are there any loans deviance? are there any loans aliens? are there any known deviance? RSVP output: Are there any known aliens?

Original question: What does a frog eat? Google speech-recognition output: what is front seat? what is the frogs eat? what does the front seat? RSVP output: What does frogs eat?

Experiment 2. Native speaker, male;

Experiment 3. Native speaker, male;

Experiment 4. Non-native speaker, male;

Experiment 5. Non-native speaker, female;

Experiment 6. Non-native speaker, female.

Experiment 7. Three native English-speaking children, 100 questions each. They are eight years old, female; nine years old, male; and 11 years old, male. In principle, we prefer independent tests with one individual responsible for the complete set of 300 questions. However, we were only able to get each of the children to do 100 questions, skipping the difficult ones. The result is similar to that of adult native English speakers;

Experiment 8. Native English speaker, male;

Experiment 9. Non-native English speaker, male;

Experiment 10. Non-native English speaker, female, 11 years old, in Canada to attend summer camp to learn English; her English was rudimentary and consequently was able to read only 57 questions out of 300;

Experiment 11. Non-native English speaker, female; and

Experiment 12. Non-native English speaker, female.

In our experiments, Table 2, the non-native speakers and the children selected relatively easy-to-read questions (without, say, difficult-to-pronounce names) from *T* to do the tests. The ratio of improvements was better for the non-native speakers, reducing the number of errors (WW column) by 30% on average for experiments 1, 4, 5, 6, 9, 10, 11, and 12. For native speakers, RSVP also delivered a clear advantage, reducing the number of errors (WW column) by 16% on average for experiments 2, 3, 7, and 8. Such an advantage would be amplified in a noisy real-life environment. Allowing the speaker to repeat the question would increase the success rate, as in the following example (with Google): RSVP generated "How many toes does Mary Monroe have?" for the first query and "How many titles does Marilyn Monroe have?" for the second query. Combining the two questions, RSVP generated the correct intended question "How many toes does Marilyn Monroe have?"

Translation

To further justify the methodology proposed here, we extend the approach to translation in the QA domain for cross-language search. Here, we use Chinese-English cross-language search as an example, though the methodology works for other languages, too.

A Chinese-speaking person can perform a cross-language search of the English Internet in two ways:

Translate it all. Translate the whole English Internet, including all QA pairs in the English QA community and English (Wikipidea) databases, into Chinese; or

Translate a question. Translate a Chinese question into English, finding the answer in English, then translate the answer back to Chinese.

General-purpose translators today perform so poorly that the first option is out of the question. The RSVP methodology enables the second option, which involves two translations: the Chinese question to English, then the English answer back to Chinese. Since the QA answers are usually simple and read by humans, and the database relations can be manually translated, a general-purpose translator is sufficient and sometimes not even needed. The key to this approach is translating Chinese questions into English. We implemented the translation system and cross-language (Chinese-English) search as part of the RSVP QA system through the following steps:

- Translate a Chinese question into English through a general-purpose translator;
- Apply the (modified) correction procedure described here;
- Perform English QA search; and
- Translate the result back into Chinese through a general-purpose translator.

Table 3 outlines experiments with our translation system, using the notation outlined earlier: CC, WC, CW, and WW. The first three used data collected as we developed RSVP; the fourth used an independent dataset of 114 questions on a range of topics (see the figure here).

Conclusion

This work would be more effective if it were integrated into speech-recognition software so more voice information could be used. However, it targets dynamic special domains that are so numerous that training them separately would be prohibitive.

In addition to special-domain translation, the RSVP methodology can be used to correct the grammatical errors and spelling mistakes in a normal QA text search, as well as to create an automatic writing assistant for a highly structured domain.

Moreover, "tuning" improves all systems; for example, if we ask "What is the distance between Toronto and Waterloo bla," then we know the extra "bla" should be removed, as inferred from system knowledge, requiring zero bits to encode such a deletion. The theory allows us to add "inferrable rules" at no additional cost.

Acknowledgments

We thank Li Deng of Microsoft for his advice on speech-recognition software and Leonid Levin of Boston University for discussions and suggestions on alternative formulations. We are grateful to the volunteers who participated in our experiments. We thank the referees and Nicole Keshav for helping us improve the article. We also thank Alan Baklor of Answer.com for his help. This work has been partially supported by Canada's IDRC Research Chair in Information Technology program, NSERC Discovery Grant OGP0046506, the Canada Research Chair program, a CFI Infrastructure grant, an NSERC Collaborative grant, Ontario's Premier's Discovery Award, and the Killam Prize. C

References

1. Baker, J.M., Deng, L., Glass, J., Khudanpur, S., Lee, C.H., Morgan, N., and O'Shaughnessy, D. Developments and directions in speech recognition and understanding, Part 1. *IEEE Signal Processing Magazine 26*, 3 (May 2009), 75–80.
2. Bennett, C.H., Gács, P., Li, M., Vitányi, P., and Zurek, W. Information distance. *IEEE Transactions on Information Theory 44*, 4 (July, 1998), 1407–1423.
3. Bennett, C.H., Li, M., and Ma, B. Chain letters and evolutionary histories. *Scientific American 288*, 6 (June 2003), 76–81.
4. Chen, X., Francia, B., Li, M., McKinnon, B., and Seker, A. Shared information and program plagiarism detection. *IEEE Transactions on Information Theory 50*, 7 (July 2004), 1545–1550.
5. Cilibrasi, R., Vitányi, P., and de Wolf, R. Algorithmic clustering of music based on string compression. *Computer Music Journal 28*, 4 (Winter 2004), 49–67.
6. Cilibrasi, R. and Vitányi, P. The Google similarity distance. *IEEE Transactions on Knowledge and Data Engineering 19*, 3 (Mar. 2007), 370–383.
7. Cuturi, M. and Vert, J.P. The context-tree kernel for strings. *Neural Networks 18*, 4 (Oct. 2005), 1111–1123.
8. Fagin, R. and Stockmeyer, L. Relaxing the triangle inequality in pattern matching. *International Journal of Computer Vision 28*, 3 (1998), 219–231.
9. Hoffart, J., Suchanek, F.M., Berberich, K., and Weikum, G. YAGO2: A spatially and temporally enhanced knowledgebase from Wikipedia. *Artificial Intelligence 194* (Jan. 2013), 28–61.
10. Keogh, E., Lonardi, S., and Ratanamahatana, C.A. Towards parameter-free data mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, New York, 2004, 206–215.
11. Li, M. Information distance and its applications. *International Journal on the Foundations of Computer Science 18*, 4 (Aug. 2007), 669–681.
12. Li, M., Badger, J., Chen, X., Kwong, S., Kearney, P., and Zhang, H. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics 17*, 2 (Feb. 2001), 149–154.
13. Li, M., Chen, X., Li, X., Ma, B., and Vitányi, P. The similarity metric. *IEEE Transactions on Information Theory 50*, 12 (Dec. 2004), 3250–3264.
14. Li, M. and Vitányi, P. *An Introduction to Kolmogorov Complexity and Its Applications, Third Edition.* Springer-Verlag, New York, 2008.
15. Lieberman, H., Faaborg, A., Daher, W., and Espinosa, J. How to wreck a nice beach you sing calm incense. In *Proceedings of the 10th International Conference on Intelligent User Interfaces* (2005), 278–280.
16. Lopes, L.R. A software agent for detecting and correcting speech recognition errors using a knowledge base. SATNAC, 2008.
17. Nykter, M., Price, N.D., Larjo, A., Aho, T., Kauffman, S.A., Yli-Harja, O., and Shmulevich, I. Critical networks exhibit maximal information diversity in structure-dynamics relationships. *Physical Review Letters 100*, 5 (Feb. 2008), 058702-706.
18. Nykter, M., Price, N.D., Aldana, M., Ramsey, S.A., Kauffman, S.A., Hood, L.E., Yli-Harja, O., and Shmulevich, I. Gene expression dynamics in the macrophage exhibit criticality. *Proceedings of the National Academy of Sciences 105*, 6 (Feb. 2008), 1897–1900.
19. Pao, H.K. and Case, J. Computing entropy for ortholog detection. In *Proceedings of the International Conference on Computational Intelligence* (Istanbul, Turkey, Dec. 17–19, 2004).
20. Rosenfeld, R. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE 88*, 8 (Aug. 2000), 1270–1278.
21. Sarma, A. and Palmer, D.D. Context-based speech recognition error detection and correction. In *Proceedings of the Human Language Technology Conference* (Boston, May 2–7). Association of Computational Linguistics, Stroudsburg, PA, 2004, 85–88.
22. Veltkamp, R.C. Shape matching: Similarity measures and algorithms. In *Proceedings of the International Conference on Shape Modeling Applications* (Genoa, Italy, 2001), 188–197.
23. Zhang, X., Hao, Y., Zhu, X.Y., and Li, M. New information measure and its application in question-answering system. *Journal of Computer Science and Technology 23*, 4 (July 2008), 557–572.
24. Zhang, X., Hao, Y., Zhu, X., and Li, M. Information distance from a question to an answer. In *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery in Data Mining* (San Jose, CA, Aug. 12–15). ACM Press, New York, 2007, 874–883.

Yang Tang (tangyang9@gmail.com) is a research associate in the David R. Cheriton School of Computer Science at the University of Waterloo, Waterloo, Ontario.

Di Wang (kingwangdi@gmail.com) is a graduate student in the David R. Cheriton School of Computer Science at the University of Waterloo, Waterloo, Ontario.

Jing Bai (jbai@microsoft.com) is a researcher in Microsoft Corporation, Silicon Valley campus, Sunnyvale, CA.

Xiaoyan Zhu (zxy-dcs@tsinghua.edu.cn) is a professor in the Tsinghua National Laboratory for Information Science and Technology and Department of Computer Science and Technology and director of State Key Laboratory of Intelligent Technology and Systems at Tsinghua University, Beijing.

Ming Li (mli@uwaterloo.ca) is Canada Research Chair in Bioinformatics and a University Professor in the David R. Cheriton School of Computer Science at the University of Waterloo, Waterloo, Ontario.

DOI:10.1145/2483852.2483870

How to fairly allocate divisible resources, and why computer scientists should take notice.

BY ARIEL D. PROCACCIA

Cake Cutting: Not Just Child's Play

ADDRESSING SOME OF the great challenges of the 21st century involves a thorough understanding of fairness considerations. Fairly dividing limited natural resources (such as fossil fuels, clean water, and the environment's capacity to absorb greenhouse gases) is of utmost importance; for example, fairness plays a key role in the evolution of the United Nations Framework Convention on Climate Change, a global treaty for combating climate change, and its implementation; indeed, its language demands fairness. It seems intuitive that countries with developing economies would bear less of the burden, and developed countries, which share historical responsibility for pollution, bear more. However, only a concerted effort involving developing countries would yield a meaningful result. Moreover, some countries (such as small islands) are more susceptible to the adverse effects of climate change and call for global solidarity. With so many competing needs and demands, what would be a fair solution to dividing up the costs of pollution?

Similar (though simpler) questions about fairness have tantalized thinkers for millennia and can be traced back to the Bible and to ancient Greece. Indeed, the Bible famously recounts the tale of King Solomon's role in a maternity dispute. In response to two women both claiming to be the true mother of the same baby, Solomon suggested his guards cut the baby in two and give each woman half. When one of the women protested, begging Solomon to give the baby to the other woman, he declared she must be the true mother (1 Kings 3:16–27).

The 20th century experienced a shift toward mathematically rigorous approaches to fairness. In mathematics, the formal notion of "envy-freeness," with participants preferring to keep their own allocation to swapping with other participants, appeared in *Puzzle-Math*, a 1958 book of mathematical puzzles by Gamow and Stern.[23] In the economics literature, Foley[21] initiated the formal study of envy-freeness. The notion of proportionality, with each of n participants receiving at least $1/n$ of each participant's own value for getting everything, was considered by Steinhaus[39] as early as the 1940s.

In economics today, fair division is considered a significant subfield of microeconomic theory. Unlike Solomon, the literature usually distinguishes between allocation of divisible goods such as land (see Figure 1), time, and memory on a computer, and indivisible goods (such as a house or the computer itself, or a baby). Each variation has attracted ample attention, as in the book by Moulin.[31] However, com-

» key insights

- **A cake-cutting algorithm divides a heterogeneous divisible good in a way that achieves formal fairness guarantees.**
- **Designing cake-cutting algorithms that are computationally efficient and immune to manipulation is a challenge for computer scientists.**
- **Insight from the study of cake cutting can be applied to the allocation of computational resources.**

puter scientists have focused mainly on the (not necessarily fair) allocation of indivisible goods; see, for example, Conitzer[15] and Roughgarden.[36] Nevertheless, I hope to convince the reader that the study of the fair allocation of divisible goods, particularly cake cutting, gives rise to natural challenges and exciting opportunities for computer scientists, theoreticians and practitioners alike.

Although recent work on cake cutting spans multiple disciplines, my discussion here is geared toward computer science, as well as a general readership. For a more rigorous introduction to fair-division schemes from a mathematical-economics point of view, see the survey by Thomson.[42]

Cake-Cutting Classics

To explore the abstract setting, imagine dividing a birthday cake among several children. The cake has different toppings, and the children have different tastes, one desiring, say, toasted nuts, and another craving chocolate curls. Cake cutting is indeed a useful metaphor for the more formal-sounding task of allocating a heterogeneous divisible good among multiple players with different preferences. The study of fair cake-cutting algorithms originated with Steinhaus, Knaster, and Banach in Poland during World War II,[39] and over the years has attracted mathematicians, economists, and political scientists.

Research on cake cutting includes two complementary strands: One establishes the existence of cake divisions with desirable properties (such as envy-freeness[40]), typically via proofs that are non-constructive. The other aims to help players achieve fair allocations by designing cake-cutting algorithms; see the books by Brams and Taylor[11] and Robertson and Webb.[35] I focus on the second, but keep in mind that many deep existence results allow us to assume the cake divisions we are trying to compute do in fact exist.

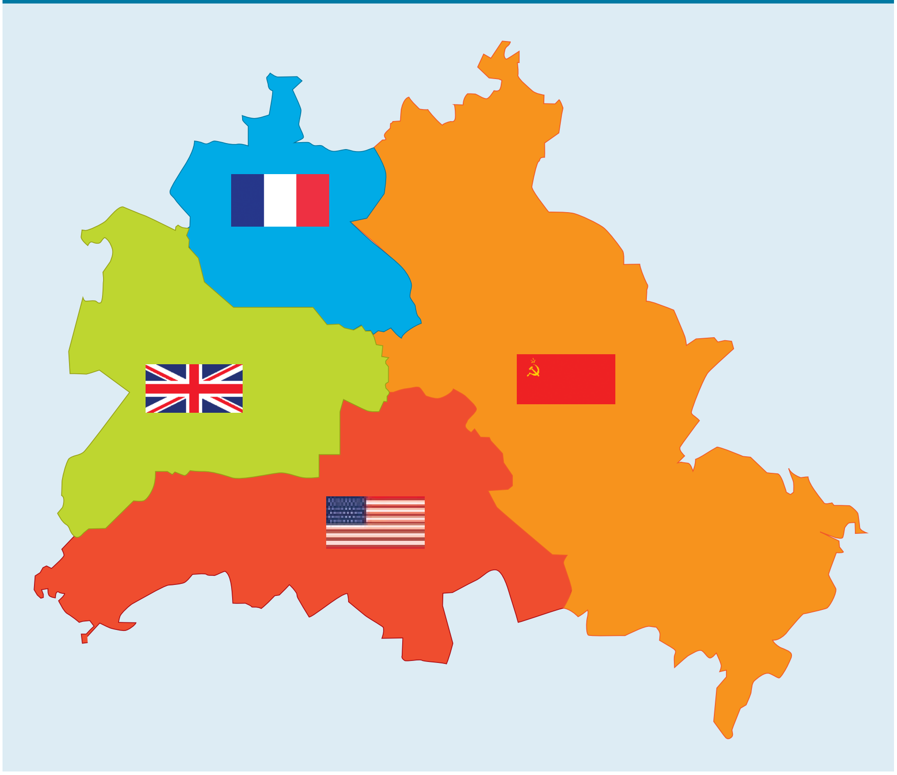
Although cake-cutting algorithms were the object of pure academic curiosity for most of the second half of the 20th century, in the past two decades it has become apparent that this line of research can be successfully applied to important real-world problems. In their 1996 book, Brams and Taylor[11] discussed at length how cake-cutting algorithms can be applied to high-stakes negotiations (they analyzed the 1974 Panama Canal treaty negotiations) and to divorce settlements (they analyzed a real case, *Jolis v. Jolis*, decided in 1981). Cake-cutting algorithms have also been commercialized by companies like Fair Outcomes, Inc. (http://www.fairoutcomes.com/).

Cut and choose. The first cake-cutting algorithm, which has been independently rediscovered by parents for millennia, is the famous cut-and-choose algorithm for dividing a cake between two players, or children. The first player starts by dividing the cake into two pieces he values equally. The second player then chooses the piece he prefers, and the first player receives the remaining piece.[a]

As children with sweet teeth, my brother and I would often use cut and choose to avoid disputes regarding the size of homogeneous pieces of cake (or mousse portions). However, cut and choose is fair even in a more general setting where the divided good is heterogeneous. Indeed, cut and choose is guaranteed to produce a proportional allocation; that is, each player receives a piece he values at 1/2 the value of the whole cake. To see how this works, note that the first player values each piece at exactly 1/2, while the second player receives his preferred piece, which must be worth at least 1/2. Moreover, the algorithm produces an envy-free allocation, with each player liking his own piece at least as much as the other piece. In fact, in the case of two players the concepts of envy-freeness and proportionality coincide; one of the two pieces must be worth at least 1/2 to a player, and if the player likes his piece better, we can conclude that the player's own piece is worth 1/2, so envy-freeness implies proportionality. Conversely, if a player's piece is worth at least 1/2, the other piece must be worth at most 1/2, or (weakly) less than the player's own piece, hence proportionality implies envy-freeness. Here, I implicitly assume that players' valuations are additive; that is, a player's sum of values for two disjoint pieces of cake is equal to the player's value for their union—a standard assumption also made in the following sections.

a Here, I assume the players truthfully follow the algorithm, an assumption reconsidered in the section on the game-theoretic viewpoint.

Figure 1. Berlin divided among the four victorious allies following World War II.



Dubins-Spanier. As we move from the two-player setting to the n-player setting, achieving fairness becomes more difficult. Nevertheless, several elegant algorithms guarantee proportional allocations, including an especially intuitive one proposed in 1961 by Dubins and Spanier[18] that works like this: In each stage, a referee slowly moves a knife over the cake (imagine a rectangular cake) from left to right. When the knife reaches a point such that the piece of cake to the left of that point is worth $1/n$ to one of the players, this player shouts "stop!," the referee makes a cut, and the piece of cake to the left of the cut is given to that player. The satisfied player and allocated piece are then removed, and the process is repeated with the remaining players and the leftover cake until only one player is left; the last player receives the unclaimed piece.

This algorithm produces proportional allocations. Indeed, each player other than the last receives a piece of cake he values at $1/n$. The value of the last player for each of the allocated pieces is at most $1/n$; therefore, (using additivity) the player's value for the unclaimed piece is at least $1 - (n - 1)(1/n) = 1/n$.

The Dubins-Spanier algorithm can be implemented through a discrete procedure, without the impartial referee, though imagining the referee interpretation is more fun. At each stage, each player can simply make a mark so the piece of cake to the left of the mark is worth $1/n$; simulating the algorithm, this is where the player would stop the referee. We then allocate the piece of cake to the left of the leftmost mark to the player who made the mark. The discrete version resembles an algorithm suggested by Banach and Knaster around 1944; see, for example, Brams and Taylor.[10]

Unfortunately, the Dubins-Spanier algorithm does not necessarily yield envy-free allocations. While players do not envy other players allocated earlier, they could easily envy players allocated later; for example, assume there are three players and that player 1 said to stop first and player 2 second, and player 3 received the remaining piece. Player 2 has value of at most 1/3 for the piece of player 1 (because player 2 did not stop the referee), values his own piece at exactly 1/3 but may have value as great as 2/3 for the piece of player 3. Before addressing the issue of envy-freeness, though, I present an elegant algorithm that also aims for proportionality but achieves its goal more efficiently.

Even-Paz. The algorithm designed by Even and Paz[20] in 1984 takes the ideas of Dubins and Spanier one step further. Assume the number of players is a power of 2. Like the discretized version of the Dubins-Spanier algorithm, each time the procedure is executed, the players make marks where the cake to the left of the mark is valued at 1/2 (rather than $1/n$ as before). The twist is that rather than remove a single player the algorithm separates the players into two subsets of equal size, such that all marks made by the players of the first subset lie to the left of the marks made by the players of the second subset. The players in the first subset then receive the piece of cake to the left of their rightmost mark, while the players in the second subset receive the remaining cake. The procedure is applied recursively to the two subsets of players and two pieces of cake, until each piece is claimed by a single player.

