

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

06/2012 VOL.55 NO.6

Judea Pearl

ACM's A.M. Turing
Award Winner



MATCH YOUR SERVER TO YOUR BUSINESS. ONLY PAY FOR WHAT YOU NEED!



- Change hardware configurations in real time to meet your business needs.
- Control costs with pay-per-configuration and hourly billing
- Up to 6 Cores, 24 GB RAM, 800 GB storage
- 2000 GB of traffic included free
- Parallels® Plesk Panel 10 for unlimited domains, reseller ready.

1&1 DYNAMIC CLOUD SERVER

Starting at

\$49.99
PER MONTH*



- **NEW:** Monitor and manage your cloud server through 1&1 mobile apps for Android™ and iPhone®.



www.1and1.com



*Set-up fee of \$49 applies. Base configuration includes 1 core, 1 GB Ram, 100 GB Storage. Other terms and conditions may apply. Visit www.1and1.com for full promotional offer details. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners. © 2012 1&1 Internet. All rights reserved.



Frederick P.
BROOKS, JR.



Charles P. (Chuck)
THACKER



Charles W.
BACHMAN



Stephen A.
COOK



Barbara
LISKOV

34



Richard M.
KARP



Leslie G.
VALIANT



Marvin
MINSKY



Judea
PEARL



Adi
SHAMIR

TURING AWARD WINNERS TOGETHER?



Richard E.
STEARNS



John
HOPCROFT



Robert E.
KAHN



Vinton G.
CERF



Robert
TARJAN



Ivan
SUTHERLAND



Butler
LAMPSON



Juris
HARTMANIS



Andrew C.
YAO



Donald E.
KNUTH



Dana S.
SCOTT



Raj
REDDY



Fernando J.
CORBATÓ



Edward A.
FEIGENBAUM



Alan C.
KAY



William (Velvel)
KAHAN



Frances E.
ALLEN



E. Allen
EMERSON



Niklaus
WIRTH



Ken
THOMPSON



Leonard
ADLEMAN



Ronald
RIVEST



Edmund
CLARKE



Joseph
SIFAKIS

WHAT DO YOU GET WHEN YOU PUT

**THE ACM TURING
CENTENARY CELEBRATION**
A ONCE-IN-A-LIFETIME
EVENT THAT YOU'LL
NEVER FORGET...
AND YOU'RE INVITED!

CHAIR:

Vint Cerf, '04 Turing Award Winner,
Chief Internet Evangelist, Google

PROGRAM CO-CHAIRS:

Mike Schroeder, Microsoft Research

John Thomas, IBM Research

Moshe Vardi, Rice University

MODERATOR:

Paul Saffo, Managing Director of Foresight, Discern
Analytics and Consulting Associate Professor, Stanford

WHEN:

June 15th & June 16th 2012

WHERE:

Palace Hotel, San Francisco CA

HOW:

Register for the event at turing100.acm.org

REGISTRATION IS FREE-OF-CHARGE!



THE A.M. TURING AWARD, often referred to as the "Nobel Prize" in computing, was named in honor of Alan Mathison Turing (1912-1954), a British mathematician and computer scientist. He made fundamental advances in computer architecture, algorithms, formalization of computing, and artificial intelligence. Turing was also instrumental in British code-breaking work during World War II.

Departments

- 5 **Letter From USACM Chair**
USACM and U.S. Legislation
By Eugene H. Spafford
-
- 6 **Letters To The Editor**
The Halting Problem in the Clear Light of Probability
-
- 10 **BLOG@CACM**
Protecting Against Data Breaches; Living with Mistakes
Jason Hong writes about security breaches and offers a three-pronged approach. Greg Linden discusses the differences between computers and the human brain and their tolerance of errors.
-
- 12 **CACM Online**
Don't Throw the Baby Out with the Bathwater
By Scott E. Delman
-
- 37 **Calendar**
-
- 131 **Careers**

Last Byte

- 133 **Puzzled**
Solutions and Sources
By Peter Winkler
-
- 136 **Q&A**
A Sure Thing
Artificial intelligence pioneer Judea Pearl discusses probability, causation, the calculus of intervention, and counterfactuals.
By Leah Hoffmann