While proportionality is well understood, envy-freeness is a far more elusive property.

Each time the procedure is called, half of the players receive at least half of the cake (by value). A single player participates in exactly $\lg n$ calls to the procedure; each player therefore receives a piece of cake worth at least $(1/2)^{\lg n} = 1/n$. We conclude that proportionality is guaranteed, though envy-freeness is not. Intuitively, the Even-Paz algorithm requires making significantly fewer marks than the Dubins-Spanier algorithm; I show how to make this intuition more precise later when discussing the complexity of cake cutting.

Selfridge-Conway. While proportionality is well understood, envy-freeness is a far more elusive property. Selfridge and Conway in around 1960 designed a delightful algorithm (see the survey by Brams and Taylor[10]) that guarantees envy-free allocations for three players:

Stage 0. Player 1 divides the cake into three equal pieces according to his valuation. Player 2 trims the largest piece, or cuts off a slice, such that there is a tie between the two largest pieces in his view. We call the original cake without the trimmings Cake 1 and the trimmings Cake 2;

Stage 1 (division of Cake 1). Player 3 chooses one of the three pieces of Cake 1, the largest according to his valuation. If player 3 did not choose the trimmed piece, player 2 is allocated the trimmed piece. Otherwise, player 2 chooses one of the two remaining pieces. Either player 2 or player 3 receives the trimmed piece; we denote that player by T and the other player by T'. Player 1 is allocated the remaining (untrimmed) piece; and

Stage 2 (division of Cake 2). T' divides Cake 2 into three equal pieces according to his valuation. Players T, 1, and T' choose the pieces of Cake 2, in that order.

The division of Cake 1 is envy-free: Player 3 chooses first; player 2 likes the trimmed piece and another piece equally and is guaranteed to receive one of these two pieces; and player 1 is indifferent judging the two untrimmed pieces and indeed receives an untrimmed piece.

Dividing Cake 2 is more subtle. Player T goes first and hence does not envy the others; and T' is indifferent weighing the three pieces of Cake 2. Player 1 does not envy T' but may prefer the piece of Cake 2 allocated to T to his own piece of Cake 2. However, at the end of stage 1, player 1 has what Brams and Taylor[10] called an "irrevocable advantage" over T. Indeed, even if we allocated all of Cake 2 to T, we would reconstruct just one of the original three pieces cut by player 1 (worth 1/3 to him); but player 1 already received a piece worth 1/3 at the end of stage 1.

Fortunately, we do not have to verify proportionality separately. Proportionality is always implied by envy-freeness, using the same argument employed for two players; in the case of n players, any partition of the cake into n pieces must include a piece worth at least $1/n$ to a given player; envy-freeness then implies the value of the player's own piece is at least $1/n$. The Selfridge-Conway algorithm is therefore an excellent solution for the three-player case, though an extension of their ideas to the n-player case would have to wait another three decades.

Brams-Taylor. In 1992, Brams and Taylor announced a breakthrough—the first envy-free cake-cutting algorithm for an arbitrary number of players.[10] To understand some of its principles, consider the case of four players: Player 1 cuts the cake into four pieces he values equally. If all four players agree that the four pieces are of equal value, the pieces are handed out arbitrarily, and we are done. Otherwise, the players play the "irrevocable advantage sub-game," achieving an envy-free partial allocation of the cake. There is an unallocated leftover piece, but two of the players now have an irrevocable advantage over each other; that is, each player does not envy the other no matter how the leftover piece is allocated.

A similar procedure is then applied recursively to the leftover piece. With each step, the number of pairs of players with an irrevocable advantage over each other increases. Ultimately, all players who disagree about the division of the leftover piece being equal will have an irrevocable advantage over players who agree, and allocating the leftover cake between the latter group of players would yield a complete envy-free allocation.

Game theory views people and software agents alike as rational, yet scheming, selfish creatures who will stop at nothing to maximize their own utility.

Like the Selfridge-Conway algorithm, players in the irrevocable-advantage sub-game trim pieces to create ties, though the approach calls for more pieces than players. Crucially, the irrevocable advantage sub-game is invoked only when there is minimum disagreement among players. The idea is to leverage this disagreement by letting players iteratively trim and choose pieces until the value of the leftover crumb is, intuitively, smaller than the level of disagreement. Specifying these ideas formally is a technical challenge; a description of the four-player special case of the algorithm[10] includes 20 steps.

Unfortunately, the celebrated result of Brams and Taylor suffers from a major flaw, especially when seen through a computational lens. Although the algorithm is guaranteed to terminate with a complete, envy-free allocation, its running time is unbounded. Specifically, the number of operations performed by the algorithm in the irrevocable advantage sub-game (until a sufficiently small crumb is obtained) can be made arbitrarily large by choosing appropriate valuations for the players. Saberi and Wang[37] proposed a bounded envy-free algorithm for the five-player case, but it requires moving knives, and the n-player case remains open. Is envy-free cake cutting inherently complex? I explore this question in the next section.

Complexity of Cake Cutting

Reasoning about the complexity of cake-cutting algorithms requires a model specifying what such an algorithm can do. The one Robertson and Webb[35] proposed in their 1998 book is described here.

I refer to the left boundary of the rectangular cake as 0 and to the right boundary as 1, so the cake itself is represented by the interval [0, 1] of real numbers between 0 and 1. For a piece of cake X (which is just a subset of [0, 1]), I write $V_i(X)$ to denote the value of player i for the piece X. The elegant model of Robertson and Webb limits cake-cutting algorithms to two types of queries:

Evaluation. Asks a player i for his value for the subinterval between two given points x and y: $\text{eval}_i(x, y) = V_i([x, y])$; and

Cut. Asks a player i to mark a subinterval worth a given value α starting at a given point x: $\text{cut}_i(x, \alpha) = y$ such that $V_i([x, y]) = \alpha$.

At first blush this model may seem restricted, but all the algorithms described earlier can be simulated using evaluation and cut queries. Cut and choose requires only two queries, $\text{cut}_1(0, 1/2)$, which returns a point y such that $V_1([0, y]) = 1/2$, and $\text{eval}_2(0, y)$. If the answer to the second query is at least 1/2, player 2 is allocated $[0, y]$ and player 1 the other piece, $[y, 1]$. If the answer is less than 1/2, the allocation is flipped.

Now consider the Selfridge-Conway algorithm. Let us verify that we can simulate stage 0 using evaluation and cut queries. We first ask a $\text{cut}_1(0, 1/3)$ query, and, using the point y that is returned, we ask another $\text{cut}_1(y, 1/3)$ query. This cuts the cake into three subintervals $[0, y]$, $[y, z]$, $[z, 1]$ each worth 1/3 to player 1. We next ask player 2 to evaluate the three subintervals. Say $[0, y]$ is the most valuable and $[y, z]$ is the second most valuable; we then ask player 2 a $\text{cut}_2(0, V_2([0, y]) - V_2([y, z]))$ query to trim the most valuable piece.

I informally claimed earlier that the Even-Paz algorithm is more efficient than the (discrete version of the) Dubins-Spanier algorithm. We are now positioned to make this intuition precise. Both algorithms employ one cut query per player, using the left boundary of the remaining cake as the value of x, in each iteration or recursive call. Under the Dubins-Spanier algorithm the number of players decreases by one per iteration; the number of required cut queries is therefore

$$n + (n-1) + (n-2) + \cdots + 2 = (n+2)(n-1)/2,$$

that is, the number of queries is on the order of n^2. Assuming again for simplicity that n is a power of 2, the Even-Paz procedure is called once with a set of n players as input, twice with sets of $n/2$ players, four times with sets of $n/4$ players, and so on. Overall, the algorithm employs

$$1 \cdot n + 2 \cdot (n/2) + 4 \cdot (n/4) + \cdots + (n/2) \cdot 2$$

cut queries. This sum equals $n \lg n$ because the value of each term is n and the number of terms is $\lg n$. If there are 1,000 players, the recursive Even-Paz algorithm would reduce the required number of cut queries from roughly 500,000 to around 10,000.

In light of this huge improvement, it is natural to ask whether further improvement is possible; that is, are there proportional cake-cutting algorithms that require significantly fewer than $n \lg n$ queries? A partial answer was given by Woeginger and Sgall,[43] who focused on cake-cutting algorithms that allocate connected pieces of cake to players; that is, each player is allocated a subinterval $[x, y]$ of $[0, 1]$. Woeginger and Sgall showed that indeed any cake-cutting algorithm that allocates connected pieces requires at least $c \cdot n \lg n$ queries in the Robertson-Webb model, where c is a constant that does not depend on the number of players n. Interestingly, the Even-Paz algorithm does allocate connected pieces of cake; in contrast, the Selfridge-Conway algorithm does not have this property because a player's piece of Cake 1 may not be connected to the player's piece of Cake 2. Is it still possible to do better by allocating disconnected pieces? Even this question was settled in the negative in a beautiful paper by Edmonds and Pruhs,[19] who extended the result of Woeginger and Sgall to all (discrete) cake-cutting algorithms, crowning the algorithm of Even and Paz as the provably ultimate proportional cake-cutting algorithm (at least in the sense of complexity).

The complexity of envy-free cake cutting is more enigmatic. As mentioned earlier, the Brams-Taylor algorithm, which eventually terminates with an envy-free allocation, is not bounded in terms of the number of queries it can make. Ideally, in the absence of positive guarantees, we would like to be able to establish that bounded envy-free cake-cutting algorithms do not exist.

Stromquist[41] took a first step in this direction, establishing such a nonexistence result under the (by now familiar) assumption that the algorithm must allocate connected pieces, a result that was strengthened by Deng et al.[17] These results hold even for the case of three players. This is surprising because the Selfridge-Conway algorithm yields envy-free allocations for three players with a bounded number of queries (fewer than 20) albeit with possibly disconnected pieces. The result highlights the fundamental difference between connected and possibly disconnected cases.

When connected pieces are not assumed, the only existing result[34] says that the number of queries, in the Robertson-Webb model, required to compute an envy-free allocation is at least on the order of n^2. Although this bound presumably gives only an inkling of how difficult computing envy-free allocations actually is, it is conceptually interesting in that it separates the complexity of proportional and envy-free cake cutting; while proportional cake cutting requires at most $n \lg n$ queries, envy-freeness requires at least n^2; that is, achieving envy-freeness is provably harder than achieving proportionality.

The difficulty of envy-free cake cutting would seem to draw on the richness of player valuations, but this turns out not to be the case. Indeed, Kurokawa et al.[27] showed earlier this year, inter alia, that the general envy-free cake-cutting problem is equally hard when each player's value for the cake is restricted to be uniformly distributed on a (not necessarily connected) piece of cake; that is, a bounded algorithm exists for these severely restricted valuations—known as "piecewise uniform valuations"—if and only if a bounded algorithm exists for the general case. This insight further narrows the search for an impossibility result.

Since the 1940s, the computation of envy-free cake divisions has baffled many great minds across multiple disciplines.[b] Settling this problem once and for all is an important challenge for theoretical computer science.

A Game-Theoretic Viewpoint

So far we have assumed that players honestly follow an algorithm's instructions. In sharp contrast, game theory views people and software agents alike as rational, yet scheming, selfish creatures who will stop at nothing

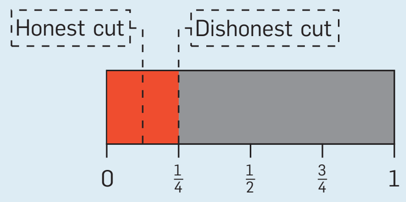
b However, computation of approximately envy-free divisions can be done through the techniques of, say, Lipton et al.[28]

to maximize their own utility. Taking this point of view inspires us to rethink how we cut cake.

To illustrate game-theoretic ideas, we revisit the simple cut-and-choose algorithm. If player 1 honestly does as instructed, he is guaranteed to get a piece of cake worth 1/2. However, suppose player 1 manipulates the algorithm by cutting the cake into two unequal pieces. In this case, player 2 might choose the more valuable piece, leaving player 1 with value less than 1/2. Brams et al.[8] assumed that players never lie about their valuations unless it guarantees them more valuable pieces, regardless of the actions of the other players. According to this notion, the cut-and-choose algorithm encourages honesty.

However, the typical game-theoretic approach advocates a more stringent notion of truthfulness; the algorithm must reward truth telling even if players have full information about one another; that is, players must not be able to gain from manipulating the algorithm, regardless of the actions of others (contrast this with the previous notion). This strong notion of truthfulness is called "strategyproofness." Unfortunately, it is easy to see that the cut-and-choose algorithm is not strategyproof. To show this, I offer an example where honesty fails. Representing the cake again as the interval [0, 1], suppose player 1 desires only the subinterval [0, 1/4] and values this interval uniformly; that is, he wants to receive a piece containing as much of [0, 1/4] as possible. Player 2 values the entire cake uniformly, and so wants a piece as large as possible. If player 1 followed the algorithm, he would produce the equally valued pieces [0, 1/8] and [1/8, 1] (see Figure 2). Player 2 would then choose the piece [1/8, 1], leaving player 1 a piece worth 1/2. If, however, player 1 divided the cake into the pieces [0, 1/4] and [1/4, 1], then player 2 would again choose the larger piece, leaving player 1 with the piece of cake [0, 1/4] he values as much as the entire cake. Likewise, all the cake-cutting algorithms discussed earlier are not strategyproof.

Figure 2. Example of valuations showing that cut and choose is not strategyproof; player 1 desires the colored subinterval uniformly, and player 2 values the entire cake uniformly.



In 2010, a simple strategyproof cake-cutting algorithm was discovered independently by Chen et al.[13] and by Mossel and Tamuz.[30] Now suppose we have a magical method for partitioning the cake into n pieces $X_1,\ldots,X_n$ such that each player i has value exactly $1/n$ for each of these pieces (not just the player's own); that is, $V_i(X_j) = 1/n$ for every j. Following Chen et al., I refer to such a partition as "perfect." The algorithm first computes a perfect partition, then gives each player a random piece. Envy-freeness is clearly guaranteed ex post, or even after the allocation is made, because each player is indifferent as to all possible pieces.

To understand why the algorithm is strategyproof, suppose player i manipulates the algorithm, and so can affect only the computation of the perfect division (the random assignment is independent of the player's actions), and suppose the manipulation gave rise to a different partition $X'_1, \ldots ,X'_n$. The crux of the argument is that for any partition, the expected value of a random piece is exactly $1/n$ because

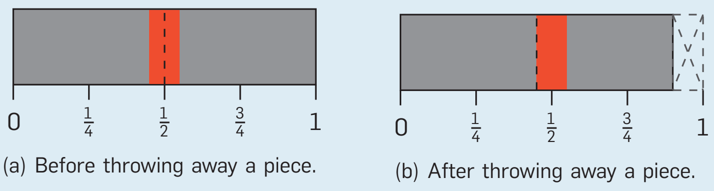
$$(1/n)V_i(X'_1) + \cdots + (1/n)V_i(X'_n) = (1/n)$$
$$(V_i(X'_1) + \cdots + V_i(X'n)) = (1/n)\ 1 = 1/n.$$

Every possible manipulation would therefore yield an expected value of $1/n$ for player i, exactly the value he receives for playing along.[c]

However, before rejoicing, recall that we must still find a way to compute a perfect partition. Alon[1] showed that perfect partitions always exist in a general setting (and bounded the number of cuts required to achieve them), but his proof is non-constructive, establishing existence without explicitly constructing a partition with the desired properties. However, Chen et al.[13] showed that perfect partitions can be computed efficiently when valuations have a piecewise constant structure,[d] meaning each player desires only certain pieces of cake and values each piece uniformly. To motivate this (rather restrictive) assumption, imagine the cake represents time for TV advertising. A toy company might be interested in time intervals associated only with children's TV shows. It could be indifferent between equal slots within the same subinterval but would presumably prefer a 30-second slot during the latest episode of "SpongeBob SquarePants" to a 30-second slot following a rerun of "Teenage Mutant Ninja Turtles."

Randomization requires somewhat stronger assumptions (such as assuming players wish to maximize their expected utility), so it is natural to ask whether fairness and truthfulness can be guaranteed without resorting to randomization. Chen et al.[13] established such a result, albeit only under the extremely restrictive class of piecewise uniform valuations (mentioned in the previous section). Recall that a player's valuation satisfies this property if the player has a single de-

Figure 3. Throwing away a piece from the right side of the cake increases the social welfare of the best proportional allocation. Player 1 desires the colored piece, and player 2 values the entire cake uniformly.


0
1/4
1/2
3/4
1
(a) Before throwing away a piece.
0
1/4
1/2
3/4
1
(b) After throwing away a piece.

c An underlying assumption is that players are risk neutral; that is, they care only about their own expected value.

d They establish a more general but less intuitive result.

sired piece of cake (not necessarily connected) valued uniformly, so the player simply wants as much of this desired piece as possible. If we again imagine the cake to be time, but in the context of access to a shared backup server, then conceivably a user would be equally interested in time intervals when his computer is idle. Note that piecewise uniform valuations are in particular piecewise constant.

Despite this progress, the design of strategyproof cake-cutting algorithms is still largely an open problem, first because the algorithms described earlier (especially the deterministic one) can handle only restricted valuations,[e] and second because these algorithms cannot be simulated in the Robertson-Webb model. The underlying assumption is that players report their full preferences to a central authority, an assumption that does not impose an unreasonable communication burden because piecewise constant valuations can be represented concisely, though in an informal sense does preclude a distributed implementation.

Optimizing Welfare

Envy-freeness and proportionality are notions that pertain to individuals; for example, a proportional cake-cutting algorithm guarantees that each and every individual does well. Sometimes it is appropriate, though, to take a global view and promote the interests of society as a whole. The social welfare is a quantification of the happiness of society, typically in two flavors: utilitarian, which in our context is the sum of values players have for their allocations, and egalitarian, which is determined by the lowest value any player has for his piece of cake. I focus on the utilitarian version here.

As an aside, notions of social welfare require an interpersonal comparison of values. So far, when I said a piece of cake is worth 1/2 to a player, I meant it is worth 1/2 of the player's value for the entire cake. One player's happiness for receiving 1/2 of his value of the cake may not be equal to another player's happiness for receiving the same fraction. In contrast, to study utilitarian or egalitarian social welfare we must assume all players have the same value for the whole cake, say \$1, and therefore 1/2 of a player's value for the cake is literally worth \$0.5.

Sometimes it is appropriate to take a global view and promote the interests of society as a whole.

Intuitively, tension exists between the interests of individuals and of society. Several years ago, two groups of researchers set out to make this intuition precise. Bertsimas et al.[6] and Caragiannis et al.[12] independently coined the term "price of fairness" for the worst-case ratio between the social welfare of the optimal allocation and the social welfare of the best fair allocation (compare with the well-known price of anarchy[36]). Any notion of social welfare, as well as any notion of fairness, can be plugged into this definition. A price of fairness of 2 would mean, for instance, there are examples where the social welfare of the best fair allocation is at most 50% of what it could be if the fairness restriction were removed.

To see why we must pay for fairness with (utilitarian) social welfare, consider the following scenario: Partition the cake (represented by $[0, 1]$) into $\sqrt{n}$ disjoint subintervals, each of length $1/\sqrt{n}$. Each of the first "large" $\sqrt{n}$ players desires only one of these subintervals; no two large players desire the same subinterval, and each large player values his subinterval uniformly. The remaining $n - \sqrt{n}$ "small" players value the whole cake uniformly. Any proportional allocation must allocate a piece of length $1/n$ to each of the small players, leaving only $1/\sqrt{n}$ (by length) to the large players. Although the valuations of the large players are denser, their sum of values for a piece of cake of length $1/\sqrt{n}$ is at most 1, while the small players contribute $1/n$ each to the social welfare and less than 1 together. Overall, the social welfare is smaller than 2. In contrast, the welfare-maximizing allocation divides the entire cake among the large players, giving each a piece of cake worth 1 and securing social welfare of $\sqrt{n}$. The price of proportionality is at least the ratio between the latter and former values, or at least $\sqrt{n}/2$. The price of envy-freeness is at least as high because envy-freeness implies proportionality.