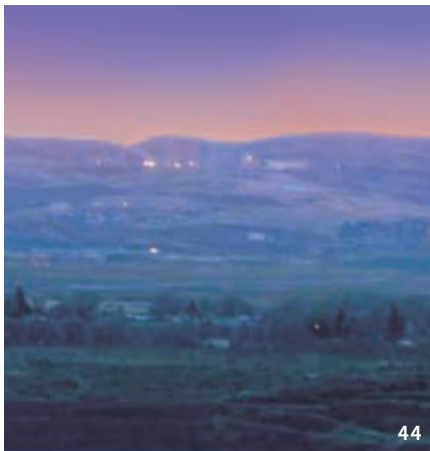
News

- 13 **Analyzing Medical Data**
Electronic patient records contain a treasure trove of data, and researchers are using natural language processing technology to mine the structured data and free text.
By Gregory Goth
-
- 16 **Smarter Photography**
Improvements in camera hardware, image processing, camera-photographer interfaces, and image viewing are advancing the state of the art in digital photography.
By Gary Anthes
-
- 19 **Data Mining Meets City Hall**
Local and national governments are turning to open data to cut their costs, increase transparency and efficiency, and respond to the needs of citizens.
By Leah Hoffmann
-
- 22 **Game Changer**
Judea Pearl's passionate advocacy of the importance of probability and causality helped revolutionize artificial intelligence.
By Neil Savage
-
- 24 **An Influential Theoretician**
Sanjeev Arora, winner of the 2011 ACM-Infosys Award, discusses his pivotal role in theoretical computer science.
By Paul Hyman

Viewpoints

- 26 **The Business of Software**
A Measure of Control
Some limitations on measurements in software.
By Phillip G. Armour
-
- 29 **Inside Risks**
The Cybersecurity Risk
Increased attention to cybersecurity has not resulted in improved cybersecurity.
By Simson L. Garfinkel
-
- 33 **Kode Vicious**
Scale Failure
Using a tool for the wrong job is OK until the day when it isn't.
By George V. Neville-Neil
-
- 35 **Privacy and Security**
Security of the Internet and the Known Unknowns
Seeking answers to questions about Internet vulnerabilities.
By Chris Hall
-
- 38 **The Profession of IT**
The Myth of the Elevator Pitch
Instead of pitching, listen and offer.
By Peter J. Denning and Nicholas Dew
-
- 41 **Viewpoint**
Why Computer Scientists Should Care About Cyber Conflict and U.S. National Security Policy
Cybersecurity and policy issues for computer scientists.
By Herbert Lin

Practice



- 44 **A Guided Tour of Data-Center Networking**
A good user experience depends on predictable performance within the data-center network.
By Dennis Abts and Bob Felderman

- 52 **Modeling People and Places with Internet Photo Collections**
Understanding the world from the sea of online photos.
By David Crandall and Noah Snaveley

- 61 **Real-Time Computer Vision with OpenCV**
Mobile computer-vision technology will soon become as ubiquitous as touch interfaces.
By Kari Pulli, Anatoly Baksheev, Kirill Korniyakov, and Victor Eruhimov



Articles' development led by **acmqueue**
queue.acm.org

Contributed Articles



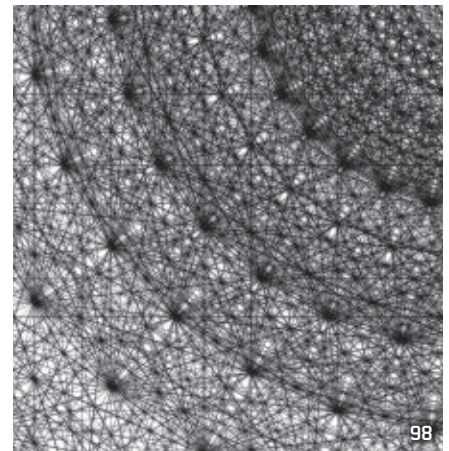
- 70 **Why Rumors Spread So Quickly in Social Networks**
A few hubs with many connections share with many individuals with few connections.
By Benjamin Doer, Mahmoud Fouz, and Tobias Friedrich

- 76 **Trust Extension for Commodity Computers**
A user's trust in a single device can be extended to many other devices.
By Bryan Parno

**About the Cover:**

Judea Pearl, winner of the 2011 ACM A.M. Turing Award, pictured outside his office on the UCLA campus by photographer Richard Morgenstein (www.morgenstein.com). Pearl was recognized for his innovations that enabled remarkable advances in the partnership between humans and machines—the foundation of AI.

Review Articles



- 86 **Data Services**
Exploring the technology trends in basic, integrated, and cloud data services.
By Michael J. Carey, Nicola Onose, and Michalis Petropoulos
- 98 **The GCT Program Toward the P vs. NP Problem**
Exploring the power and potential of geometric complexity theory.
By Ketan D. Mulmuley

Research Highlights

- 110 **Technical Perspective**
Reconstructing the Unknown, Balancing Structure and Uncertainty
By Pablo A. Parrilo
- 111 **Exact Matrix Completion via Convex Optimization**
By Emmanuel Candès and Benjamin Recht
- 120 **Technical Perspective**
The Fox and the Hedgehog
By Peter Lee
- 121 **Lightweight Modular Staging: A Pragmatic Approach to Runtime Code Generation and Compiled DSLs**
By Tiark Rompf and Martin Odersky



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.

Executive Director and CEO

John White

Deputy Executive Director and COO

Patricia Ryan

Director, Office of Information Systems

Wayne Graves

Director, Office of Financial Services

Russell Harris

Director, Office of SIG Services

Donna Cappel

Director, Office of Publications

Bernard Rous

Director, Office of Group Publishing

Scott E. Delman

ACM COUNCIL

President

Alain Chesnais

Vice-President

Barbara G. Ryder

Secretary/Treasurer

Alexander L. Wolf

Past President

Wendy Hall

Chair, SGB Board

Vicki Hanson

Co-Chairs, Publications Board

Ronald Boisvert and Jack Davidson

Members-at-Large

Vinton G. Cerf; Carlo Ghezzi;

Anthony Joseph; Mathai Joseph;

Kelly Lyons; Mary Lou Soffa; Salil Vadhan

SGB Council Representatives

G. Scott Owens; Andrew Sears;

Douglas Terry

BOARD CHAIRS

Education Board

Andrew McGettrick

Practitioners Board

Stephen Bourne

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Fabrizio Gagliardi

ACM India Council

Anand S. Deshpande, PJ Narayanan

ACM China Council

Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs

Ronald F. Boisvert; Jack Davidson

Board Members

Marie-Paule Cani; Nikil Dutt; Carol Hutchins;

Joseph A. Konstan; Ee-Peng Lim;

Catherine McGeoch; M. Tamer Ozsu;

Vincent Shen; Mary Lou Soffa

ACM U.S. Public Policy Office

Cameron Wilson, Director

1828 L Street, N.W., Suite 800

Washington, DC 20036 USA

T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association

Chris Stephenson,

Executive Director

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.

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

Jack Rosenberger

Web Editor

David Roman

Editorial Assistant

Zarina Strakhan

Rights and Permissions

Deborah Cotton

Art Director

Andrij Borys

Associate Art Director

Alicia Kubista

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

acmhlp@cacm.org

Letters to the Editor

letters@cacm.acm.org

WEB SITE

<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 acmm mediasales@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

EDITORIAL BOARD

EDITOR-IN-CHIEF

Moshe Y. Vardi

vic@cacm.acm.org

NEWS

Co-Chairs

Marc Najork and Prabhakar Raghavan

Board Members

Hsiao-Wuen Hon; Mei Kobayashi;

William Pulleyblank; Rajeev Rastogi;

Jeannette Wing

VIEWPOINTS

Co-Chairs

Susanne E. Hambrusch; John Leslie King;

J Strother Moore

Board Members

P. Anandan; William Aspray;

Stefan Bechtold; Judith Bishop;

Stuart I. Feldman; Peter Freeman;

Seymour Goodman; Shane Greenstein;

Mark Guzdial; Richard Heeks;

Rachelle Hollander;

Richard Ladner; Susan Landau;

Carlos Jose Pereira de Lucena;

Beng Chin Ooi; Loren Terveen

Q PRACTICE

Chair

Stephen Bourne

Board Members

Eric Allman; Charles Beeler; Bryan Cantrill;

Terry Coatta; Stuart Feldman; Benjamin Fried;

Pat Hanrahan; Marshall Kirk McKusick;

Erik Meijer; George Neville-Neil;

Theo Schlossnagle; Jim Waldo

The Practice section of the CACM

Editorial Board also serves as

the Editorial Board of *COMMUNIQUE*.

CONTRIBUTED ARTICLES

Co-Chairs

Al Aho and Georg Gottlob

Board Members

Robert Austin; Yannis Bakos; Elisa Bertino;

Gilles Brassard; Kim Bruce; Alan Bundy;

Peter Buneman; Erran Carmel;

Andrew Chien; Peter Druschel; Blake Ives;

James Larus; Igor Markov; Gail C. Murphy;

Shree Nayar; Bernhard Nebel; Lionel M. Ni;

Sriram Rajamani; Marie-Christine Rousset;

Avi Rubin; Krishan Sabnani;

Fred B. Schneider; Abigail Sellen;

Ron Shamir; Marc Snir; Larry Snyder;

Manuela Veloso; Michael Vitale; Wolfgang

Wahlster; Hannes Werthner;

Andy Chi-Chih Yao

RESEARCH HIGHLIGHTS

Co-Chairs

Stuart J. Russell and Gregory Morrisett

Board Members

Martin Abadi; Stuart K. Card; Jon Crowcroft;

Alon Halevy; Monika Henzinger;

Maurice Herlihy; Norm Jouppi;

Andrew B. Kahng; Mendel Rosenblum;

Ronitt Rubinfeld; David Salesin;

Guy Steele, Jr.; David Wagner;

Alexander L. Wolf; Margaret H. Wright

WEB

Co-Chairs

James Landay and Greg Linden

Board Members

Gene Golovchinsky; Marti Hearst;

Jason I. Hong; Jeff Johnson; Wendy E. MacKay



ACM Copyright Notice

Copyright © 2012 by Association for

Computing Machinery, Inc. (ACM).