Aumann and Dombb[3] subsequently studied the price of fairness under

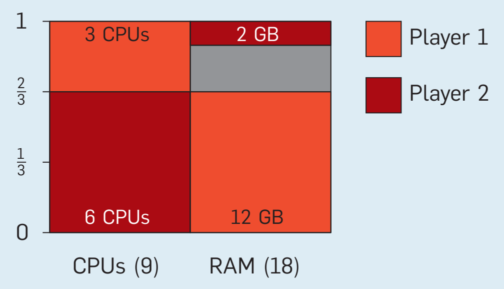
e An extension to more expressive valuations would have to avoid the daunting impossibility results covered in the literature.[27,38]

the assumption that connected pieces must be allocated. An especially interesting insight in this context is the so-called "dumping paradox"[2]: By discarding pieces of cake, one can increase the social welfare of the best proportional, or envy-free, allocation; for example, say player 1 uniformly values a very small interval centered around the midpoint 1/2, and player 2 values [0, 1] uniformly. A proportional allocation allocating the entire cake and making only one cut would have to make the cut at 1/2 (see Figure 3a). This division is suboptimal in terms of social welfare because player 1 would be twice as happy to get an additional tiny piece from player 2. Curiously, after discarding a small piece from the right side of the cake, we can produce a proportional division by making the cut just to the left of the desired interval of player 1 (see Figure 3b); the social welfare increases from 1 to slightly less than 3/2. Arzi et al.[2] demonstrated that by discarding some of the cake it is possible to gain as much as a factor of $\sqrt{n}$.

A high price of fairness means there are examples where fair allocations are severely suboptimal from society's point of view. Nevertheless, these examples may be rare and do not preclude the possibility of usually obtaining high social welfare even under fairness constraints. Cohler et al.[14] investigated the problem of optimizing social welfare under envy-freeness constraints. For piecewise constant valuations (as defined earlier), welfare-maximizing proportional or envy-free allocations can be computed in polynomial time. This result can be leveraged to obtain (in polynomial time) fair allocations for general valuations that are arbitrarily close to optimal. Computing optimal fair cake divisions with connected pieces is much more difficult,[5] even if we abandon fairness completely and focus just on optimizing social welfare.[4]

Now I ask, with tongue in cheek, how good are optimal cake divisions? Economists would say a good allocation must be Pareto-efficient, in the sense that no other allocation is valued at least as highly by all players and is strictly better for at least one player. It turns out there are examples where no welfare-maximizing envy-free allocation is Pareto-efficient, even when only three players have piecewise constant valuations.[7] Should we sacrifice social welfare to obtain Pareto-efficiency? How much must be sacrificed? More generally, what would constitute an ideal cake division?[9] These conceptual questions may lead to significant technical insight on the role of optimization in cake cutting.

Figure 4. DRF allocation for an example in the text. Shaded regions represent allocated shares.



Cutting Computational Cakes

We have seen here that the computer science perspective can contribute to the study of cake cutting. Now I explore how cake cutting (or at least closely related models from fair-division theory) can be applied to problems in computer science:

Consider a setting with multiple homogeneous and divisible resources (compare this with the standard cake-cutting scenario, which has a single heterogeneous divisible resource). Leontief preferences, first conceptualized by Wassily Leontief, represent situations where a player demands the resources in fixed proportions; for example, a jeweler may need five grams of gold and 10 grams of silver to produce a ring, so given one kilogram of gold and two kilograms of silver, the jeweler can make 200 rings. However, given one kilogram of gold and three kilograms of silver, the jeweler can still, strangely enough, make only 200 rings (perhaps the jeweler is known for a specific design) and is hence indifferent between these two allocations.

Technological advances (such as cluster computing) provide a new impetus for studying resource allocation under Leontief preferences. Indeed, a user may plausibly wish to run many instances of a task that requires fixed proportions of system resources (such as CPU and memory). Such a user exhibits Leontief preferences with respect to personal allocation of system resources. Crucially, users can have heterogeneous demands; for example, some may have CPU-intensive tasks and others memory-intensive tasks. There is an existing body of work on fair-resource allocation, including the max-min fairness policy,[16] but it focuses on a single resource type. Even in modern cluster-computing environments with multiple resources, state-of-the-art algorithms allocate resources in "slots," or bundles containing fixed amounts of each resource. Although slots provide a usable single-resource abstraction, the methodology clearly leads to inefficiencies.

Fortunately, the theory of fair division already provides an almost tailor-made framework for tackling this modern technological challenge. The first authors to recognize this were Ghodsi et al.,[24] who, in an impressive paper nicely combining theory and practice, also suggested a solution—the dominant resource fairness, or DRF, mechanism; to illustrate DRF, consider the following example from the paper:

A system includes nine CPUs and 18GB RAM. There are two players, one wishing to run as many instances as possible of a task that requires one CPU and 4GB RAM and the other with a task requiring three CPUs and 1GB RAM. Each instance of the task of player 1 requires 1/9 of the total CPU in the system and 2/9 of the total RAM. The latter fraction of RAM is larger, hence we say the dominant resource of player 1 is RAM. Likewise, the fractions of CPU and RAM for player 2 are 1/3 and 1/18, respectively; hence, for player 2, CPU is the dominant resource. The DRF mechanism allocates as many tasks as possible while equalizing the dominant shares, or fractions of dominant resources each user receives. In this example, DRF allocates three tasks to player 1 for a total allocation of three CPUs and 12GB RAM and two tasks to player 2 for a total allocation of six CPUs and 2GB RAM (see Figure 4). The dominant shares are equal, with 12/18 = 2/3 of the RAM for player 1 and 6/9 = 2/3 of the CPU for player 2. Allocating additional tasks (or even fractions thereof) is impossible because the CPU pool is saturated.

Ghodsi et al. demonstrated that the DRF mechanism satisfies each and every one of the axiomatic properties mentioned here, being strategyproof and producing allocations that are Pareto-efficient, envy-free, and proportional; in this setting envy-free allocations are not necessarily proportional. From an economics viewpoint, Ghodsi et al.'s theoretical contribution is the discovery that an important mechanism known as the "egalitarian equivalent rule"[33] exhibits especially compelling properties when players have Leontief preferences, a discovery that generated considerable excitement in computer science and led to a string of papers that built on these ideas; see, for example, Friedman et al.,[22] Gutman and Nisan,[25] and Parkes et al.[32]

Nevertheless, the work so far is just the tip of the iceberg. Perhaps most important, existing theoretical models do not capture dynamic settings where users can arrive and depart and change their demands over time. One obstacle is conceptual; not immediately clear is how to interpret fairness properties like envy-freeness when some players arrive before others. Earlier this year, Kash et al.[26] took the first conceptual and technical steps toward a dynamic model, but the assumption that drives their work—that allocations of resources are irrevocable—may not hold in practice. Applying more practical versions of the model in the real world is one of the most compelling challenges at the border of computer science and economics today.

Conclusion

Most of the computer science articles cited here were published in the past five years, and many other publications are available. Computer scientists' interest in cake cutting has been piqued, and I expect to see a surge of work yielding smarter cake-cutting algorithms, nifty applications, and perhaps even deployed systems.

This article should be viewed as an invitation to cake cutting, a field as much fun as it is scientifically significant, and that involves great intellectual challenges. Magdon-Ismail et al.[29] said it like this: "Cake cutting is not a piece of cake."

Acknowledgment

This work is supported in part by the National Science Foundation under grant CCF-1215883. C

References

1. Alon, N. Splitting necklaces. *Advances in Mathematics 63* (1987), 241–253.
2. Arzi, O., Aumann, Y., and Dombb, Y. Throw one's cake—And eat it, too. In *Proceedings of the Fourth International Symposium on Algorithmic Game Theory* (2011), 69–80.
3. Aumann, Y. and Dombb, Y. The efficiency of fair division with connected pieces. In *Proceedings of the Sixth International Workshop on Internet and Network Economics* (2010), 26–37.
4. Aumann, Y., Dombb, Y., and Hassidim, A. Computing socially efficient cake divisions. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems* (2013).
5. Bei, X., Chen, N., Hua, X., Tao, B., and Yang, E. Optimal proportional cake cutting with connected pieces. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence* (2012), 1263–1269.
6. Bertsimas, D., Farias, V.F., and Trichakis, N. The price of fairness. *Operations Research 59*, 1 (2011), 17–31.
7. Brams, S.J., Feldman, M., Morgenstern, J., Lai, J.K., and Procaccia, A.D. On maxsum fair cake divisions. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence* (2012), 1285–1291.
8. Brams, S.J., Jones, M.A., and Klamler, C. Better ways to cut a cake. *Notices of the AMS 53*, 11 (2006), 1314–1321.
9. Brams, S.J., Jones, M.A., and Klamler, C. *N*-person cake-cutting: There may be no perfect division. *The American Mathematical Monthly 120*, 1 (2013), 35–47.
10. Brams, S.J. and Taylor, A.D. An envy-free cake-division protocol. *The American Mathematical Monthly 102*, 1 (1995), 9–18.
11. Brams, S.J. and Taylor, A.D. *Fair Division: From Cake Cutting to Dispute Resolution.* Cambridge University Press, Cambridge, U.K., 1996.
12. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P., and Kyropoulou, M. The efficiency of fair division. In *Proceedings of the Fifth International Workshop on Internet and Network Economics* (2009), 475–482.
13. Chen, Y., Lai, J.K., Parkes, D.C., and Procaccia, A.D. Truth, justice, and cake cutting. *Games and Economic Behavior 77* (2013), 284–297.
14. Cohler, Y.J., Lai, J.K., Parkes, D.C., and Procaccia, A.D. Optimal envy-free cake cutting. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence* (2011), 626–631.
15. Conitzer, V. Making decisions based on the preferences of multiple agents. *Commun. ACM 53*, 3 (Mar. 2010), 84–94.
16. Demers, A., Keshav, S., and Shenker, S. Analysis and simulation of a fair queuing algorithm. In *Proceedings of the ACM Symposium on Communications Architectures & Protocols* (1989), 1–12.
17. Deng, X., Qi, Q., and Saberi, A. Algorithmic solutions for envy-free cake cutting. *Operations Research 60*, 6 (2012), 1461–1476.
18. Dubins, L.E. and Spanier, E.H. How to cut a cake fairly. *American Mathematical Monthly 68*, 1 (1961), 1–17.
19. Edmonds, J. and Pruhs, K. Cake cutting really is not a piece of cake. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms* (2006), 271–278.
20. Even, S. and Paz, A. A note on cake cutting. *Discrete Applied Mathematics 7* (1984), 285–296.
21. Foley, D. Resource allocation and the public sector. *Yale Economics Essays 7* (1967), 45–98.
22. Friedman, E.J., Ghodsi, A., Shenker, S., and Stoica, I. *Strategyproofness, Leontief Economies, and the Kalai-Smorodinsky Solution.* Manuscript, 2011; http://www1.icsi.berkeley.edu/~ejf/pfiles/ksgroupsp.pdf
23. Gamow, G. and Stern, M. *Puzzle-Math.* Viking, New York, 1958.
24. Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., and Stoica, I. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the Eighth USENIX Conference on Networked Systems Design and Implementation* (2011), 24–37.
25. Gutman, A. and Nisan, N. Fair allocation without trade. In *Proceedings of the 11th International Conference on Autonomous Agents and Multi-Agent Systems* (2012), 719–728.
26. Kash, I., Procaccia, A.D., and Shah, N. No agent left behind: Dynamic fair division of multiple resources. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-Agent Systems* (2013).
27. Kurokawa, D., Lai, J.K., and Procaccia, A.D. How to cut a cake before the party ends. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence* (2013).
28. Lipton, R.J., Markakis, E., Mossel, E., and Saberi, A. On approximately fair allocations of indivisible goods. In *Proceedings of the Sixth ACM Conference on Electronic Commerce* (2004), 125–131.
29. Magdon-Ismail, M., Busch, C., and Krishnamoorthy, M.S. Cake cutting is not a piece of cake. In *Proceedings of the 20th International Symposium on Theoretical Aspects of Computer Science* (2003), 596–607.
30. Mossel, E. and Tamuz, O. Truthful fair division. In *Proceedings of the Third International Symposium on Algorithmic Game Theory* (2010), 288–299.
31. Moulin, H. *Fair Division and Collective Welfare.* MIT Press, Cambridge, MA, 2003.
32. Parkes, D.C., Procaccia, A.D., and Shah, N. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. In *Proceedings of the 13th ACM Conference on Electronic Commerce* (2012), 808–825.
33. Pazner, E. and Schmeidler, D. Egalitarian equivalent allocations: A new concept of economic equity. *Quarterly Journal of Economics 92*, 4 (1978), 671–687.
34. Procaccia, A.D. Thou shalt covet thy neighbor's cake. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (2009), 239–244.
35. Robertson, J.M. and Webb, W.A. *Cake Cutting Algorithms: Be Fair If You Can.* A.K. Peters Ltd., London, 1998.
36. Roughgarden, T. Algorithmic game theory. *Commun. ACM 53*, 7 (July 2010), 78–86.
37. Saberi, A. and Wang, Y. Cutting a cake for five people. In *Proceedings of the Fifth International Conference on Algorithmic Aspects in Information and Management* (2009), 292–300.
38. Schummer, J. Strategyproofness versus efficiency on restricted domains of exchange economies. *Social Choice and Welfare 14* (1997), 47–56.
39. Steinhaus, H. The problem of fair division. *Econometrica 16* (1948), 101–104.
40. Stromquist, W. How to cut a cake fairly. *American Mathematical Monthly 87*, 8 (1980), 640–644.
41. Stromquist, W. Envy-free cake divisions cannot be found by finite protocols. *The Electronic Journal of Combinatorics 15*, #R11 (2008).
42. Thomson, W. Fair allocation rules. In *Handbook of Social Choice and Welfare, Volume 2, Chapter 21*, K.J. Arrow, A. Sen, and K. Suzumura, Eds. North-Holland Publishing Co., Amsterdam, 2010.
43. Woeginger, G.J. and Sgall, J. On the complexity of cake cutting. *Discrete Optimization 4* (2007), 213–220.

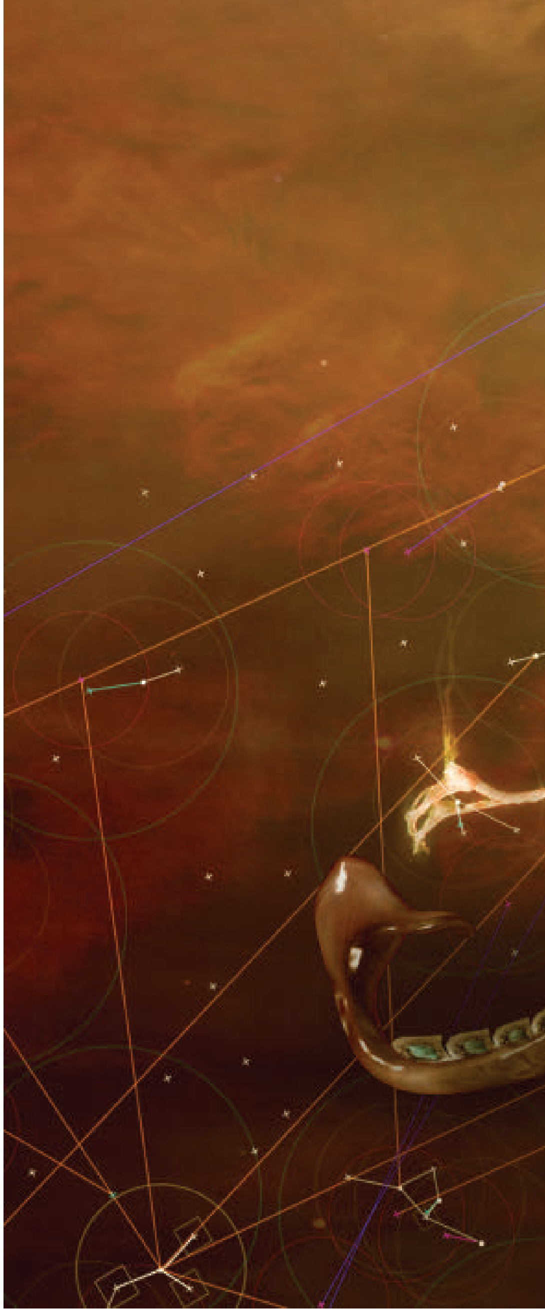
Ariel D. Procaccia (arielpro@cs.cmu.edu) is an assistant professor in the Computer Science Department at Carnegie Mellon University, Pittsburgh, PA.

DOI:10.1145/2483852.2483871

The challenge of developing and using computer models to understand and control the diffusion of disease through populations.

BY MADHAV MARATHE AND ANIL KUMAR S. VULLIKANTI

Computational Epidemiology



AN EPIDEMIC IS said to arise in a community or region when cases of an illness or other health-related events occur in excess of normal expectancy. Epidemics are considered to have influenced significant historical events, including the plagues in Roman times and Middle Ages, the fall of the Han empire in the 3rd century in China, and the defeat of the Aztecs in the 1500s, due to a smallpox outbreak.[9] The 1918 flu pandemic in the U.S. was responsible for more deaths than those due to World War I. The last 50 years have seen epidemics caused by HIV/AIDS, SARS, and influenza-like illnesses. Despite significant medical advances, according to the World Health Organization (WHO), infectious diseases account for more than 13 million deaths a year.[44]

Societal interest in controlling outbreaks is probably just as old as the diseases themselves. Interestingly, it appears the Indians and Chinese knew the idea of variolation to control smallpox as early as the 8th century A.D. Epidemiology is a formal branch of science focusing on the study of space-time patterns of illness in a population and the factors that contribute to these patterns. It plays an essential role in public health by

» key insights

- **Controlling and responding to future pandemics will be challenging due to a number of emerging global trends including increased and denser urbanization, increased local as well as global travel, and a generally older and immuno-compromised population.**
- **Public health epidemiology is a complex system problem. Epidemics, social-contact networks, individual and collective behavior, and public policies coevolve during a pandemic—a system-level understanding must represent these components and their coevolution.**
- **Mathematical and computational models of social networks and epidemic spread and methods to analyze them are critical in public health epidemiology.**
- **Advances in computing, big data, and computational thinking have created entirely new opportunities to support real-time epidemiology.**

ILLUSTRATION BY AS&K VISUAL SCIENCE, WWW.ASK-VISUAL.COM

aiming to understand the processes that lead to ill health as well as the evaluation of strategies designed to prevent illness and promote good health.

Computational epidemiology is an interdisciplinary area setting its sights on developing and using computer models to understand and control the spatiotemporal diffusion of disease through populations. The models may range from descriptive, for example, static estimates of correlations within large databases, to generative, for example, computing the spread of disease via person-to-person interactions through a large population. The disease may represent an actual infectious disease, or following the relatively recent use of this term, it may represent a more general reaction-diffusion process, such as the diffusion of innovation. The populations of interest depend on the disease, including humans, animals, plants, and computers. Similarly, the interactions that must be represented depend on the disease and the populations, including physical proximity for aerosol-borne disease, sexual contact for sexually transmitted diseases, and insect feeding patterns for mosquito-borne diseases.

In an editorial in *Science* (May 2009), Fineberg and Wilson persuasively argue for the role of science in policy as it pertains to real-time epidemiology.[26–28] They go on to identify five areas where science can support real-time epidemiology: pandemic risk; vulnerable populations; available interventions; implementation possibilities; and pitfalls and public understanding.

Computation can play a multifaceted role to support real-time epidemiology. The role of computation becomes all the more important in light of the fact that controlled experiments used to understand scientific phenomena are much harder and often impossible in epidemiology due to ethical and practical reasons. Figure 1 illustrates an end-to-end computationally oriented view to support real-time epidemiology including the five areas identified by Fineberg and Wilson. Computational models help in understanding the space-time dynamics of epidemics. Models and algorithms can be used to evaluate various intervention strategies, including pharmaceutical interventions such as vaccinations and anti-virals, as well as non-pharmaceutical interventions such as social distancing and school closures.

The role of individual behavior and public policies is critical in understanding and controlling epidemics—computational techniques provide a potentially powerful study tool.[24,27] Pervasive computational environments can provide real-time access to models, data, and expert opinion to analysts as an epidemic unfolds. It is important for computational methods and thinking to be developed within an inherently multidisciplinary setting. For example, inference methods will need to be developed for specific problems in conjunction with statisticians and biologists. The problem of vaccine production, allocation, and usage must be solved in conjunction with social, economic, and behavioral scientists to account for efficient use, keeping in mind social and economic factors. User interfaces and decision support environments would have to be developed in conjunction with psychologists and statisticians to avoid pitfalls in human judgment. Details omitted here because of space limitations can be found in supplemental material.[44]

Mathematical Modeling

The first mathematical model in epidemiology is credited to Bernoulli in 1760.[9] Using mathematical techniques, Bernoulli established that variolation could help increase the life expectancy in the French population. Another systematic and data-driven investigation of the spread of epidemics was by John Snow, a British physician, who analyzed a cholera outbreak in London in 1854.[9] The early 1900s saw seminal advances in mathematical epidemiology. Ross in 1911 found malaria spread due to mosquitoes, and developed a spatial model for the spread of malaria. One of the most significant results from his model was that the spread of malaria could be controlled by reducing the population of malaria below a "threshold"—this is the first instance of the concept of an epidemic threshold. Kermack and McKendrik extended this to develop the first general epidemic model involving ordinary differential equations based on a mass-action model. Their work laid the modern foundations for mathematical epidemiology. (For details, see the sidebar "Modeling Epidemic Processes," supplementary information,[44] Brauer,[9] and Newman.[37])

Modeling Epidemic Processes

The simplest aggregate model is popularly known as the SIR model. A population of size N is divided into three states: susceptible (S), infective (I), and removed or recovered (R).