Permission to make digital or hard copies

of part or all of this work for personal

or classroom use is granted without

fee provided that copies are not made

or distributed for profit or commercial

advantage and that copies bear this

notice and full citation on the first

page. Copyright for components of this

work owned by others than ACM must

be honored. Abstracting with credit is

permitted. To copy otherwise, to republish,

to post on servers, or to redistribute to

lists, requires prior specific permission

and/or fee. Request permission to publish

from permissions@acm.org or fax

(212) 869-0481.

For other copying of articles that carry a

code at the bottom of the first or last page

or screen display, copying is permitted

provided that the per-copy fee indicated

in the code is paid through the Copyright

Clearance Center; www.copyright.com.

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 acmhlp@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



USACM and U.S. Legislation

Computing professionals sometimes find policy issues ill-defined, confusing, or irrelevant. Perhaps that is why—to date—policy participation has been limited within our community.

When confronted with an issue, someone with a computing background typically gathers data, applies a decision algorithm, and makes a definitive choice. We are, after all, dealing with ones and zeroes on a daily basis!

Real-life policy choices are not quite that simple, however, especially those with political aspects. A policy choice often includes considerations about consistency with past choices (and laws), philosophical positions about the role of government, economic consequences, reputation and image, timing, and other factors that seldom present a single, obvious choice. Furthermore, choices are compounded by political considerations (especially near election seasons), and by simple ignorance (for example, the late Sen. Ted Stevens' description of the Internet as a "series of tubes"). The way these choices and priorities are mixed often result in outcomes that perplex—and possibly enrage—observers.

One recent example was the controversy over the Stop Online Piracy Act (SOPA) and its companion Preventing Real Online Threats to Economic Creativity and Theft of Intellectual Property Act (the PROTECT IP Act, or PIPA). In late 2011, people throughout the U.S. (and elsewhere) grew concerned about these proposed laws, with particular attention focused on SOPA. They feared the legislative language would inappropriately allow authorities to shut down whole domains and penalize fair use of copyrighted materials, among other possible results. An online protest grew, eventually resulting in a massive online "blackout" on January 18, 2012.

ACM's U.S. Public Policy Council (USACM) was involved in this issue well before the blackout. USACM's mandate is to help policymakers understand the computing-related aspects of their ac-

tivities. We are uniquely positioned for this task because ACM is a non-partisan, professional organization devoted to computing. USACM focuses on the technical issues of computing, while acknowledging there are often more factors involved in policy. Thus, our usual mode of operation is to provide education and background on issues, although we sometimes take an advocacy position.

It was clear that SOPA (and PIPA) could result in adverse effects for some Internet users—a point not lost on some legislators. However, they also believed other factors outweighed or mitigated these risks. Three of these were specifically presented to USACM during our discussions about this legislation:

- There are millions of people in the U.S. whose jobs, directly or indirectly, depend on intellectual property protection and licensing. Pension funds and investors hold significant equity positions in companies that depend on intellectual property. New and existing companies (including many not in the entertainment industry) depend on intellectual property protections to compete in the international marketplace.

- As an international issue, the U.S. lacks legal jurisdiction where the most egregious violations originate. Some of the host nations have agendas that include the weakening or destruction of the U.S. economy, thus this activity is tolerated. Some governments have close ties to the criminal elements involved and will not take action. Others are resource-constrained and unable to mount investigations and prosecutions.

- Most legal systems depend on discretion. Everything from deciding whether someone should get a speeding ticket to capital murder cases depend on some amount of discretion and expenditure of resources.

USACM did not address any of those concerns. Instead, we focused on the proposed legislation's technical aspects. Members of USACM provided briefings to Congressional staff and others about the ramifications should the legislation be passed. We also submitted formal statements to legislators in both the House and the Senate, with particular emphasis on how the proposed legislation could damage the deployment of the DNS Security Extensions (DNSSEC). In mid-January, one senior staff member told us that our private meeting gave them the first true understanding of how DNS and DNSSEC worked, and thus why the legislation was problematic.

The combination of technical problems and political pressure were overwhelming; the bills were eventually withdrawn by their sponsors for further consideration. Some in the Internet community viewed this as a victory, but it will be fleeting: the underlying problems of intellectual property violations and fraud continue. Thus, we expect ongoing pressure for some legislative proposals.

The computing community must continue to be involved in the process to ensure that all such legislation is technically sound and consistent with our vision of computing. It involves being sensitive to the nuances and factors that influence policy beyond simply our own narrow interests, and continuing to provide expert technical advice. It may be confusing, but it is not impossible.

You can learn more about USACM (including how to participate) at <http://usacm.acm.org/>. 