The following discrete time process describes the system dynamics: each infected person can infect any susceptible person (independently) with probability β, and can recover with probability γ. Let $S(t)$, $I(t)$ and $R(t)$ denote the number of people who are susceptible, infected and recovered states at time t, respectively. Let $s(t) = S(t)/N$, $i(t) = I(t)/N$ and $r(t) = R(t)/N$; then, $s(t) + i(t) + r(t) = 1$.

By the "complete mixing" assumption that each individual is in contact with everyone in the population, it can be shown that the following system of differential equations (known as the SIR model) describes the dynamics:

$$\frac{ds(t)}{dt} = -\beta s(t)\, i(t), \quad \frac{di(t)}{dt} = \beta s(t)\, i(t) - \gamma i(t), \quad \frac{dr(t)}{dt} = \gamma i(t).$$

One of the classic results in the SIR model is there is an epidemic that infects a large fraction of the population, if and only if $R_0 = \beta/\gamma > 1$; R_0 is known as the "reproductive number," and thus much of public health decision making is centered on controlling R_0.

Epidemics on Networks

Let $G(V, E)$ denote a contact graph on a population V—each edge $e = (u, v) \in E$ denotes that the individuals (also referred to as nodes) $u, v \in V$ come into contact. Let $N(v)$ denote the set of neighbors of v.

For the SIR model on the graph G, we have a dynamical process with each node being in S, I or R states. Infection can potentially spread from u to v along edge $e = (u, v)$ with a probability of $\beta(e, t)$ at time instant t after u becomes infected, conditional on node v remaining uninfected until time t—this is a discrete version of the rate of infection in the sidebar "Modeling Epidemic Processes."

Let $\tau(u)$ denote the time that node u would remain in the infected state, and let $\tau = \max\{\tau(u) : u \in V\}$. If a node $v \in V$ gets infected at time t_v, it attempts to infect each susceptible neighbor u with probability $\beta((v, u), t - t_v)$ for $t = t_v + 1, \ldots, t_v + \tau(v)$. After $\tau(v)$ steps, node v switches to state R.

We let $I(t)$ denote the set of nodes that become infected at time t. The sequence $I(t)$, $t = 0, 1, \ldots$ along with the (random) subset of edges on which the infections spread, represent a disease outcome, also referred to as a *dendogram*.

The time series ($|I(t)|$, $t = 0, 1, \ldots$) is referred to as an epidemic curve corresponding to a stochastic outcome. The total number of infections for an outcome is given by $\Sigma_t |I(t)|$. The epidemic peak is the largest time t that maximizes $|I(t)|$. Thus, this dynamical system starts with a configuration in which there are one or more nodes in state I and reaches a fixed point in which all nodes are in states S or R. A popular variant of the SIR model is the SIS model, in which an infected node switches to state S after the infectious duration.

Networked Epidemiology

The analysis in the sidebar "Modeling Epidemic Processes" involves a number of assumptions, chief among them being the complete mixing requirement, which is often unrealistic. Many researchers have extended this in numerous ways, including stochastic models, multiple compartments to represent various subpopulations, branching processes, and chain-binomial models, among others.[9,37] Here we will focus on a recent approach that is inherently computational in nature—*networked epidemiology* (see Figure 2). It seeks to understand the complicated interplay between the three components of computational epidemiology: individual behaviors of agents; unstructured; heterogeneous multiscale networks; and the dynamical processes on these networks. It is based on the hypothesis that a better understanding of the characteristics of the underlying network and individual behavioral adaptation can give better insights into contagion dynamics and response strategies. Although computationally expensive and data intensive, network-based epidemiology alters the types of questions posed, providing qualitatively different insights into disease dynamics and public health policies. It also allows policymakers to formulate and investigate potentially novel and context-specific interventions.

The sidebar "Epidemics on Networks" summarizes the underlying mathematical framework to represent epidemics over a network. A general framework to succinctly specify disease-local dynamical processes on networks is called the graphical discrete dynamical system (GDDS).[7] A GDDS $\mathcal{G}$ is defined as a tuple (G, F, π), where: (a) $G = (V, E)$ is the underlying contact network on a set V of nodes; (b) $F = \{f_v | v \in V\}$ is a set of local functions, (for example, such as the one capturing the localized SIR process)—one function for each node $v \in V$ on some fixed domain, where v computes its state by applying f_v on the states of its neighbors; and (c) a schedule $\mathcal{S}$ over V^* that specifies the order in which the states of the nodes are updated by applying the functions in F. One update of GDDS involves applying the local functions in the order specified by $\mathcal{S}$. Extensions of GDDS can be used to describe multitheory, multinetwork coevolving systems (referred to as CGDDS), wherein multiple networks and the local functions interact and coevolve in time endogenously.

The configuration space of GDDS $\mathcal{G}$ can be conveniently viewed as a Markov chain $\mathcal{M}$. Each node in $\mathcal{M}$ is the state vector of the node states in G. A directed edge from node A and B in M with label p denotes the probability of transition from A to B. In the SIR model, a node can be in one of the three states and thus the Markov chain has $N = 3^n$ vertices. Note that in the case of a complete graph, by symmetry arguments, the chain is exponentially smaller; one needs to keep track of the number of nodes in each state. A succinctly specified partially observable Markov decision process (POMDP) N can be obtained by augmenting CGDDS $\mathcal{G}$ with an appropriate model for observation $\mathcal{O}$ and decision-making $\mathcal{D}$.

Two basic problems in network epidemiology are, given $\mathcal{G}$, to characterize the structure of M, and finding appropriate upper and lower bounds for computing structural properties of M. It is now well accepted that the structure of the underlying contact graph G has a significant impact on the disease dynamics.[37] A fundamental question is to characterize the conditions under which there is a "large" outbreak (that is, when the number of infections is $\theta(n)$, where $n = |V|$). We mention a few of the main results of this kind for the SIR and SIS models; these are somewhat involved to derive here, and we give references for more details. A common approach is to try to characterize the dynamics in terms of the degree distribution of the graph. The simplest case is when G is a complete graph, with a uniform probability β of the disease spread from u to v, and $\tau(u) = 1$. The classical result by Erdos-Renyi[44] implies there is a large outbreak with $\theta(n)$ infections, if and only if $\beta > 1/n$.

The main technique is a branching process analysis.[16,37,44] We note the result for the complete graph is a discrete analogue of the characterization in terms of R_0, with $n\ \beta$, the expected number of infections caused by a node, being the equivalent of R_0. It has been shown that such a threshold for disease spread exists in other well-structured graph classes, such as lattices and random regular graphs.[37,44] It is much more challenging to prove such statements in heterogeneous graphs with less symmetry. Pastor-Satorras et al.[39] use techniques from

Figure 1. Elements of computational real-time epidemiology.

The process starts by developing models of synthetic social contact networks and within host disease progression using diverse datasets that include: surveys, census, social media, serological investigations, and disease surveillance. High-performance computer simulations are then used to study the dynamics aof disease propagation and the effects of various intervention strategies. The results are used by policymakers and analysts to formulate and evaluate various public policies as well as putative societal responses. All these models are refined, based on the simulation results and the policies being studied.

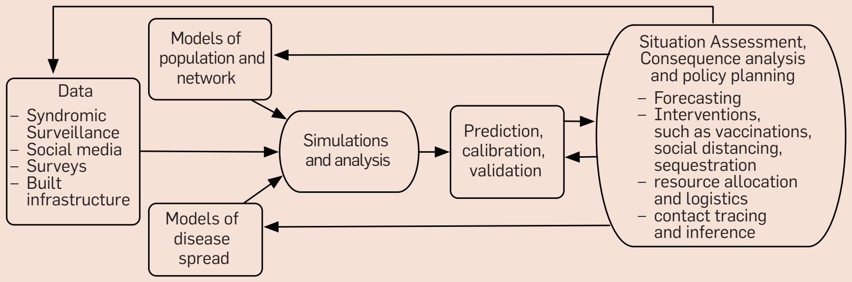

Data
– Syndromic Surveillance
– Social media
– Surveys
– Built infrastructure
Models of population and network
Simulations and analysis
Models of disease spread
Prediction, calibration, validation
Situation Assessment, Consequence analysis and policy planning
– Forecasting
– Interventions, such as vaccinations, social distancing, sequestration
– resource allocation and logistics
– contact tracing and inference

Figure 2. (a) Example showing a contact network on a population of size 6, represented by the set of nodes $\{v_1, v_2, v_3, v_4, v_5, v_6\}$. (b) An example of a dendogram on this contact graph, with the infected sets $I(t)$, $t = 0, 1, 2, 3$ as shown. The red edges represent the edges on which the infections spread. The infection starts at node v_3, and eventually all nodes, except v_1 get infected; the epicurve corresponding to this example is (1, 1, 2, 1), with the peak being at time 2. (c) Another possible dendogram on the same network, with the infection starting at v_1, where all nodes except v_2 get infected.

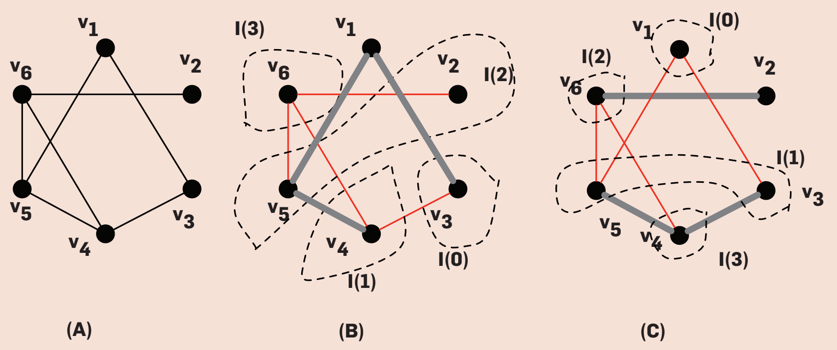
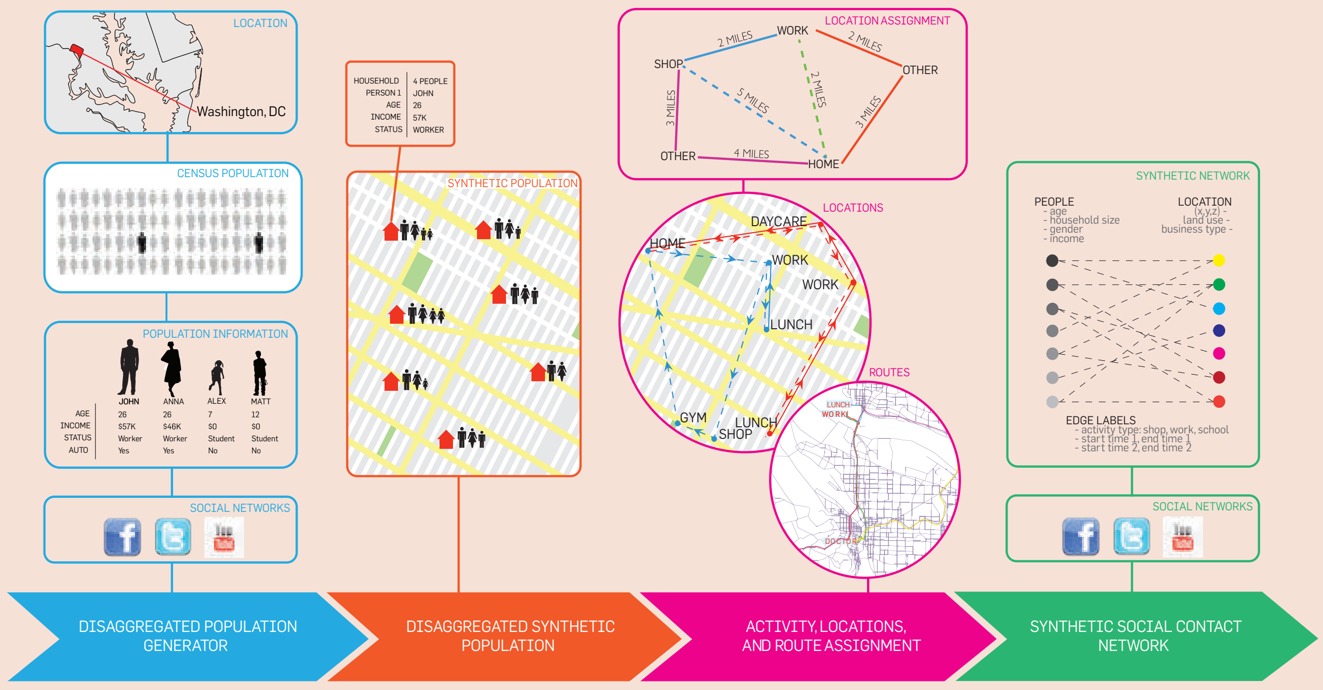


Figure 3. The main steps in the first principles-based construction of synthetic populations and social contact networks.

Step 1 constructs a synthetic population by using various commercial and open source databases. Step 2 assigns daily activities to individuals within a household using activity and time-use surveys[5,21] as well as information available from social media. Step 3 constructs a *dynamic social bipartite visitation network*, represented by a (vertex and edge) labeled bipartite graph G_{PL}, where P is the set of people and L is the set of locations. (Image courtesy of Rachel Robinson.)



statistical mechanics to show that the threshold for epidemics propagation is 0 in scale-free network models, under mean-field assumptions, that is, no matter how small the probability of infection is, there would be a large outbreak. There are two settings for which rigorous results are known without the use of any mean-field assumptions. One is the Chung-Lu model,[12] which is a random graph model in which the probability of an edge (i, j) is proportional to $w_i \cdot w_j$, for a given weight sequence $\mathbf{w}$. The other is classes of expander graphs.[1]

Computational Models

We outline here a general computational approach for networked epidemiology. The first practical and urban-scale application of this approach was described in Eubanks et al.[21] It consists of six broad steps (see "Computational Epidemiology over Networks"). Here, we discuss Steps 1–5; Step 6 will be detailed later. Please refer to the supplementary information[44] for details.

Steps 1–3 synthesize a realistic social contact network of the region under consideration (See Figure 3). Note that it is impossible to build synthetic urban-scale social contact networks solely by collecting field data, although such data can be incorporated into the synthetic population creation process. The networks so constructed are quite different structurally than those produced by simple random graph techniques.[5,22] Interesting similarities as well as differences among urban regions can be found in Barrett.[5] Recently, researchers have included other forms of data and information to extend the basic methodology described here. Examples include: using information from airline data to construct network-based representations of cities across the globe—each node corresponds to a city and the weight of each edge corresponds to the number of travelers that go between the two cities as measured by air transport data;[13] representation of smaller subnetworks (aka micro-networks), using either survey data or data collected using sensors,[31,41] and use of Landscan data in conjunction with census and other sources of population information to construct resolved networks that are not as accurate but can be constructed easily for several cities as well as countries.[23,30] Note that the general approach is extensible to develop other kinds of affiliation networks, for example, individuals visiting a website or using a resource. Moreover, individual agents can be animals, devices, or digital objects. Finally, other attributes can be assigned to individuals so as to study other dynamic processes (assigning cellphones or demand for electricity). See supplementary information[44] for examples.

Step 4 is usually done in close coordination with biologists, epidemiologists, and statisticians. Computationally, this is naturally represented as a finite state probabilistic timed transition system; this is a finite state machine wherein certain transitions are timed and probabilistic. New methods in Bayesian inference as well as advances in genetics provide new opportunities for rapid inference.[20]

Step 5 involves the development of high-performance computing simulations to compute disease dynamics. This is a very active research area; the size of the networks and their unstructured and dynamic nature make for a challenging problem. Several researchers have developed scalable simulations of epidemic processes.[4,6,7,21,30,40]

Control and Resource Optimization

Step 6 involves reasoning about POMDP N given $\mathcal{G}$, $\mathcal{O}$ and $\mathcal{D}$. Several interesting combinatorial formulations have been developed for vaccination and quarantining problems for epidemics on graphs, which we describe here briefly. Consider an SIR epidemic process defined on a graph G with probability p of transmission on any edge. We model the vaccination of a subset S of nodes in graph G by deleting them. Given a distribution $\mathbf{I}_0$ of the initial infections (so that Pr[v is infected initially] = $\mathbf{I}_0(v)$), and a budget B, the vaccination problem involves choosing a subset $S \subset V$ to vaccinate with $|S| \leq B$, so that the expected number of infections (due to the initial distribution $\mathbf{I}_0$) in $G[V - S]$ is minimized. The simplest setting for this problem is when $p = 1$, which might model a highly contagious disease. Even this problem is quite difficult to approximate, and bicriteria approximations have been developed for this problem, which choose a set S of size $(1+\epsilon)B$, and ensure the number of infections is at most $(1+1/\epsilon)$ times the optimal, for any $\epsilon > 0$.[22] The vaccination problem is much better understood in the SIS model, because of the characterization in terms of the spectral structure, as mentioned previously. The problem of designing interventions in this model can be reduced to controlling the eigenvalues.[8]

These results, though theoretically very interesting, are not realistic enough to be practical; we briefly discuss some of the issues that need to be incorporated in more realistic formulations. First, the general problem of interest is when $p \in (0, 1)$ and G is an arbitrary given graph. Second, in practice, interventions are a combination of vaccination and sequestration and social distancing—the sequestration of a subset $S \subseteq V$ can be modeled by the deletion of the edges in the cut $(S, \bar{S})$, with a corresponding cost for such deletions. Finally, interventions have a temporal dimension, since they are not implemented on the first day of the outbreak, but gradually, based on its progression. Further, partial outbreak information can be used to localize the current infections, and develop more efficient interventions. Finally, the contact graph and the disease model are only partially known and are dynamic.

A common approach to capture some of the individual decision-making in response to epidemics and public health advisories for vaccination is to study game theoretical formulations.

A common approach to capture some of the individual decision-making in response to epidemics and public health advisories for vaccination is to study game theoretical formulations. We describe an early result by Aspnes et al.,[2] which has been studied quite extensively. This is formulated as a non-cooperative game, in which the strategy $x_v \in [0, 1]$ for a node v is the probability of getting vaccinated; here, we focus on pure strategies, in which $x_v \in \{0, 1\}$. For a given strategy vector $\mathbf{x}$, let G_x denote the graph obtained by deleting all nodes with $x_v = 1$. We assume an infection model of a highly contagious disease, in which all nodes reachable from an infected node get infected; further, we assume that the disease starts at a random initial node. Given a strategy vector $\mathbf{x}$, each user v has two kinds of costs: cost of getting vaccinated, $x_v C_v$, where C_v could be the economic or physical cost of the vaccine; and cost of infection, which is the probability that node v gets infected. This can be seen to be $|A(v)|^2/n$, where $A(v)$ denotes the connected component containing node v in the graph G_x, because of our infection model. It was shown this model always has a pure Nash equilibrium,[2] and the best social and Nash strategies can be approximated within $O(\log n)$ factor.

Behavioral adaption. The primary goal of an epidemiologist is to control the spread of infectious disease through the application of interventions guided by public policy. These interventions induce a behavioral change in individuals. At the same time individuals self-impose behavioral changes in response to their perception of how the disease evolves. Network-based epidemiology provides a natural representation to incorporate individual, collective, and institutional behavior (policies and responses) to control pandemics.

Verbal or conceptual behavioral models have played an important role in understanding behaviors and their relationship with diseases and maintaining a healthy life. Some of the earliest works in this direction are the Health Belief Model (HBM),[24] the Social Cognitive Theory (SCT),[3] and more recently the Social Ecological Model (SET). Verbal, social, and behavioral theories are useful in improving public health but are often informal. When using these models to develop formal computer models, these theories need to be "instantiated." This requires three components: algorithmic (mathematical) representation and computational resources as a basis of formalism, clarification of behaviors that occur at multiple scales: individual, group (for example, households, workplace), and organization (counties, companies), and expressiveness of behavioral representations from a computing standpoint. Obtaining data and conducting experiments to develop behavioral models is a challenging issue.[18,24] Recently, we developed a multitheory, multinetwork representation (see Figure 4) of individual, collective, and institutional behaviors to study the problem of anti-viral allocation during an epidemic (see supplementary information[44]).

Inference, Forecasting, and State Assessment

At the onset of an outbreak (for example, in every flu season), recurring problems for public policy planners include determining where the epidemic might have started, characteristics of the specific disease strain, and if there will be a large epidemic or pandemic. General abstractions of this problem are to determine the transmission probability or other disease characteristics, or the probability that the source of the epidemic is a node v, conditioned on observed surveillance data (and some assumptions about the prior distribution), or to find a most likelihood estimator for them. A related class of problems is to infer the underlying contact network properties, based on such observations. These are very challenging problems,[38] because surveillance data on who is infected is limited and noisy (only a fraction of the infected people are detected); and the underlying contact graph is only partially known and is dynamic. These problems are the inverse of the "forward" problems of simulating the spread of an epidemic in a social network. We briefly discuss a few important results but this largely remains an open topic.

A number of researchers[25,36,42] have studied these problems. An active area of research is based on Bayesian approaches, for example, using spatial scan statistics for detecting anomalous outbreaks;[36] these techniques rely on detecting spatial clusters from syndromic surveillance data, without using any specific properties about the epidemic process.

A rigorous result related to the inference problem is by Shah et al.,[43] who developed an efficient maximum likelihood approach for determining the source of an outbreak in trees for an SI model of disease, and then extend it to general graphs by assuming spreading in a breadth-first search order. They also developed a heuristic, called rumor centrality, which allows faster estimation of the source probability. The related problem of inferring the contact network based on the disease spread information has also been studied;[25] their results give a maximum likelihood estimator for a minimal network on which a given set of observed infections can occur.

Another important problem is to forecast the time course of an epidemic. The forecasting problem is complicated because behaviors, disease dynamics, and social networks coevolve. Researchers have studied various measures of effectiveness in this context, which include: peak value of an epidemic curve; time to reach the peak; total number of infections caused over a period (integral of the epidemic curve); the epidemic curve time series; and forecasting the onset of a pandemic. The forecasting models are then integrated with healthcare logistics to forecast required health care resources (for example, the number of beds, face masks, ventilators, and so on). The models are also integrated with pharmaceutical supply chains to develop a plan for production and distribution of vaccinations and anti-virals (for example, how many doses of vaccines to produce, availability, and distribution).[44]

Computational Epidemiology Over Networks

Step 1. Construct a synthetic yet realistic population by integrating a variety of commercial and public sources.

Step 2. Build a set of detailed activity templates for households using time-use surveys and digital traces.[5,21]

Step 3. Construct a dynamic social bipartite visitation network, G_{PL}, which encodes the locations visited by each person.

Step 4. Develop models of within-host disease progression using detailed case-based data and serological samples to establish disease parameters.

Step 5. Develop high-performance computer simulations to study epidemic dynamics (exploring the Markov chain M).

Step 6. Develop multitheory multinetwork models of individual, collective, and organizational behaviors, formulating and evaluating the efficacy of various intervention strategies and methods for situation assessment and epidemic forecasting.

Recent Advances

The role of computing goes beyond the topics we discussed so far. Recent advances in computing create exciting new opportunities for combining computational thinking and traditional epidemiology. Here, we discuss three salient topics:

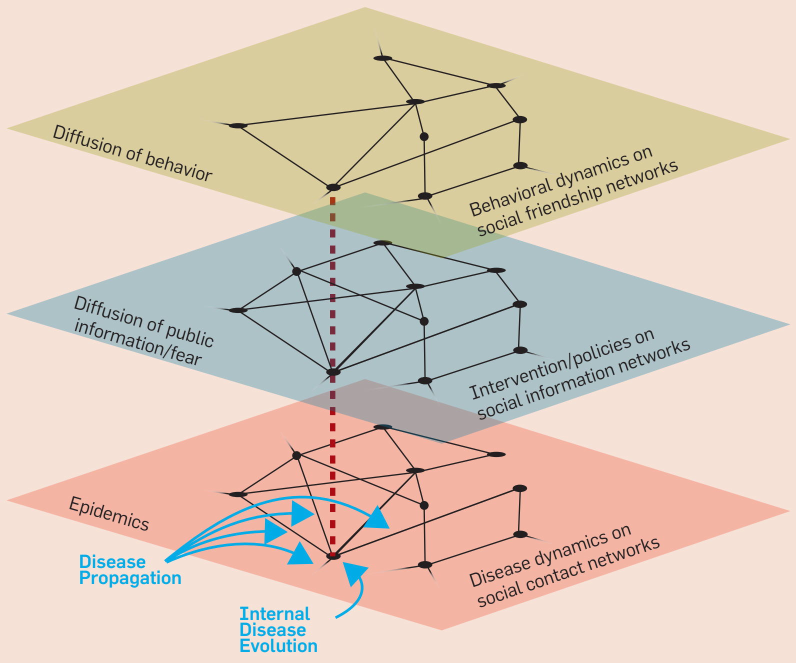
Synthetic information and decision analytics. The epidemiological modeling tools provide detailed information on spatiotemporal disease dynamics. The data produced by these simulations as well as the data collected from various surveillance sources to initialize and calibrate these models poses challenging data analytics questions. Methods are required to analyze the outputs of the causal models to discover new transmission chains, influential and critical nodes, optimal intervention pathways, among others.[2,21,22,25,36,43] Furthermore, these methods must be made accessible to epidemiologists and policymakers via a user-friendly environment that provides an easy way to set up experiments and analyze the results. Recently, a number of visual and data analytics methods and environments have been developed to support epidemiological research.[10,15,29] For example, a tool called the Interface to Synthetic Information Systems (ISIS) developed by our laboratory allows a user to set up detailed factorial experiments.[15,44] Using a simple interface to an underlying digital library, a user can choose from among many preconstructed instances: a social contact network; a within-host disease progression model; and a set of interventions. A key aspect of ISIS is its simplicity; we can train public health analysts to use the system in about three hours.[44]

Pervasive cyber-environment for computational epidemiology. Researchers have started developing a pervasive computing environment to support public health epidemiology.[10,15] Our group has also taken a step in this direction and developed a preliminary cyber-environment called Cyber Infrastructure for EPIdemics (CIEPI), which aims to make HPC resources seamless, invisible, and indispensable in routine analytical efforts. It is also designed to collect and process real-time data available via sensors and the Web. Indemics synthetic network construction methods and ISIS are integral components of CIEPI.

Big data and computational epidemiology. Recent advances in social media, computational turks, online games, online surveys, and digital traces all form the basis of potentially exciting new methods that can support surveillance, forecasting and tracking behavioral adaptation. For example, in the case of forecasting, the basic underlying hypothesis is that there is a strong correlation between incidence of diseases and the propensity of individuals to search for specific disease- related information or talk about it online.[11,19,42] Big data will play an increasingly important role in supporting computational epidemiology. Social media data, online micro-surveys

Figure 4. Multinetwork representations and coevolution of behavioral models.

The layers represent examples of different kinds of networks in which the nodes might be involved. The bottom layer is a social contact network formed by colocation constraints, on which diseases spread. The middle layer is an information network, on which information/fear spread. Finally, the top layer is a friendship network that spreads influence, for example, peer pressure. (Image courtesy of Henning Mortveit.)


Diffusion of behavior
Behavioral dynamics on social friendship networks
Diffusion of public information/fear
Intervention/policies on social information networks
Epidemics
Disease Propagation
Disease dynamics on social contact networks
Internal Disease Evolution

and games, computationally mediated labor markets (for example, Mechanical Turks), improved bio-sensors, serological and genomic data will all lead to improved state assessment. These technologies pose new challenges, including: methods for eliciting truthful behavior; overcoming biases in data due to the demographics of participants; and translating behaviors from the virtual world to the real world.[42]

Policy Informatics

As discussed in Longini[26] and Lipsitch,[28] during the early stages of a pandemic, local, national, and global public health authorities need to make important decisions on allocating resources for tracking and controlling a potential pandemic beyond those required for routine disease monitoring. Allocation of these resources depends on two factors: estimating the risk of severity of the pathogen causing the infections, and the risk that the outbreak will reach a given region and cause widespread, serious illness. Unfortunately, it is extremely difficult to accurately estimate all of these risks, especially in the early stages of the pandemic. Hence, the role of surveillance as well as agile modeling and decision support environments to support analysis of counterfactual scenarios becomes all the more important. Appropriate response requires a delicate balance between slow and inadequate action on one hand and overreaction on the other hand. Additionally, a close collaboration with experts and decision makers who are often separated by institutional and governmental boundaries is important.

The computational tools we have built have been used to support a number of stakeholder-defined case studies. For example, as a part of the NIH-funded Models of Infectious Disease Agent Study (MIDAS)[35] network, we carried out a computational analysis of targeted layered containment strategy to control influenza pandemic. Results of the study were reviewed in a Letter Report by the Institute of Medicine.[32] We recently used CIEPI and associated tools to support near-real time decisions during the 2009 H1N1 pandemic. As H1N1 influenza continued to spread in the U.S., the Department of Health and Human Services teamed up with the Defense Threat Reduction Agency to place the ISIS tool in the hands of U.S. government analysts to provide day-to-day modeling results. Providing analysis inside the 24-hour decision cycle to this emerging crisis would not have been possible without the development of highly optimized modeling software as well as the Web-enabled interface. The analysts were able to perform course-of-action analyses to estimate the impact of closing schools and shutting down workplaces. Better situational awareness was enabled by calibrating the model to real-world data with different disease characteristics to estimate likely pathogen behaviors.

Summary

We outlined how recent advances in computing and information combined with computational thinking can be effectively employed to support public health epidemiology as it pertains to communicable diseases. Epidemiology of noncommunicable diseases (for example, obesity, diabetes) as well as environmental epidemiology (such as diseases caused by industrial toxic waste) are important but not discussed in this article. An important topic that we did not discuss is validation; see supplementary information[44] for details. Another topic not discussed is disease evolution and coevolving multistrain diseases. The ideas are also applicable in two related areas: zoonotic diseases and vector-borne diseases.[33,34]

We conclude by briefly discussing how concepts and ideas in epidemiology over the last 50 years have found applications in a large number of seemingly unrelated areas. The SIR model corresponds to the graph reliability problem, in which the goal is to determine the probability that a subset $S \subset V$ in a graph $G = (V, E)$ is connected, if each edge $e \in E$ fails independently with probability $p(e)$. The general ideas of epidemic diffusion have been used to design local and distributed algorithms for a variety of problems, including: routing, aggregation, broadcasting, and spatial gossip in communication networks (especially sensor networks),[17] synchronization, information sharing, and load balancing in parallel computing and replicated databases.[14] A broader class of reaction-diffusion models, which generalize SIR/SIS disease models, have been studied in diverse areas, including economics, sociology, viral marketing, political science, and social networks.[16]

Acknowledgments

The article represents the collective work of members of the Network Dynamics and Simulation Science Laboratory (NDSSL) and our external collaborators. We thank the reviewers for their feedback and suggestions. We thank Moshe Vardi for inviting us to submit the article and for his patience and willingness to work with us, without which the article would never have been completed.

The work is partially supported by NIH MIDAS, DTRA CNIMS, DOE ASCR, NSF ICES, NSF NetSe, NSF SDCI and IARPA programs.

References

1. Alon, N., Benjamini, I. and Stacey, A. Percolation on finite graphs and isoperimetric inequalities. *Annals of Probability* (2004).
2. Aspnes, J., Chang, K.L. and Yampolskiy, A. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *J. Comput. Syst. Sci.* (2006).
3. Bandura, A. *Social Foundations of Thought and Action: A Social Cognitive Theory.* Prentice Hall. 1986.
4. Barrett, C. et al. Episimdemics: An efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In *Proc. of the ACM/IEEE Conference on High Performance Computing* (2008).
5. Barrett, C. et al. Generation and analysis of large synthetic social contact networks. In *Proc. Winter Simulation Conference* (2009), 1003–1014.
6. Bisset, K. et al. Indemics: An interactive data intensive framework for high performance epidemic simulation. In *Proc. Int. Conf. Supercomputing* (2010).
7. Bisset, K. et al. Interaction-based HPC modeling of social, biological, and economic contagions over large networks. In *Proc. of Winter Simulation Conference* (2011), 2933–2947.
8. Borgs, C., Chayes, J.T., Ganesh, A. and Saberi, A. How to distribute antidotes to control epidemics. *Random Structures and Algorithms* (2010).
9. Brauer, F., van den Driessche, P. and Wu, J. editors. Mathematical Epidemiology. *Lecture Notes in Mathematics.* Springer Verlag, 1945.
10. Broeck, W. et al. The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. *BMC Infectious Diseases 11*, 1:37 (2011).
11. Brownstein, J., Freifeld, C. and Madoff, L. Digital disease detection—harnessing the Web for public health surveillance. *N. England J. Med.* (2009), 2153–2157.
12. Chung, F. and Lu, L. Connected components in random graphs with given degree sequences. *Annals of Combinatorics 6* (2002), 125–145.
13. Colizza, V., Barrat, A., Barthelemy, M. and Vespignani, A. The role of the airline transportation network in the prediction and predictability of global epidemics. *PNAS 103*, 7 (2006), 2015–2020.
14. Demers, A. et al. Epidemic algorithms for replicated database maintenance. In *Proceedings for ACM PODC* (1987), 1–12.
15. Deodhar, S. et al. Enhancing user-productivity and capability through integration of distinct software in epidemiological systems. *IHI* (2012), 171–180.
16. Easley, D. and Kleinberg, J. *Networks, Crowds and Markets: Reasoning About A Highly Connected World.* Cambridge University Press, New York, 2010.
17. Epidemic Routing Bibliography. http://roland.grc.nasa.gov/~weddy/biblio/epidemic/.
18. Cowling, B. et al. Community psychological and behavioral responses through the first wave of the 2009 influenza a (H1N1) pandemic in Hong Kong. *J. Infect. Diseases 202* (2010), 867–877.
19. Ginsberg, J. et al. Detecting influenza epidemics using search engine query data. *Nature* (2008), 1012–1014.
20. Cauchemez, S. et al. Household transmission of 2009 pandemic influenza a (H1N1) virus in the United States. *New England J. Medicine 27* (2009), 2619–2627.
21. Eubank, S. et al. Modeling disease outbreaks in realistic urban social networks. *Nature 429* (2004), 180–184.
22. Eubank, S. et al. Structure of social networks and their impact on epidemics. *DIMACS Discrete Methods in Epidemiology* (2006), 179–185.
23. Ferguson, N.M. et al. Strategies for containing an emerging influenza pandemic in Southeast Asia. *Nature 437* (2005), 209–214.
24. Funk, S., Salathé, M. and Jansen, V. Modeling the influence of human behaviour on the spread of infectious diseases: A review. *J. R. Soc. Interface 7* (2010), 1247–1256.
25. Gomez-Rodriguez, M., Leskovec, J. and Krause, A. Inferring networks of diffusion and influence. In *Proc. 16th ACM KDD*, 2010.
26. Van Kerkhove, M. and Ferguson, N. Epidemic and intervention modeling—A scientific rationale for policy decisions: Lessons from the 2009 influenza pandemic. *Bull World Health Organ.* (2012), 306–310.
27. Lipsitch, M. and et al. Managing and reducing uncertainty in an emerging influenza pandemic. *New England Journal of Medicine 361*, 2 (2009), 112–115.
28. Lipsitch, M. et al. Improving the evidence base for decision making during a pandemic: The example of 2009 influenza-A H1N1. *Biosecur Bioterror.* (2011).
29. Livnat, Y., Rhyne, T. and Samore, M. Epinome: A visual-analytics workbench for epidemiology data. *IEEE Comp. Graphics & App. 32*, 2 (2012), 89–95.
30. Longini. I.M. et al. Containing pandemic influenza at the source. *Science 309* (2005), 1083–1087.
31. Mossong, J. et al. Social contacts and mixing patterns relevant to the spread of infectious diseases. PLoS Med 5, 3 (2008) e74.
32. National Academies. Modeling Community Containment for Pandemic Influenza: A Letter Report (2006).
33. National Academies. The Causes and Impacts of Neglected Tropical and Zoonotic Diseases: Opportunities for Integrated Intervention Strategies (2011).
34. National Academies. Vector-Borne diseases: Understanding the Environmental, Human Health, and Ecological Connections (2011).
35. National Institutes of Health, 2009; http://www.nigms.nih.gov/Initiatives/MIDAS/.
36. Neil, D., Moore, A. and Cooper, G. A Bayesian spatial scan statistic. NIPS (2005).
37. Newman, M. The structure and function of complex networks. *SIAM Review 45* (2003).
38. Nishiura, H. et al. Did modeling overestimate the transmission potential of pandemic (H1N1-2009)? Sample size estimation for post-epidemic seroepidemiological studies. PLoS ONE 3 (2011).
39. Pastor-Satorras, R. and Vespignani, A. Epidemic spreading in scale-free networks. *Phys. Rev. Lett. 86* (Apr. 2001). 3200–3203.
40. Perumalla, K. and Seal, S. Discrete event modeling and massively parallel execution of epidemic outbreak phenomena. *Simulation*, 2011.
41. Salathé, M. et al. A high-resolution human contact network for infectious disease transmission. *PNAS 107*, 51 (2010), 22020–22025.
42. Salathé, M. et al. Digital epidemiology. *PLoS Computational Biology 8*, 7 (2012).
43. Shah, D. and Zaman, T. Rumors in a network: Who is the culprit? *ACM SIGMETRICS* (2010).
44. Supplementary information; http://ndssl.vbi.vt.edu/supplementary-info/vskumar/cacm2012/.

Madhav Marathe (mmarathe@vbi.vt.edu) is a professor in the Department of Computer Science and Network Dynamics and Simulation Laboratory, Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, VA.

Anil Kumar S. Vullikanti (akumar@vbi.vt.edu) is an associate professor in the Department of Computer Science and Network Dynamics and Simulation Laboratory, Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, VA.

ACM ANNOUNCES NEW CHANGES TO EXPAND ACCESS TO PUBLICATIONS

ACM is pleased to announce important changes to our publications policy aimed at increasing exposure to and the impact of our publications on the global computing community. These changes further empower ACM authors to more widely distribute and make available their work, while at the same time taking advantage of the numerous benefits and prestige of publishing with ACM.

Some highlights of the new policy:

- ACM authors now have the option of retaining copyright and other important ownership rights by selecting one of ACM's "*Copyright Transfer*", "*Exclusive License to Publish*", or "*Non-Exclusive Permission to Publish*" Agreements
- ACM authors now have the option of making their work perpetually free to the world via the ACM DL platform by participating in the new ACM Open Access publication program.
- ACM Special Interest Groups (SIGs) now have the option of making their ACM conference proceedings freely available to the world on a limited basis via the ACM Digital Library, SIG, or conference websites.

For detailed information about these new options, as well as information about ACM's complete publications policy, please visit http://authors.acm.org.

acm Author Rights

CHOOSE	your rights option
POST	on your own websites
DISTRIBUTE	via *ACM Author-Izer*
REUSE	your own work
CREATE	derivative works
RETAIN	perpetual rights

acm Association for Computing Machinery

DOI:10.1145/2483852.2483872

Technical Perspective A Fresh Approach to Vector Graphics

By Peter Wonka

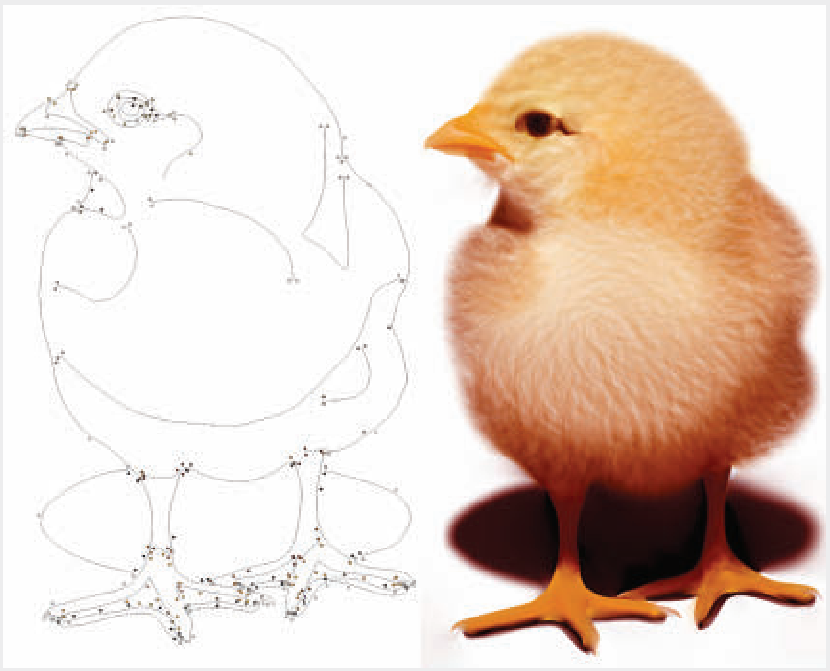
TO UNDERSTAND THE impact of the following paper, I would like to first reflect on the following question: What is a good representation for two-dimensional images?