Eugene H. Spafford (spaf@purdue.edu) is chair of USACM, a professor of computer science, and founder and executive director of Purdue University's Center for Education and Research in Information Assurance and Security (CERIAS), West Lafayette, IN.

© 2012 ACM 0001-0782/12/06 \$10.00

The Halting Problem in the Clear Light of Probability

SBARRY COOPER'S article "Turing's Titanic Machine?" (Mar. 2012) considered Alan Turing's contributions to computability theory, concentrating on the halting problem; that is, decide whether a given program will stop or continue indefinitely. The fact that in general no one can know makes it undecidable. Moreover, it is fundamental to many proofs of undecidability and algorithm complexity, and computer scientists have used it to devise a hierarchy of complexity classes. However, such complexity analysis has also led to seeming contradictions.

Boolean satisfiability (SAT) has been proved *NP*-complete, so reasonable-size SAT problems should not be solvable, yet in practice some have been solved quickly. Just as there are Turing machines that do not halt, some SAT problems cannot be solved. But how many Turing machines stop? And how many SAT problems can be solved?

Cooper considered relatively recent work on non-classical computers (such as biological computation and quantum computers) and even mentioned evolving intelligent machines. For example, by recasting Turing's halting problem in a probabilistic light, we were able to show that programs (in a minimal computer architecture) do not, on average, halt.¹ We addressed the halting problem from a probabilistic point of view; that is, if a random program starts, it is, with overwhelming probability, not going to stop.

Execution traces of random programs are typically short since the program counter might fall into a tight loop in which a small number of instructions repeats infinitely. Likewise, most traces in human-written software cover only a small fraction of the program. In both human-written and random programs most of the code is not run, so might as well be random. Studying random execution

paths could give results on the ineffectiveness of random testing of human-written programs and its inability to cover the software under test and lead to improved search-based and other testing methods.

Software engineers do not write random programs; neither does genetic programming. Both human-written and genetic-programming programs contain many repeated instructions, or clones not found in their random counterparts. However, studying random programs helps reveal the fundamental nature of coding. For example, I proved that the fitness of lossless functions (such as in reversible and quantum computing) follows a Gaussian distribution. Also if inputs are unprotected, traditional computing quickly loses information. Loss of information provides a theoretical justification for common heuristics (such as write-protecting inputs). Meanwhile, information theory (in particular Shannon entropy) can be used as part of the fitness calculation a programmer would use to guide artificial evolution.

W. B. Langdon, London

Reference

1. Langdon, W.B. and Poli, R. Mapping non-conventional extensions of genetic programming. *Natural Computing* 7, 1 (Mar. 2008), 21–43.

Guard Against Innovation for Its Own Sake

Peter J. Denning's Viewpoint "The Idea Idea" (Mar. 2012) resonated with me due to discussions I had at Hewlett-Packard on how best to lead process improvement and innovation. New practices capable of delivering at the bleeding edge must be scouted and deployed/replicated. It is usually only after such an effort that a practice gains a theoretical basis, a two-step sequence that goes like this:

- ▶ It works (replicate/deploy/leverage); and
- ▶ This is why it works (institutionalize/educate).

During my stint, 1997–2000, as a member of the Software Engineering Process Group (SEPG) at Hewlett-Packard India Software Operations facilitating process improvement across all business lines, SEPG and line management concluded the best ideas originate as new practices in working teams. Teams were therefore encouraged to submit their own best practices to help identify such ideas, which were then analyzed for soundness and selected for deployment and replication. The teams were encouraged to maintain documentation using Deming/Shewart's Plan, Do, Check, Act (PDCA) at the project level and the Initiating Diagnosing Establishing Acting Leveraging (IDEAL) model at the SEPG level so they could quantify and demonstrate improvement. (PDCA and IDEAL both overlap the Prime Innovation Pattern outlined by Denning.) Selected best practices were then engineered as repeatable processes and institutionalized through associated training and tools. What we wished to guard against was innovation for its own sake.

Ideas generated by research teams often entail too much risk to deploy as-is in industry without first undergoing feasibility studies. For one thing, they often do not scale well, as when, say, a toy programming language is used to make a (perhaps valid) point in a research paper. For another, they may be solutions looking for a problem or inventions without a context—yet.

Mallika Chellappa, Bangalore, India

Jahromi Partially Restored

With two letters to the editor—"Reinstate Jahromi" (Jan. 2012) and "Justice for Jahromi" (Nov. 2011)—I wrote that Masaud Jahromi, a professor and chair, telecommunications engineering, at the University of Ahlia in Bahrain, had been arrested and imprisoned in April 2011 for speaking at a rally in the Kingdom of Bahrain,

in clear violation of the International Covenant on Civil and Political Rights to which Bahrain has acceded.