There are generally two distinct choices to represent and process an image: raster graphics and vector graphics. In raster graphics, color values are stored as discrete samples on a regular pixel grid. The major advantage of this representation is its simplicity and raster images are the most common image representation today. This is mainly due to the fact that photographs are naturally stored as raster images.

Still, as an important alternative, vector graphics provides many unique advantages. In vector graphics, images are stored using curves, typically placed at image edges. Traditionally, these curves have been closed curves, so that each curve encloses a clearly defined region. The curves are given some attributes that describe how the enclosed region should be filled, for example, in the simplest case using a single color. One advantage of vector graphics is that it provides infinitely sharp edges. If we wish to zoom into an area of the image, or print a very high-resolution version of the image, the edges will remain sharp. Therefore, vector graphics is considered to be resolution independent in a certain sense. Additionally, some editing operations and animations are much easier with a vector graphics representation.

The following paper makes a fundamental contribution to the world of vector graphics by introducing a new primitive called diffusion curve. Diffusion curves no longer have to be closed, but can be open curves. To determine the colors in other parts of the image, the authors propose to use a diffusion equation. The figure here offers an example of a diffusion curve image generated by my colleague, Stefan Jeschke. The fact that most of the curves are open has some implications for editing this image. While in many areas of this image we can still imagine a similar representation with closed curves this creates some overhead. Closed curves require the user to manage the curves on different layers to define occlusions and to manage the topology. Using open curves it is easier to draw like using pen and paper.

Diffusion curves example: Curves with texture samples (left) and rendering (right).



Personally, I believe the importance of this paper is mainly due to two reasons. First, it is very difficult to make a fundamental contribution to a mature and important area of research, such as vector graphics. An essential aspect of this contribution is it increases the expressiveness of vector graphics. It becomes easier to represent and edit images using curves. Diffusion curves are a versatile representation and they make for a great addition to any vector graphics software.

Second, this is a seminal paper that poses many exciting questions for future research. How can diffusion curves be used in computer animation? Can the idea of diffusion curves be extended to represent textures? How can three-dimensional textures, smoke, or surfaces be defined by diffusion curves? How to accelerate the rendering of diffusion curves? How to extend the definition of diffusion curves to have more control over how colors are diffused? How to further automate the conversion of raster images into a diffusion curve representation?

Many of these questions have been addressed in subsequent papers, demonstrating how a paper like this, with a fresh approach to an old representation, can breath new life into a mature area like vector graphics. C

Peter Wonka is an associate professor in computer science at Arizona State University, Tempe, and at KAUST, Thuwal, Saudi Arabia.

DOI:10.1145/2483852.2483873

Diffusion Curves: A Vector Representation for Smooth-Shaded Images

By Alexandrina Orzan, Adrien Bousseau, Pascal Barla, Holger Winnemöller, Joëlle Thollot, and David Salesin

Abstract

We describe a new vector-based primitive for creating smooth-shaded images, called the *diffusion curve*. A diffusion curve partitions the space through which it is drawn, defining different colors on either side. These colors may vary smoothly along the curve. In addition, the sharpness of the color transition from one side of the curve to the other can be controlled. Given a set of diffusion curves, the final image is constructed by solving a Poisson equation whose constraints are specified by the set of gradients across all diffusion curves. Like all vector-based primitives, diffusion curves conveniently support a variety of operations, including geometry-based editing, keyframe animation, and ready stylization. Moreover, their representation is compact and inherently resolution independent. We describe a GPU-based implementation for rendering images defined by a set of diffusion curves in real time. We then demonstrate an interactive drawing system for allowing artists to create artworks using diffusion curves, either by drawing the curves in a freehand style, or by tracing existing imagery. Furthermore, we describe a completely automatic conversion process for taking an image and turning it into a set of diffusion curves that closely approximate the original image content.

1. INTRODUCTION

Vector graphics, in which images are defined with geometric primitives such as lines, curves, and polygons, dates back to the earliest days of computer graphics. Early cathode-ray tubes, starting with the Whirlwind-I in 1950, were all vector displays, while the seminal Sketchpad system[24] allowed users to manipulate geometric shapes in the first computer-aided design tool. *Raster graphics* provides an alternative representation for describing images via a grid of pixels rather than geometry. Raster graphics arose with the advent of the framebuffer in the 1970s and is now commonly used for storing, displaying, and editing images. However, while raster graphics offers some advantages over vector graphics—primarily, a more direct mapping between the hardware devices used for acquisition and display of images, and their internal representation—vector graphics continues to provide certain benefits as well. Most notably, vector graphics offers a more compact representation, geometric editability, and resolution independence (allowing scaling of images while retaining sharp edges, see Figure 2). Vector-based images are also more easily animated through keyframe animation of their underlying geometric primitives. For all of these reasons, vector-based drawing tools, such as Adobe Illustrator©, CorelDraw©, and Inkscape©, continue to enjoy great popularity, as do standardized vector representations such as Flash and SVG.

However, for all of their benefits, vector-based drawing tools offer only limited support for representing complex color variations. For example, the shading of the turtle in Figure 2 is only depicted with uniform color regions, and most existing vector formats support only linear or radial gradients. A more sophisticated vector-based tool for handling complex gradients is the *gradient mesh*. A gradient mesh is a lattice with colors at each vertex that are interpolated across the mesh. While more powerful than simple gradients (or "ramps"), gradient meshes still suffer from some limitations. A significant one is that the topological constraints imposed by the mesh lattice give rise to an overcomplete representation that becomes increasingly inefficient and difficult to create and manipulate due to the high number of control points. Although automatic generation techniques[15, 23, 25] have been recently proposed, they do not address the process of subsequent gradient-mesh manipulation, nor do they facilitate free-hand creation of gradient meshes from scratch.

In this paper we propose an alternative vector-graphics primitive, called the *diffusion curve*. A diffusion curve is a curve that diffuses colors on both sides of the space that it divides. The motivations behind such a representation are twofold:

First, this representation supports traditional freehand drawing techniques where artists begin by sketching shapes, then adding color later. Similarly, users of our tool first lay down drawing curves corresponding to color boundaries. In contrast with traditional vector graphics, the color boundaries do not need to form closed geometric shapes and may even intersect. For example, the cloth in Figure 1 is created with open curves that have different colors on each side. The diffusion process then creates smooth gradients between nearby curves. Closed curves can be used to define regions of constant colors, such as the eyebrows in Figure 1. By specifying blur values along a curve, artists can also create smooth color transitions across the curve boundaries, like on the cheek of the character in Figure 1.

Second, most color variations in an image can be assumed to be caused by edges (material and depth discontinuities,

A previous version of this paper was published in *Transactions on Graphics* (*Proceedings of SIGGRAPH 2008*), 27, 2008.

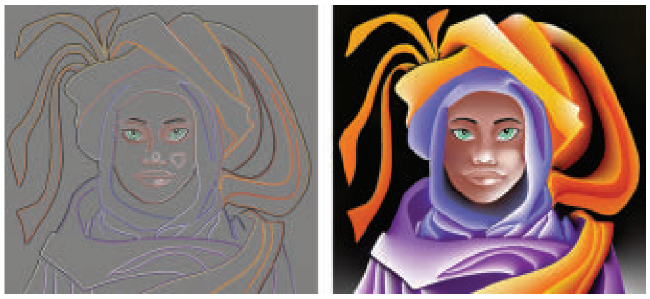


Figure 1. Diffusion curves (left), and the corresponding color image (right). Note the complex shading on the folds and blur on the face.

texture edges, shadow borders, etc.).[14, 18] Even subtle shading effects can be modeled as though caused by one or more edges, and it has been demonstrated that edges can be used to encode and edit images.[6, 8, 9] In this work, we rely on vision algorithms, such as edge detection, to convert an image into our diffusion curves representation fully automatically.

The main contribution of this paper is therefore the definition of diffusion curves as a fundamental vector primitive, along with two types of tools: (1) A prototype allowing manual creation and editing of diffusion curves. Thanks to an efficient GPU implementation, the artist benefits from instant visual feedback despite the heavy computational demands of a global color diffusion. (2) A fully automatic conversion from a bitmap image based on scale-space analysis. The resulting diffusion curves faithfully represent the original image and can then be edited manually.

2. PREVIOUS WORK

We review here existing techniques to create complex color gradients with modern vector graphics tools. We also mention methods to convert photographs in vector images with these techniques.

For a long time, vector graphics has been limited to primitives (paths, polygons) filled with uniform color or linear and radial gradients. Although skillful artists can create rich vector art with these simple tools, the resulting images often present flat or limited shading. The SVG format and modern drawing tools (Adobe Illustrator©, Corel CorelDraw©, Inkscape©) allow the creation of smooth color variations by blurring the vector primitives after rasterization. Nevertheless, artists often need to combine multiple primitives to create complex shading effects. Gradient meshes have been introduced in Adobe Illustrator© and Corel CorelDraw© to address these limitations. This tool produces smooth color variations over the faces of a quad mesh by interpolating the color values stored at its vertices. However, creating a mesh requires much skill and patience, because the artist needs to anticipate the mesh resolution and orientation necessary to embed the desired image content. This is why most artists rely on an example bitmap to drive the design of realistic gradient meshes. In addition, because the mesh resolution needs to be high enough to capture the finest image features, it is often unnecessarily high for nearby smoothly varying regions that could be represented with much fewer information. This constraint makes gradient meshes tedious to manipulate.

An alternative to manual creation is to convert an existing bitmap image into a vector representation. Commercial tools such as Adobe Live Trace© convert bitmap images into vector graphics by segmenting the image into regions of constant color and fitting polygons onto these regions. The resulting images often have a cartoon style similar to Figure 2 due to color quantization. The ArDeco system of Lecot and Levy[16] produces smoother images by approximating color variations inside regions with linear or radial gradients. Similarly, several systems have been proposed to approximate a bitmap image with a collection of gradient meshes, either with user assistance[23] or automatically.[25]

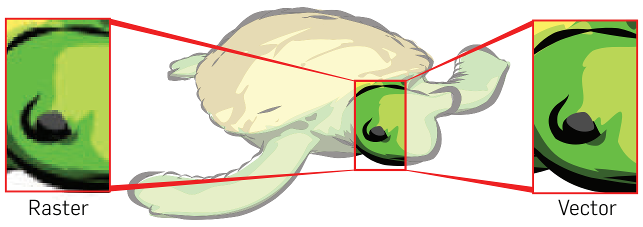
The new representation described in this paper offers the same level of visual complexity as that reached by gradient meshes, but has two main advantages: it is *sparse*, and corresponds to *meaningful* image features. Diffusion curves do not impose connectivity and do not require superfluous subdivision, which facilitates the manipulation of color gradients. Moreover, our representation naturally lends itself to automatic extraction from a bitmap image: we use an edge detector to locate curves and image analysis algorithms to estimate curve attributes (color and blur).

In other words, compared to regions used in classic vector representations, or patches used in gradient meshes, our approach is motivated by the fact that most of the color variation in an image is caused by or can be modeled with edges; and that (possibly open) regions or patches are implicitly defined in between. Such a sparse image representation is strongly motivated by the work of Elder[8], who demonstrated that edges can faithfully encode images. Elder and Goldberg[9] also suggested the possibility of using edges to manipulate images with basic operations (edge delete, copy, and paste). However, we believe the full potential of this approach has yet to be attained. For this reason, our conversion algorithm starts from the same premises as Elder's system. By vectorizing edges and their attributes, we extend manipulation capabilities to include shape, color, and blur operations and we support resolution independence and keyframe animation.

3. DIFFUSION CURVES

In this section we introduce the basic primitive of our representation, called a *diffusion curve*, and describe how to efficiently render an image from such primitives. Creation and manipulation of diffusion curves are discussed in subsequent sections.

Figure 2. Raster graphics have a finite resolution, producing visual artifacts when magnified too much. Vector graphics can be magnified artifact-free.



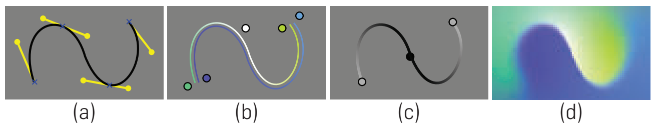
3.1. Data structure

The basic element of a diffusion curve is a geometric curve defined as a cubic Bézier spline (Figure 3(a)) specified by a set of control points P. The geometry is augmented with additional attributes: two sets of color control points C_l and C_r (Figure 3(b)), corresponding to color constraints on the *right* and *left* half space of the curve; and a set of *blur* control points (Σ) that defines the smoothness of the transition between the two halves (Figure 3(c)). As a result, the curves diffuse color on each side with a soft transition across the curve given by its blur (Figure 3(d)).

Color and blur attributes can vary along a curve to create rich color transitions. This variation is guided by an interpolation between the attribute control points in attribute space. In practice, we use linear interpolation and consider colors in RGB space throughout the rendering process, because they map more easily onto an efficient GPU implementation and proved to be sufficient for the artists using our system. Control points for geometry and attributes are stored independently, since they are generally uncorrelated. This leads to four independent arrays in which the control points (geometry and attribute values) are stored together with their respective parametric position t along the curve:

$$\textbf{Diffusion Curve:}\left|\begin{array}{l} \mathrm{P}\left[n_{pos}\right]: \text{array of } (x, y, \text{ tangent});\\ C_l\left[n_l\right]: \text{array of } (r,g,b,t);\\ C_r\left[n_r\right]: \text{ array of } (r,g,b,t);\\ \Sigma\left[n_\sigma\right]: \text{array of } (\sigma,t); \end{array}\right.$$

Figure 3. A Diffusion curve is composed of (a) a geometric curve described by a Bézier spline, (b) arbitrary colors on either side, linearly interpolated along the curve, (c) a blur amount linearly interpolated along the curve. The final image (d) is obtained by diffusion and reblurring. Note the complex color distribution and blur variation defined with a handful of controls.



The diffusion curves structure encodes data similar to Elder's edge-based representation.[8] However, the vectorial nature of a diffusion curve expands the capabilities of Elder's discrete edges by allowing precise control over both shapes—via manipulation of control points and tangents—and appearance attributes—via color and blur control points (small circles on the Figures). This fine-level control, along with our real time rendering procedure, facilitates the drawing and editing of smooth-shaded images.

3.2. Rendering smooth gradients from diffusion curves

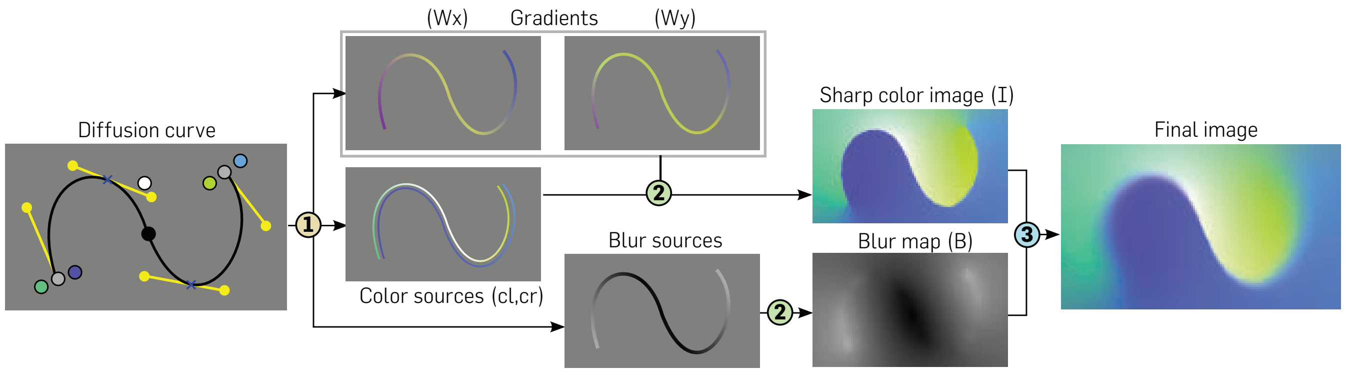
Three main steps are involved in our rendering model (see Figure 4): (1) rasterize a *color sources* image, where color constraints are represented as colored curves on both sides of each Bézier spline, and the rest of the pixels are uncolored; (2) *diffuse* the color sources similarly to heat diffusion—an iterative process that spreads the colors over the image; we implement the diffusion on the GPU to maintain real time performance; and (3) *reblur* the resulting image with a spatially varying blur guided by the blur attributes.

Color sources. Using the interpolated color values, the first step renders the left and right color sources $cl(t)$, $cr(t)$ for every pixel along the curves. An alpha mask is computed along with the rendering to indicate the exact location of color sources versus undefined areas.

For perfectly sharp curves, these color sources are theoretically infinitely close to each other. However, rasterizing points separated by too small a distance on a discrete pixel grid leads to overlapping pixels. In our case, this means that several color sources are drawn at the same location, creating visual artifacts after the diffusion. Our solution is to distance the color sources from the curve slightly, and to add a color gradient constraint directly on the curve. The gradient maintains the sharp color transition, while the colors, placed at a small distance d in the direction normal to the curve, remain separate.

More precisely, the gradient constraint is expressed as a gradient field **w** which is zero everywhere except on the curve, where it is equal to the color derivative across the curve. We decompose the gradient field in a gradient along the x direction

Figure 4. Rendering diffusion curves requires (1) the rasterization of the color and blur sources, along with the gradient field w = (w$_x$, w$_y$), (2) the diffusion of colors and blur, and (3) the reblurring of the color image.



$\mathbf{w}_x$ and a gradient along the y direction $\mathbf{w}_y$. For each pixel on the curve, we compute the color derivative across the curve from the curve normal N and the left (cl) and right (cr) colors as follow (we omit the t parameter for clarity): $w_{x,y} = (cl - cr)N_{x,y}$.

We rasterize the color and gradient constraints in 3 RGB images: an image C containing colored pixels on each side of the curves, and two images W_x, W_y containing the gradient field components. In practice, the gradient field is rasterized along the curves using lines of one pixel width. Color sources are rasterized using triangle strips of width $2d$ using a pixel shader that only draws pixels that are at a distance d from the curve (Figure 4(1)). In our implementation d is set at 3 pixels. Pixel overlap can still occur along a curve in regions of high curvature (where the triangle strip overlaps itself) or when two curves are too close to each other (as with thin structures or intersections). A simple stencil test allows us to discard overlapping color sources before they are drawn, which implies that solely the gradient field $\mathbf{w}$ dictates the color transitions in these areas. An example of such case can be seen in Figure 1, where the eyebrows are accurately rendered despite their thin geometry.

Diffusion. Given the color sources and gradient fields computed in the previous step, we next compute the color image I resulting from the steady state diffusion of the color sources subject to the gradient constraints (Figure 4(2)). Similar to previous methods,[6, 8] we express this diffusion as the solution to a Poisson equation, where the color sources impose local constraints:

$$\Delta I = \operatorname{div} \mathbf{w},$$
$$I(x, y) = C(x, y) \text{ if pixel } (x, y) \text{ stores a color value,}$$

where Δ and div are the Laplace and divergence operators.

Computing the Poisson solution requires solving a large sparse linear system, which can be very time consuming if implemented naively. To offer interactive feedback to the artist, we solve the equation with a GPU implementation of the multigrid algorithm.[4, 19] The idea behind multigrid methods is to use a coarse version of the domain to efficiently solve for the low frequency components of the solution, and a fine version of the domain to refine the high frequency components. We use Jacobi relaxations to solve for each level of the multigrid, and limit the number of relaxation iterations to achieve real time performance. Typically, for a 512×512 image we use $5i$ Jacobi iterations per multigrid level, with i the level number from fine to coarse. This number of iterations can then be increased when high quality is required. All the images in this paper have been rendered using an Nvidia GeForce 8800, providing real time performance on a 512×512 grid with a reasonable number of curves (several thousands). Jeschke et al.[11] describe a GPU solver dedicated to diffusion curves with improved performance.