When he was released on bail in September 2011, a Bahraini colleague brought to his attention my January letter. Jahromi himself then took the initiative to send me an email message explaining what had happened. He also said how much he appreciated *Communications* for publicizing his situation. Following several delays, a trial was held January 19, 2012. Though the Bahrain Public Prosecutor recommended the charges be dropped, as they related to “freedom of speech,” consistent with the Bahrain Independent Commission of Inquiry, Jahromi was convicted and sentenced to four months in prison and ordered to pay a fine of approximately \$1,400. As he had already been imprisoned for five months, the court simultaneously suspended the four months. However, Jahromi had also been suspended from his university positions without pay from the date of his arrest, and the university did not reinstate him following the January 19 court ruling.

On February 20, 2012, Jahromi sent me another email message saying he had been reinstated in the university but not to his old positions. On March 20, 2012, he sent yet another email message, and then a clarification April 10, 2012, saying he had at last been reinstated to his old positions by the university. However, the government had not returned his passport or ID card, which it took from him April 14, 2011. Hence, he is not allowed to travel or attend scientific conferences outside Bahrain. He has since appealed and will continue to do so until all his rights are restored.

In his last email message, he thanked those who had supported him, including the Committee of Concerned Scientists (I am Vice-Chair, Computer Science), Scholars at Risk, the Committee on Academic Freedom in the Middle East and North Africa of the Middle East Studies Association, and Islamic Human Rights (London).

Publicity due to *Communications* and other organizations may have played a crucial role, helping pressure the government of Bahrain to abide by its commitment to human rights. *Communications* has always played an important role in publishing scientific

results but also by giving worldwide attention when scientific freedom and human rights of computer professionals are violated or at risk. While human rights remain a problem in Bahrain, it is uplifting to know that Jahromi has had at least some of his restored. We now look forward to the time when all are restored.

Jack Minker, College Park, MD

Survival and Computation

Daniel Reed’s blog (Sept. 2, 2011) and Douglas Robertson’s related letter “Insight, Not Numbers” (Apr. 2012) speculated on why we compute, suggesting two noble motivations: “know and understand” and “insight.” Robertson also added interesting comments regarding algorithmic information theory. However, both authors seemed to take a purely philosophical or research perspective, ignoring the large number of real-world corporate examples in which the primary motivation for computing is that many businesses would otherwise be unable to deliver services and products to their customers or manage, organize, store, or access in a timely fashion the ever-increasing data needed to run a large enterprise, especially one for which “information” is a key part of its products.

Reed’s description of “the exhilaration when the idea takes shape in code, then begins to execute” is something that first attracted me to computing and I have always regarded it as a “perk” of the profession. However, it was always the need to solve utilitarian problems to improve the corporate ability to process data and support customers that justified my paycheck.

In real-world corporate computing, the idea of information conservation is not a limiting factor for computation, as one does not deal with a closed system. Every second of every day new data pours in from customers and corporate processes alike, accumulating in large databases. The challenge is to determine what data is no longer useful and how and when to discard it.

Joel C. Ewing, Bentonville, AR

The String at t in C

Something seemed wrong in the following code portion of the letter by

Paul W. Abrahams “Still Paying for a C Mistake” (Apr. 2012):

```
while (*s++ = *t++);
copies the string at s to the string at t.
Given that assignments happen from
right to left, would it not have made
more sense to say it copies the string
at t to the string at s? That is, would a
simple x=y assign y to x?
```

Robert Wilkens, Levittown, NY

Author’s Response:

Wilkens is right. I should have said the code copies the string at t to the string at s. It was a slip-up, though, fortunately, the intent and the repair are obvious to anyone who knows C.

Paul W. Abrahams, Deerfield, MA

The Post Office Deficit and Technology

Though Michael A. Cusumano offered interesting ideas as to new postal services in his Viewpoint “Can Services and Platform Thinking Help the U.S. Postal Service?” (Apr. 2012), he failed to explore his reference to the “annual obligations for retiree health benefits” at the depth it deserves.

The Postal Accountability and Enhancement Act of 2006 (PAEA) requires the USPS to pre-fund retiree benefits for employees who have not been hired yet—and, with the 75-year pre-fund mandate, possibly even for those not even born yet. By requiring this transfer of funds from the USPS to the U.S. government general fund, the USPS is required to make the U.S. budget deficit look smaller at the cost of an artificial deficit in its own budget. Consumer advocate Ralph Nader once said that if not for PAEA, the USPS would have a surplus of at least \$1.5 billion.¹

Though computer science is naturally drawn to technological solutions, legal and social pressure is sometimes a more appropriate place to look for the source (and solution) of a problem.

John J. Deltuvia, Jr., Jackson, NJ

Reference

1. Jilani, Z. A Manufactured ‘Crisis’: Congress Can Let the Post Office Save Itself without Mass Layoffs or Service Reductions. ThinkProgress, Center for American Progress Action Fund (Sept. 28, 2011); <http://thinkprogress.org/economy/2011/09/28/330524/postal-non-crisis-post-office-save-itself/>

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.

© 2012 ACM 0001-0782/12/06 \$10.00

ACM, Advancing Computing as a Science and a Profession



Dear Colleague,

The power of computing technology continues to drive innovation to all corners of the globe, bringing with it opportunities for economic development and job growth. ACM is ideally positioned to help computing professionals worldwide stay competitive in this dynamic community.

ACM provides invaluable member benefits to help you advance your career and achieve success in your chosen specialty. Our international presence continues to expand and we have extended our online resources to serve needs that span all generations of computing practitioners, educators, researchers, and students.

ACM conferences, publications, educational efforts, recognition programs, digital resources, and diversity initiatives are defining the computing profession and empowering the computing professional.

This year we are launching Tech Packs, integrated learning packages on current technical topics created and reviewed by expert ACM members. The Tech Pack core is an annotated bibliography of resources from the renowned ACM Digital Library – articles from journals, magazines, conference proceedings, Special Interest Group newsletters, videos, etc. – and selections from our many online books and courses, as well as non-ACM resources where appropriate.

BY BECOMING AN ACM MEMBER YOU RECEIVE:

Timely access to relevant information

Communications of the ACM magazine • ACM Tech Packs • *TechNews* email digest • Technical Interest Alerts and ACM Bulletins • ACM journals and magazines at member rates • full access to the *acmqueue* website for practitioners • ACM SIG conference discounts • the optional ACM Digital Library

Resources that will enhance your career and follow you to new positions

Career & Job Center • The Learning Center • online books from Safari® featuring O'Reilly and Books24x7® • online courses in multiple languages • virtual labs • e-mentoring services • *CareerNews* email digest • access to ACM's 36 Special Interest Groups • an acm.org email forwarding address with spam filtering

ACM's worldwide network of more than 100,000 members ranges from students to seasoned professionals and includes many renowned leaders in the field. ACM members get access to this network and the advantages that come from their expertise to keep you at the forefront of the technology world.

Please take a moment to consider the value of an ACM membership for your career and your future in the dynamic computing profession.

Sincerely,

A handwritten signature in dark ink that reads "Alain Chesnais". The signature is written in a cursive, flowing style.

Alain Chesnais
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: AD10

You can join ACM in several easy ways:

Online

<http://www.acm.org/join>

Phone

+1-800-342-6626 (US & Canada)
+1-212-626-0500 (Global)

Fax

+1-212-944-1318

Or, complete this application and return with payment via postal mail

Special rates for residents of developing countries:

<http://www.acm.org/membership/L2-3/>

Special rates for members of sister societies:

<http://www.acm.org/membership/dues.html>

Please print clearly

Name _____

Address _____

City _____

State/Province _____

Postal code/Zip _____

Country _____

E-mail address _____

Area code & Daytime phone _____

Fax _____

Member number, if applicable _____

Purposes of ACM

ACM is dedicated to:

- 1) advancing the art, science, engineering, and application of information technology
- 2) fostering the open interchange of information to serve both professionals and the public
- 3) promoting the highest professional and ethics standards

I agree with the Purposes of ACM:

Signature _____

ACM Code of Ethics:

<http://www.acm.org/serving/ethics.html>

choose one membership option:

PROFESSIONAL MEMBERSHIP:

- ☐ ACM Professional Membership: \$99 USD
- ☐ ACM Professional Membership plus the ACM Digital Library: \$198 USD (\$99 dues + \$99 DL)
- ☐ ACM Digital Library: \$99 USD (must be an ACM member)

STUDENT MEMBERSHIP:

- ☐ ACM Student Membership: \$19 USD
- ☐ ACM Student Membership plus the ACM Digital Library: \$42 USD
- ☐ ACM Student Membership PLUS Print CACM Magazine: \$42 USD
- ☐ ACM Student Membership w/Digital Library PLUS Print CACM Magazine: \$62 USD

All new ACM members will receive an
ACM membership card.

For more information, please visit us at www.acm.org

Professional membership dues include \$40 toward a subscription to *Communications of the ACM*. Student membership dues include \$15 toward a subscription to *XRDS*. Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

RETURN COMPLETED APPLICATION TO:

Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Questions? E-mail us at acmhelp@acm.org
Or call +1-800-342-6626 to speak to a live representative

Satisfaction Guaranteed!

payment:

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc. in US dollars or foreign currency at current exchange rate.

☐ Visa/MasterCard ☐ American Express ☐ Check/money order

☐ Professional Member Dues (\$99 or \$198) \$ _____

☐ ACM Digital Library (\$99) \$ _____

☐ Student Member Dues (\$19, \$42, or \$62) \$ _____

Total Amount Due \$ _____

Card # _____

Expiration date _____

Signature _____

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.