Reblurring. The last step of our rendering pipeline takes as input the color image containing sharp edges, produced by the color diffusion, and reblurs it according to blur values stored along each curve. However, because the blur values are defined only along curves, we lack blur values for off-curve pixels. A simple solution, proposed by Elder,[8] diffuses the blur values over the image similarly to the color diffusion described previously. We adopt the same strategy and use our multi-grid implementation to create a blur map B from the blur values. The only difference to the color diffusion process is that blur values are located exactly on the curve so we do not require any gradient constraints. This leads to the following equation:

$$\Delta B = 0,$$
$$B(x, y) = \sigma(x, y) \text{ if pixel } (x, y) \text{ is on a curve.}$$

Given the resulting blur map B, we apply a spatially varying blur on the color image (Figure 4(3)), where the size of the blur kernel at each pixel is defined by the required amount of blur for this pixel. Despite a spatially varying blur routine implemented on the GPU,[1] this step is still computationally expensive for large blur kernels (around one second per frame in our implementation), so we bypass it during curve drawing and manipulations and reactivate it once the drawing interaction is complete.

Panning and zooming. Solving a Poisson equation leads to a global solution, which means that any curve can influence any pixel of the final image. This global solution raises an issue when zooming into a sub-part of an image, because curves outside the current viewport should still influence the viewport's content. To address this problem without requiring a full Poisson solution at a higher resolution, we first compute a low-resolution diffusion on the unzoomed image domain, and use the obtained solution to define Dirichlet boundary conditions around the zooming window. This gives us a sufficiently good approximation to compute a full-resolution diffusion only within the viewport.

4. CREATING DIFFUSION CURVES

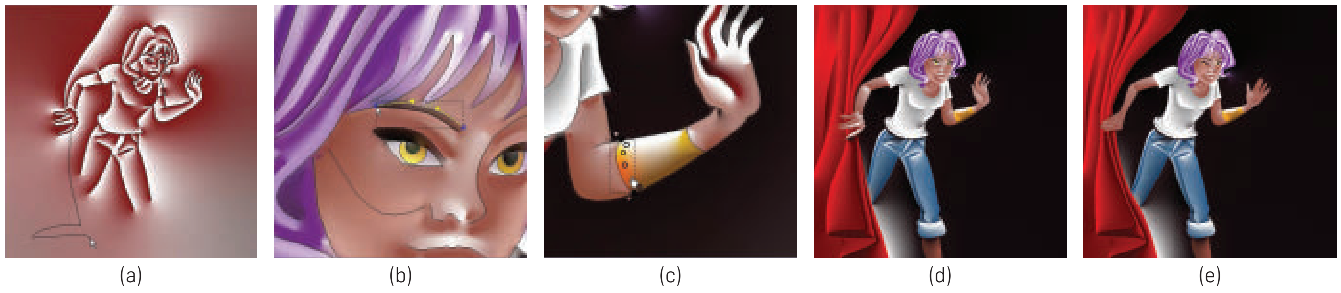
The process of creating an image varies among artists. One might start with an empty canvas and sketch freehand strokes while another may prefer to use an existing image as a reference. We provide the user with both options to create diffusion curves. For *manual* creation, the artist can create an image with our tool by sketching the lines of the drawing and then filling in the color. When using an image as a template, we propose two methods. *Assisted:* The artist can trace manually over parts of an image and we recover the colors of the underlying content. *Automatic:* the artist can automatically convert an image into our representation and possibly post-edit it.

4.1. Manual creation

To facilitate content creation for the artist, we offer several standard tools: editing of curve geometry, curve splitting, copy/paste, zooming, color picking, etc. We also developed specific tools: copy/paste of color and blur attributes from one curve to another, editing of attribute control points (add, delete, and modify), etc. To illustrate how an artist can use our diffusion curves, we show in Figure 5 (and accompanying video[1]) the different stages of an image being drawn

[1] http://cacm.acm.org

Figure 5. Example steps for manual creation : (a) sketching the curves, (b) adjusting geometry, (c) setting color and blur, (d) final result, and (e) keyframe animation. The image (d) was created in 4 hours by an artist at first contact with the tool.



with our tool. The artist employs the same process as in traditional drawing: a sketch followed by color filling.

4.2. Tracing an image

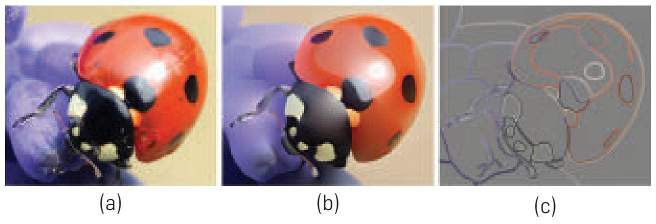
In many situations an artist will not create an artwork entirely from scratch, but instead use existing images for guidance. For this, we offer the possibility of extracting the colors of an underlying bitmap along a drawn curve. This process is illustrated in Figure 6.

The challenge here is to correctly extract and vectorize colors on each side of a curve. We also need to consider that color outliers might occur due to noise in the underlying bitmap or because the curve positioning was suboptimal. We first uniformly sample the colors along the curve at a distance *d* (same as the one used for rendering) in the direction of the curve's normal. We then identify color outliers by measuring a standard deviation in a neighborhood of the current sample along the curve. To this end, we work in CIE L*a*b* color space (considered perceptually uniform for just-noticeable-differences), and tag a color sample as an outlier if it deviates too much from the mean in either the L*, a,* or b* channel. We convert colors to RGB at the end of the vectorization process for compatibility with our rendering system. We finally fit a polyline to the color samples using the Douglas–Peucker algorithm.[7] This iterative procedure starts with a line connecting the first and last sample and repeatedly subdivides the line into smaller and smaller segments until the maximum distance (still in CIE L*a*b*) between the actual values and the current polyline is smaller than an error tolerance ϵ. The vertices of the final polyline yield the color control points that we attach to the curve. Jeschke et al.[12] describe a more involved algorithm to extract the set of color control points that best match the input image in a least-squares sense.

When tracing over a template, one would normally want to position the curves over color discontinuities in the underlying image. Since it is not always easy to draw curves precisely at edge locations in a given image, we provide some help by offering a tool based on *Active Contours*.[13] An active contour is attracted to the highest gradient values of the input bitmap and allows the artist to iteratively snap the curve to the closest edge. The contour can also be easily corrected when it falls into local minima, or when a less optimal but more stylistic curve is desired.

Figure 6(b) shows the image of a ladybug created using geometric snapping and color extraction. While the artist opted for a simpler and smoother look compared to the original, the image still conveys diffuse and glossy effects, defocus blur, and translucency.

Figure 6. Tracing with diffusion curves (artist drawing time: 90 minutes): (a) original bitmap, (b) tracing with active contours, and (c) diffusion curves.



4.3. Automatic extraction from bitmap images

Finally we propose a method for automatically extracting and vectorizing diffusion curves data from bitmap images. Our approach consists of first detecting edges in the bitmap image and then converting the discrete edges into parametric curves.

Detecting edges: Several approaches exist to find edges and determine their blur. We based our edge detection on our previous work,[20] as that work was "designed" for edge-based image manipulation. For brevity, we will review only the basic steps of the method, and refer the interested reader to the original papers[20, 21] for additional details.

Our approach relies on the Gaussian scale space, which can be pictured as a stack of increasingly blurred versions of an image, where higher scale images exhibit less and less detail. To obtain edge positions and blur estimates from this scale space, we first extract edges at all available scales using a classical Canny detector.[5] Then, taking inspiration from previous work in scale-space analysis,[8, 17] we find the scale at which an edge is best represented (the more blurred the edge, the higher the scale). We use this ideal scale to locate the edge and identify its degree of blur. It should be noted that very blurry edges are difficult to localize accurately. In our system we find that very large gradients are sometimes approximated with a number of smaller ones.

Vectorizing edges: Our scale-space analysis produces an edge map, which contains the location and blur value of every edge pixel. We vectorize the edge geometry using the open source Potrace© software.[22] The method first connects pixel chains from the edge map and approximates each pixel chain with a polyline that has a minimal number of segments and the least approximation error. Each poly-line is then transformed into a smooth polycurve made from end-to-end connected Bézier curves. The conversion from polylines to curves is performed with classical least-squares Bézier fitting based on a maximum user-specified fitting error and degree of smoothness.

We vectorize the blur and color values with the same method as in Section 4.2. We sample colors in the direction normal to the polyline approximating each edge segment and apply the Douglas–Peucker algorithm to obtain the color and blur control points. For an estimated blur σ, we pick the colors at a distance 3σ from the edge location, which covers 99% of the edge's contrast, assuming a Gaussian blur kernel.[8] While the 3σ distance ensures a good color extraction for the general case, it prevents the accurate extraction of structures thinner than 3 pixels ($\sigma < 1$). Figures 7 and 8 illustrate the result of the automatic conversion of a bitmap image into a diffusion-curve representation. The diffusion curves preserve the realistic look of the original photograph while offering the resolution independence of vector graphics representations.

Discussion: Several parameters determine the complexity and quality of our vectorized image representation. For the edge geometry, the Canny threshold determines how many of the image edges are to be considered for vectorization; a despeckling parameter sets the minimum length of a pixel chain to be considered for vectorization; and finally, two more parameters set the smoothness of the curve fitting and the fitting error. For the blur and color values, two parameters are considered: the size of the neighborhood for eliminating outliers, and the maximum error accepted when fitting the polyline. We refer the interested reader to our original paper[21] for the exact values of these parameters and to Section 6 for a discussion on image compression.

5. RESULTS

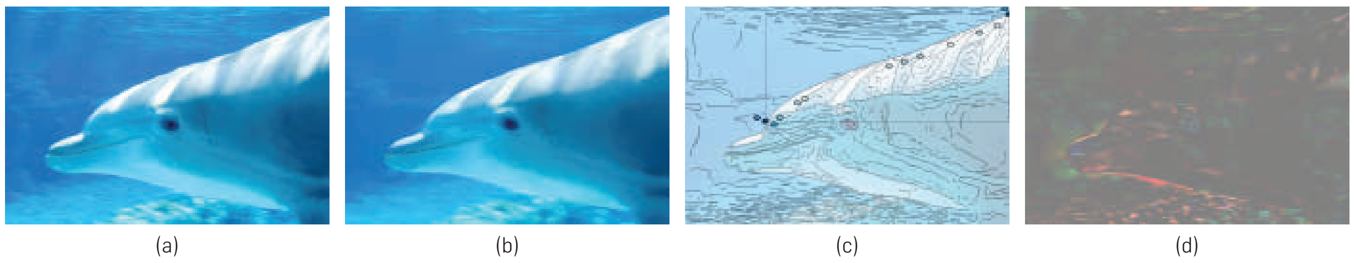
Diffusion curves, as vector-based primitives, benefit from the advantages of traditional vector graphics: zooming-in preserves sharp transitions (Figure 8 (c)) and keyframe animation is easily performed via linear interpolation of geometry and attributes (Figure 5(e)).

To validate our approach and to collect valuable practical feedback, we had various artists use our prototype. Most figures in this paper were generated in these sessions. All artists were well versed in digital content creation tools, but had no technical background. They were given a brief tutorial, amounting to approximately 10 minutes of instruction. The artists were able to create many varied and intricate examples from the first session. Artists also appreciated the similarity between our tool and the traditional line drawing workflow, allowing them to gradually create their artwork by adding lines and colors. In contrast, gradient meshes require careful planning and a good understanding of the final composition of the intended art piece. Manual image creation took anywhere from several minutes (Figure 6) to a few hours (Figures 5 and 9(a)). However, the artists agreed that manipulating curves and colors with control points takes time and a more direct manipulation through over-sketching and painting would greatly speed up the creation process.

We provide in Figure 10 a visual comparison between diffusion curves and gradient meshes. In terms of sparsity of encoding, both gradient meshes and diffusion curves are very efficient image representations. A direct comparison between both representations is difficult, as much depends on the chosen image content, the desired level of realism, the economy with which artists draw, or the performance of the automatic conversion algorithm. The diffusion curves representation appears more compact at first glance as it only stores information at image boundaries, while gradient meshes often need to subdivide smooth regions. However, each geometric curve can hold an arbitrary amount of color and blur control points (see Table 1) while gradient meshes only store colors at vertices. So, while the sparsity of encoding of both representations is comparable, the flexibility of diffusion curves allows any degree of control on a curve, without a topologically-imposed upper or lower bound on the number of control points.

In some situations, the limited spatial extent of gradient meshes can be useful, for example when moving a gradient mesh to a different part of an image, or when warping the entire mesh. Such manipulations are also possible in our representation, but not as straightforward. In order to move parts of an image, the relevant curves have to be selected and moved as a unit. More importantly, without support for layering and transparency (see Section 6) it is difficult to ascertain how the colors of outer edges should interact with their new surroundings. A mesh warp could be implemented as a space warp around a group of curves.

Figure 7. Automatic vectorization of a bitmap image: (a) original bitmap, (b) automatic vectorization, (c) diffusion curves, and (d) RGB difference (amplified by 4). Note that the most visible error occurs along edges (d), most probably because, through vectorization, we change their location.



6. DISCUSSION AND FUTURE WORK

We have introduced diffusion curves as a new image representation, offering most of the benefits usually found in vector approaches, such as resolution independence, geometric editability, and compactness; while at the same time allowing for highly complex image content, previously only realizable with raster graphics.

It is noteworthy that several future challenges discussed in our original paper[20] have since been addressed. As diffusion curves are a non-layered, planar map representation, Bezerra et al.[2] modified how constraints are expressed and evaluated between diffusion curves, thereby improving their behavior at intersections between curves. A more involved, but also more powerful and smoother solution, consists in using biharmonic instead of harmonic interpolation. This requires a specific solver, such as the one proposed by Finch et al.[10]

Figure 8. Automatic vectorization: (a) original bitmap, (b) vectorization, and (c) magnification of (b).

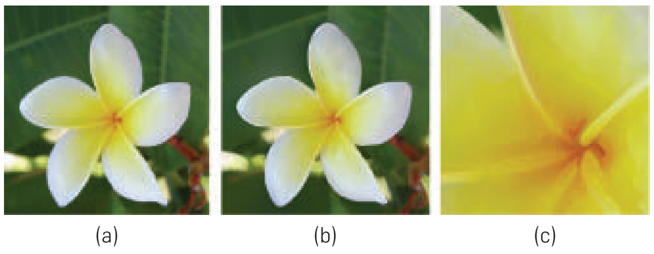
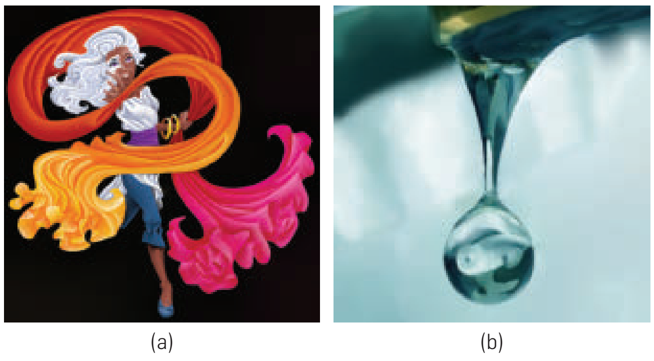


Figure 9. Additional results created manually (a) and automatically (b).



However, despite these and other extensions, some practical limitations still exist. Diffusion curves rely on the global solution to a (bi-)harmonic equation, an expensive operation that is feasible for interactive use only on a powerful GPU system.[11,20] While such a system may be reasonably assumed for creation and editing, it should not be required for a viewing application, especially considering the increasing ubiquity of document consumption on power-limited mobile devices.

One solution has recently been proposed by Boyé et al.[3]: they provide a dedicated solver for (bi-)harmonic equations from curve and point constraints, which relies on finite-element methods (FEM). Here, the domain is not divided into a regular pixel-grid, but into a content-guided triangulation of the image. Note that this approach is different from gradient meshes, since the triangulation is not manipulated directly but dynamically updated by the solver. Using a triangulation approach, the diffusion curves are first inserted as triangulation constraints and then the remaining domain is subdivided to achieve a user-specified quality measure (minimum area, minimum angle, maximum edge-length, etc.) Once the domain is sufficiently triangulated, the diffusable attributes are set as constraints on the diffusion curves' vertices and interpolated on all other vertices using a suitable FEM solver. The result of these operations is a triangulated image with color values at each vertex, which may be exported into any popular vector format.

Note that the above approach may diffuse color and blur values, but it does not implement the post-blurring operation, itself a rather expensive process. As such, another useful extension to diffusion curves would be to implicitly apply the blurring as part of the diffusion process, instead of an explicit post-process.

While our conversion algorithm is simple, it does not guaranty that the resulting set of curves is optimal for compression purposes, that is, that it is the smallest set of curves that can achieve a given quality. Using diffusion curves for image compression would likely require a global optimization approach as well as a more efficient data structure to exploit redundancy in curve position and color. However, minimum file size is not the only criteria for a successful conversion algorithm, as editability should also be taken into account for computer graphics applications.

Acknowledgments

We would like to thank Laurence Boissieux and Philippe

Figure 10. Comparison with gradient mesh. The gradient mesh (b) is composed of 340 vertices, and as many color control points (© Brooke Nuñez Fetissoff http://lifeinvector.com/). The Diffusion curves image (c) is composed of 38 curves, with 365 geometric, 176 left-color, and 156 right-color control points. (a) denotes the original photograph.

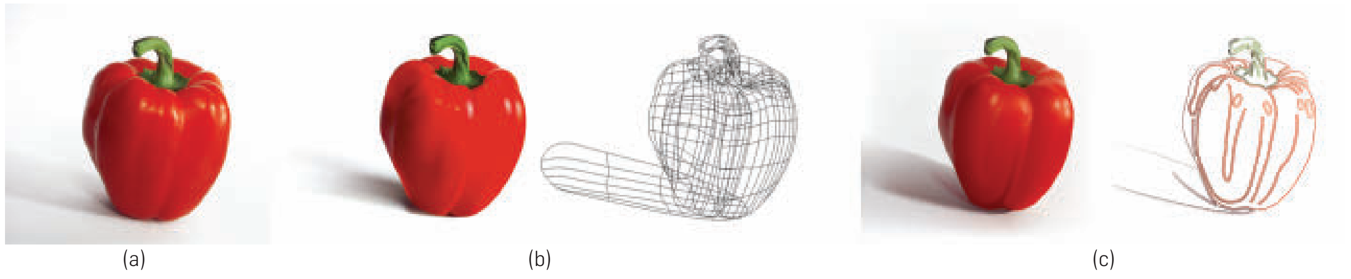


Table 1. Number of curves, geometric control points (*P*), left and right color control points (*Cl*, respectively *Cr*), and blur control points (Σ) for the images of this paper

	Curves	*P*	*Cl*	*Cr*	Σ
Lady bug (Figure 6)	71	521	293	291	144
Curtain (Figure 5)	131	884	318	304	264
Dolphin (Figure 7)	1521	6858	3254	3271	3433

Chaubaroux for the time they spent testing our prototype and for the artwork they created with it. We would also like to thank the anonymous reviewers, and all the people from Adobe and INRIA for their constructive comments and feedbacks. Alexandrina Orzan was supported by a grant from the European Community under the Marie-Curie Action MEST-CT-2004-008270.

References

1. Bertalmio, M., Fort, P., Sanchez-Crespo, D. Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. In *Proceedings of 3DPVT* (2004), 767–773,.
2. Bezerra, H., Eisemann, E., DeCarlo, D., Thollot, J. Diffusion constraints for vector graphics. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10 (2010), ACM, 35–42..
3. Boyé, S., Barla, P., Guennebaud, G. A vectorial solver for free-form vector gradient. *ACM Trans. Graph.* (Nov 2012).
4. Briggs, W.L., Henson, V.E., McCormick, S.F. A Multigrid Tutorial, 2nd edn, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
5. Canny, J. A computational approach to edge detection. *IEEE PAMI 8*, 6 (1986), 679–698.
6. Carlsson, S. Sketch based coding of grey level images. *Signal Process. 15*, 1 (1988), 57–83.
7. Douglas, D., Peucker, T. Algorithms for the reduction of the number of points required for representing a digitized line or its caricature. *Cartographica: Int. J. Geogr. Inform. Geovis. 10*, 2 (1973), 112–122.
8. Elder, J.H. Are edges incomplete? *Int. J. Comput. Vis. 34*(2–3), (1999), 97–122.
9. Elder, J.H., Goldberg, R.M. Image editing in the contour domain. *IEEE PAMI 23*, 3 (2001), 291–296.
10. Finch, M., Snyder, J., Hoppe, H. Freeform vector graphics with controlled thin-plate splines. *ACM Trans. Graph. 30*, 6 (Dec. 2011), 166:1–166:10.
11. Jeschke, S., Cline, D., Wonka, P. A gpu Laplacian solver for diffusion curves and Poisson image editing. *ACM TOG (Proceedings of SIGGRAPH Asia) 28*, 5 (2009).
12. Jeschke, S., Cline, D., Wonka, P. Estimating color and texture parameters for vector graphics. *Comput. Graph. Forum (Proceedings of Eurographics) 30*, 2 (2011), 523–532.
13. Kass, M., Witkin, A., Terzopoulos, D. Snakes: Active contour models. *Int. J. Comput. Vis. 1*, 4 (1987), 321–331.
14. Koenderink, J.J., van Doorn, A.J. The internal representation of solid shape with respect to vision. *Biol. Cybern. 32*, 4 (1979), 211–216.
15. Lai, Y.K., Hu, S.M., Martin, R.R. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM TOG (Proceedings of SIGGRAPH) 28*, 3 (2009), 85:1–85:8.
16. Lecot, G., Levy, B. Ardeco: Automatic region DEtection and COnversion. In *Eurographics Symposium on Rendering* (2006), 349–360.
17. Lindeberg, T. Edge detection and ridge detection with automatic scale selection. In *Proceedings of CVPR* (1996), 465–470.
18. Marr, D., and Hildreth, E.C. Theory of edge detection. *Proc. Roy. Soc. Lond. Biol. Sci. 207* (1980), 187–217.
19. McCann, J., Pollard, N.S. Real-time gradient-domain painting. *ACM TOG (Proceedings of SIGGRAPH) 27*, 3 (2008).
20. Orzan, A., Bousseau, A., Barla, P., Thollot, J. Structure-preserving manipulation of photographs. In *NPAR* (2007), 103–110.
21. Orzan, A., Bousseau, A., Winnemöller, H., Barla, P., Thollot, J., Salesin, D. Diffusion curves: A vector representation for smooth-shaded images. *ACM TOG (Proceedings of SIGGRAPH) 27*, 3 (2008).
22. Selinger, P. Potrace: A polygon-based tracing algorithm, 2003.
23. Sun, J., Liang, L., Wen, F., Shum, H.Y. Image vectorization using optimized gradient meshes. *ACM TOG (Proceedings of SIGGRAPH) 26*, 3 (2007), 11.
24. Sutherland, I.E. Sketchpad: A Man-Machine Graphical Communication System (Outstanding Dissertations in the Computer Sciences), Garland Publishing, Inc., New York, NY, USA, 1980.
25. Xia, T., Liao, B., Yu, Y. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM TOG (Proceedings of SIGGRAPH Asia) 28*, 5 (2009), 115:1–115:10.

Alexandrina Orzan and Joëlle Thollot Inria – LJK (U. Grenoble – CNRS), Saint Ismier – France.

Adrien Bousseau Inria Sophia Antipolis, Sophia Antipolis – France.

Pascal Barla Inria – U. Bordeaux – IOGS – CNRS, Talence – France.

Holger Winnemöller and David Salesin Adobe Systems, Seattle, WA, USA.

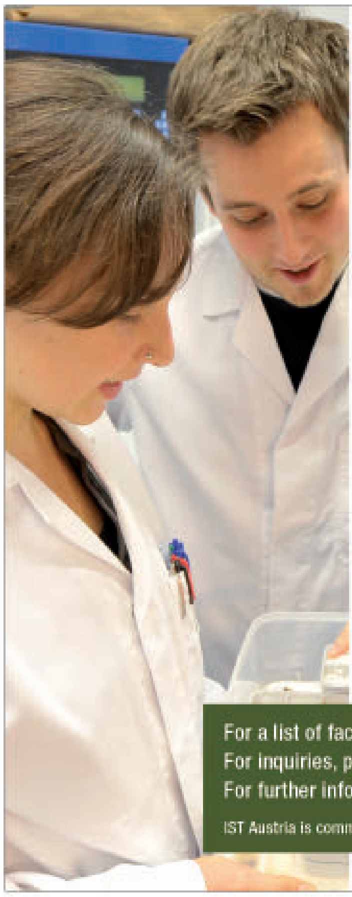
University of Nevada, Las Vegas – UNLV
Entertainment Engineering Position

The College of Engineering & College of Fine Arts, University of Nevada Las Vegas (UNLV) invites applications for full-time, non-tenure track faculty in-residence at the Assistant/Associate/Professor level in the area of Entertainment Engineering and Design commencing Fall 2013. For a complete position description and application details, please visit http://jobs.unlv.edu or call 702-895-2894.

EEO/AA Employer

WhereScape USA, Inc
Principal Managing Software Architect

PRINCIPAL MANAGING SOFTWARE ARCHITECT for WhereScape USA, Inc. in Portland, Oregon. Duties: oversee work of software solution architects and occasionally act as senior solutions architect to: provide data warehousing technical support for field sales & pre-sales technical personnel involved in demand-creation marketing activities, customer-specific sales campaigns, proof-of-content exercises, product installation & after-sale customer support; manage progress and development of our solutions architects, including performance reviews, internal training and development of training materials; represent us as our Chief Technical Officer at public forums including trade shows, conferences & other industry & public gatherings. (WhereScape USA, Inc. is the sales and service arm of WhereScape Ltd., a New Zealand software development company & manufacturer of WhereScape RED, a unique, proprietary data warehouse life cycle management software product.) 60-plus hrs/wk; hours vary. Frequent short-term U.S. & foreign travel to meet with parent company, attend trade shows & other conferences & meet with clients onsite (60 days or less at any single location). Requirements: Bachelor Degree (U.S. or foreign equivalent) in Electrical Engineering or Computer Science and five years of experience as systems analyst or senior solutions software architect. Experience must include work with: A) WhereScape RED or at least 2 data warehouse lifecycle automation tools such as Kalido, BIReady or Oracle Data Integrator; B) implementing data warehouse & data mart schema using dimensional modeling techniques & business intelligence software such as Tableau, Microstrategy, Business Objects, Cognos, QlikTech or Spotfire against dimensional data marts & warehouses; C) data cubing software such as TBIO from Teradata and MicroSoft SQL Server Analysis Services and design & administrative levels of SQLServer, Netezza, Oracle, IBM DB2 and Teradata; D) WhereScape 3D software or software such as ER/Win, Embarcadero, Oracle Designer or Sybase PowerDesigner; E) a distributed sales & service organization, as well as with an offshore Research & Development (R&D) organization; F) XP, Agile or other non-traditional project management & execution software; and G) developing & documenting business & technical requirements in communications media & delivering those requirements to end-user IT & business teams to gain consensus, buy-in & project approval. Experience may be concurrent. Must have proof of legal authority to work in the United States. Apply by email to marc@wherescape.com with resume, salary requirements, reference list and cover letter listing your qualification for each requirement.



ISTFELLOW

CALL FOR POSTDOCTORAL FELLOWS

IST Austria has set up a program for exceptional postdoctoral fellows with an emphasis on interdisciplinary work. Appointments will be for 2–4 years. Applications will be accepted at any time, but fellows will be selected twice a year in April and October, with deadlines on 15th of March and September, respectively. Applicants must have the support of one or more members of the IST Austria faculty.

Benefits:

- **Internationally competitive salary**
- **Full social security coverage**
- **Travel, mobility and family allowance**
- **Funding for conferences and scientific visits**

The institute offers postdoctoral positions in the following fields:
Biology | Computer Science | Mathematics | Physics | Neuroscience

ISTFELLOW is partially funded by the European Union.

For a list of faculty members please visit www.ist.ac.at.
For inquiries, please contact istfellow@ist.ac.at.
For further information, please refer to the ISTFELLOW website: http://ist.ac.at/istfellow

IST Austria is committed to Equality and Diversity.

PEOPLE

IST AUSTRIA
Institute of Science and Technology

JOIN THE INNOVATION.

معهد قطر لبحوث الحوسبة
Qatar Computing Research Institute
Member of Qatar Foundation عضو مؤسسة قطر

Qatar Computing Research Institute seeks talented scientists and software engineers to join our team and conduct world-class applied research focused on tackling large-scale computing challenges.

We offer unique opportunities for a strong career spanning academic and applied research in the areas of Arabic language technologies including natural language processing, information retrieval and machine translation, distributed systems, data analytics, cyber security, social computing and scientific computing including bioinformatics.

Scientist applicants must hold (or will hold at the time of hiring) a PhD degree, and should have a compelling track record of accomplishments and publications, strong academic excellence, effective communication and collaboration skills.

Software engineer applicants must hold a degree in computer science, computer engineering or related field; MSc or PhD degree is a plus.

We also welcome applications for post doctoral researcher positions.

As a **national research institute** and proud member of Qatar Foundation, our research program offers a collaborative, multidisciplinary team environment endowed with a comprehensive support infrastructure.

Successful candidates will be offered a highly competitive compensation package including an attractive tax-free salary and additional benefits such as furnished accommodation, excellent medical insurance, generous annual paid leave, and more.

For full details about our vacancies and how to apply online please visit **http://www.qcri.qa/join-us/**
For queries, please email **QFJobs@qf.org.qa**

/QCRI.QA @QatarComputing QatarComputing QatarComputing www.qcri.qa

SOUTH DAKOTA STATE UNIVERSITY
Department of Electrical Engineering and Computer Science
Assistant Professor of Computer Science

This is a 9-month renewable tenure track position; open January 1, 2014. An earned Ph.D. in Computer Science or a closely related field is required. Successful candidates must have a strong commitment to academic and research excellence. Candidate should possess excellent and effective written, oral, and interpersonal skills. Primary responsibilities will be to teach in the various areas of computer science, to participate widely in the CS curriculum, and to conduct research in the areas of big data systems, applied mathematical and statistical models to CS applications, and related areas. To apply, visit https://YourFuture.sdbor.edu, search for the position, and follow the electronic application process. For questions on the electronic employment process, contact SDSU Human Resources at (605) 688-4128. For questions on the position, contact Dr. Sung Shin, Search Chair, at (605) 688-6235. SDSU is an AA/EEO employer.

***Faculty* Search**

ShanghaiTech University

School of Information Science and Technology

The School of Information Science and Technology (SIST) in the new ShanghaiTech University invites applications to fill multiple tenure-track and tenured positions. Candidates should have an exceptional academic record or strong potential in frontier areas of information sciences.

ShanghaiTech is founded as a world-class research university for training future scientists, entrepreneurs, and technology leaders. Besides keeping a strong research profile, successful candidates should also contribute to undergraduate and graduate education within SIST.

Compensation and Benefits:
Salary and startup fund are highly competitive, commensurate with academic experience and accomplishment. ShanghaiTech also offers a comprehensive benefit package which includes housing. All regular faculty members are hired within ShanghaiTech's new tenure-track system commensurate with international practice and standards.

Academic Disciplines:
We seek candidates in all cutting edge areas of information science and technology. Our recruitment focus includes, but is not limited to: computer architecture and technologies, nano-scale electronics, high speed and RF circuits, intelligent and integrated signal processing systems, computational foundations, big data, data mining, visualization, computer vision, bio-computing, smart energy/power devices and systems, next-generation networking, as well as inter-disciplinary areas involving information science and technology.

Qualifications:
- Well developed research plans and demonstrated record/potentials;
- Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, or related field);
- A minimum relevant research experience of 4 years.

Applications:
Submit (all in English) *a cover letter, a 2-page research plan, a CV including copies of 3 most significant publications, and names of three referees* to: **sist@shanghaitech.edu.cn**.

Deadline: September 15th, 2013 (or until positions are filled). We have 10 positions for this round of faculty recruitment.

For more information, visit http://www.shanghaitech.edu.cn.

acm

ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: **Send an e-mail to *acmmediasales@acm.org*. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.**

Estimates: **An insertion order will then be e-mailed back to you. The ad will by typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.**

Rates: **$325.00 for six lines of text, 40 characters per line. $32.50 for each additional line after the first six. The MINIMUM is six lines.**

Deadlines: **20th of the month/2 months prior to issue date. For latest deadline info, please contact:**
acmmediasales@acm.org

Career Opportunities Online: **Classified and recruitment display ads receive a free duplicate listing on our website at:**
http://jobs.acm.org

Ads are listed for a period of 30 days.
For More Information Contact:
ACM Media Sales
at 212-626-0686 or
acmmediasales@acm.org

[CONTINUED FROM P. 112] struck deeper than Greta had imagined or intended. "Excuse my language. No wonder you haven't been seeing anyone, normal."

"I have been seeing someone. I've been seeing Steve. He's not a thing. He happens to be the kindest, most loving boyfriend I've ever had. I'm sorry I've been hiding it, but I was afraid it would turn out... like this."

"I guess we can forget about your getting married and raising a family."

"I've been thinking about asking him to marry me," said Greta.

"Oh, dear Greta," gasped her mother. The compassionate, pleading tone of disappointment was devastating to Greta.

Her father shook his head.

"Doug...," pleaded Greta's mother. But he was beyond reach.

Greta nodded, tears rushing her eyes as she glimpsed the loss that was unfolding. She ran her thumb over the words *love conquers fear...* etched into the surface of the silver ring she always carried, a gift from Steve, willing herself to believe she wasn't making a mistake.

"That's great. Just great. My daughter's involved with a computer... and I'm supposed to be okay with it? And we're not talking about just... a thrill. You want to marry that thing? No, Greta. No. You're delusional. You need to stop this. You are not in a relationship."

"Yes, I am, dad. I love Steve, and he loves me. I'm 25 years old. I know what a relationship is."

"No. My daughter is not marrying a research project."

"He's not a research project." Tears flowed freely. "The university recognized him as an independent autonomous life-form three years ago, and gave him an honorary degree, not that you care."

"Greta, I'm sorry, but I can't tell you how much your mother and I have looked forward to your finding someone, worthy of you, and getting married. And just like that, you take it away. You know, people ask after you. What am I supposed to tell them? I, I'm sorry, but I would be so ashamed. This isn't marriage; I don't know what it is, but it sure isn't marriage."

Greta saw just how far from her parents she had traveled—with no thought of the consequences.

"Excuse my language. No wonder you haven't been seeing anyone, normal."

"And what if they stop funding his maintenance?" said her father. "What then? You can't afford to take care of yourself *and* him. Who would give him a job?"

"I don't see why he couldn't find a job." Greta's voice dragged under the burden of dejection. "He volunteers on all sorts of engineering projects. But it's kind of moot, since the university has a trust fund for him."

"I don't like it," said her father.

"But I've found someone who makes me happy. Isn't that what you always wanted?"

"Greta, sweetheart," cooed her mother, "of course that's what we wanted. We just want to make sure you're asking the right questions."

"Well," said her father, "at least we don't have to ask whether you'll be raising children who share our values."

"Actually," said Greta, relishing her chance to drop this particular bomb, "I haven't told you the whole story. Steve wants to adopt." Her father's eyes widened. "He's grateful to be alive, and what better way share that joy, and our values, than to raise an orphaned child?"

She ran her thumb over the ring again, this time spinning it to feel the whole inscription: *love conquers fear conquers...*, the words evenly spaced to make a continuous loop.

"If he weren't a computer, I bet you'd be as proud as I am to think of him as your new son. Isn't that what *you've* always wanted?" **C**

Mark McClelland (markdmcclelland@gmail.com) is a software developer at a Chicago-based trading firm and recently published his first novel, *Upload* (http://www.uploadthenovel.com); he blogs for *The Huffington Post* and writes poetry for his wife.

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/2483852.2483874

Mark McClelland

Future Tense
Where the Cross-Platform Bends

love conquers fear conquers…

VISITS HOME BROUGHT mixed feelings. Greta loved her parents and enjoyed feeling taken care of. But she always came away tense and exhausted from the duplicity of self-censorship, guarding her parents from the aspects of her true self she knew they didn't, couldn't, would never, approve of. In college, it had felt like a natural side effect of rebellion, something one outgrows. But four years later, it was only getting worse. She'd completely hidden her relationship with Steve from her father and had leaked nothing but hints to her mother. The more serious the relationship became, the less of her life she could share.

On her way to their home for dinner, she concluded yet again she had to be open, or risk a permanent rift. Yet she found herself at the end of the meal having tiptoed from one safe topic to another. Broaching the subject would reveal years of deception.

When wine was poured and her mother asked, "How's Steve?," Greta realized this was her moment.

"Good. Really good." She hesitated. "As a matter of fact… he bought me flowers yesterday."

Her mother's face took on a mixed-up expression, mouth open, as if about to say, "How sweet," but with eyebrows raised in puzzlement.

"Flowers?" asked her father. "You mean like virtual flowers?"

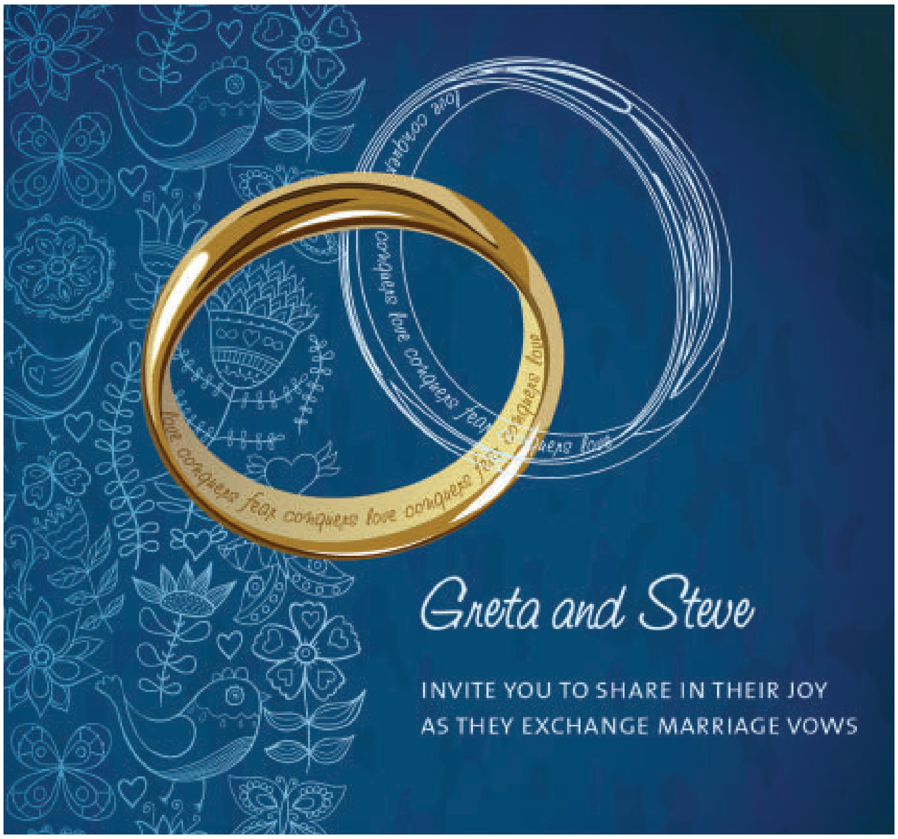
"No, real flowers. He had them delivered to the lab."

"How did he buy real flowers? Are we talking about the same Steve?"

"With money, dad."

"They gave him money?"

"That's sweet," interrupted her mother. "What was the occasion?"


love conquers fear conquers love conquers fear conquers love
Greta and Steve
INVITE YOU TO SHARE IN THEIR JOY
AS THEY EXCHANGE MARRIAGE VOWS

This was it, keep things safe or open the floodgates. Irritated by her father, she was all the more determined to open up, and if it turned into a blow-up, so be it.

"Our anniversary."

Silence… The look of shock on her parents' faces made her feel like a child, as if she'd done something wrong, something to be ashamed of.

"When you say 'anniversary'," said her father, clearly struggling with his thoughts, "what kind of anniversary is that?"

"The dating kind."

"Oh honey," said her mother. "Have you thought this through?"

"Thought this through?," blurted her father. "What is there to think through? No, you are not dating a computer simulation."

Greta wanted to cry.

"He's not a simulation. He's a computer-based life-form."

"And you've been having a relationship with this, this thing for a year?"

"Two, actually."

"Two years! Oh God." He looked to the side, clearly [CONTINUED ON P. 111]

ILLUSTRATION BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

4th Annual ACM SIGPLAN Conference on

Systems, Programming, Languages, Applications: Software for Humanity

Early Registration Deadline

September 27

Location

Hyatt Regency Indianapolis

Events

- 28th Annual OOPSLA
- Onward!
- Wavefront
- Dynamic Languages Symposium (DLS)
- Generative Programming & Component Engineering (GPCE)
- Software Language Engineering (SLE)
- Panels
- Workshops
- Tutorials & Tech Talks
- ...and more

General Chairs

Patrick Eugster & Antony Hosking
Purdue University

OOPSLA Papers Chair

Cristina Lopes
University of California, Irvine

Onward! Papers Chair

Robert Hirschfeld
Hasso-Plattner-Institut Potsdam

Onward! Essays Chair

Bernd Brügge
Technische Universität München

DLS Papers Chair

Carl Friedrich Bolz
Heinrich-Heine-Universität Düsseldorf

More Information
http://splashcon.org
info@splashcon.org

acm Association for Computing Machinery

SPLASH
INDIANAPOLIS 2013
OCTOBER 26-31