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

DOI:10.1145/2184319.2184322

<http://cacm.acm.org/blogs/blog-cacm>

Protecting Against Data Breaches; Living with Mistakes

Jason Hong writes about security breaches and offers a three-pronged approach. Greg Linden discusses the differences between computers and the human brain and their tolerance of errors.



Jason Hong
"Why Have There Been So Many Security Breaches Recently?"

<http://cacm.acm.org/blogs/blog-cacm/107800>
April 27, 2011 (updated April 30, 2012)

Just to recount, here are some of the more prominent breaches in the past 18 months:

- ▶ HBGary Federal, a beltway computer security firm, had all of its email stolen and made available on BitTorrent. This incident also raised a lot of questions about the ethics of the work it was being paid to do in the first place.

- ▶ Comodo Group had its systems breached, and several fake browser certificates were created along the way.

- ▶ Databases used to maintain RSA SecurID tokens were breached using a combination of a spear-phishing attack and a zero-day Flash exploit.

- ▶ The Epsilon mailing list service, which maintains mailing lists for

many corporations, had its databases hacked, quite possibly through a phishing attack.

- ▶ The PlayStation Network was hacked, with more than 65 million accounts compromised, including names, street addresses, email addresses, and purchase histories stolen. It is not clear from Sony's public statement whether credit card and password data was stolen. [Update: Credit card and password data was stolen, but Sony says the credit card info was protected by industry standard encryption.]

- ▶ In a self-referential loop, members of the Anonymous group released recordings of a conference call between the FBI and Scotland Yard discussing Anonymous.

- ▶ Global Payments Inc. had a data breach that may involve up to 1.5 million credit and debit card accounts.

The scary part is, it is only going to get worse.

There has been a rise in the sophistication of attacks over the past years.

These attackers, ranging from script kiddies to criminals to state-sponsored cyberwarriors, have been all too successful in breaking into online systems.

There are two interesting observations here. The first is that many of these attacks have shifted from just directly attacking a computer system, an attack model that computer security specialists are somewhat good at defending against, toward also exploiting the human vulnerabilities in these systems.

By human vulnerabilities, I mean all of the misunderstandings, laxness, and cognitive and social biases that arise with the people who use computer systems. The list of human vulnerabilities here are numerous: poor interfaces that are difficult to understand, interfaces that are easy to misconfigure, guessable passwords, reused passwords, tricking people into installing malware, tricking people into opening up documents (which might use zero-day exploits), and on, and on, and on.

These human vulnerabilities are clearly a major weakness, but are also a puzzling blind spot from both a research perspective and an industry perspective. Well, actually, it is not that puzzling. People are messy. We all have a wide range of experiences, knowledge, and motivations, and so it is natural and tempting to just buy that "magic black box" that claims to solve all your security problems and avoid having to actually deal with the messiness. Just go to any industry conference, like RSA, and you will see what I

mean—vendors selling row after row of magic black boxes. And by doing so, computer professionals can rest comfortably in the conventional wisdom that users are just stupid, users are to blame, and not actually force ourselves to adapt to smarter attackers.

The other observation is that none of these hacks are actually new or innovative (and, no, most zero-day attacks are not that innovative unless you count buffer-overflow attacks as innovative). We do not know yet about the Sony PlayStation Network attack, but I am willing to bet it did not use any advanced techniques either. [Update: Sony has described the attack as an “external intrusion,” but appears to have provided no other details.]

Instead, what is fascinating here is the sophistication of the execution. The attackers are more patient, adept at using a wide range of tools, and very capable of progressively exploiting smaller vulnerabilities into larger ones. It is analogous to the transition in martial arts from single schools of thought to cross-training and mixed-martial arts. These recent attacks have been highly creative, flexible to the situation, and make full use of a combination of techniques.

For example, in the Epsilon case, it looks like the attackers were using spear-phishing attacks for several months, trying to bait low-level employees at several mailing list companies. While there are few details, there are several interesting questions. How did the attackers know which employees to target? Did the attackers know how to format the spear-phishing email so it looked appropriate? Did they know when to send the malware for maximum effect? I suspect the answer to all these questions is yes.

The HBGary case is also an impressive case study. It started with a SQL injection on the company’s Web site, which led to stealing the password file, running a standard password crack, exploiting password reuse to remotely wipe the (now former) CEO’s iPad and access his Twitter account, and a spear-phishing attack to get the password for another site. Again, all standard techniques, but just well executed.

These kinds of security breaches are a long-term problem that we will be struggling with for the next 10–20

years. The only effective strategy I see is to take a three-pronged approach. First, get research and industry to develop new best practices, create better tools, and have better training of software developers. Second, complement these technical approaches with a stronger legal structure that can properly incentivize companies to take stronger measures in protecting customer data. Finally, develop new ways of actually addressing the human vulnerabilities, in the form of simpler and better-designed user interfaces, more research to gain a deeper understanding of human biases and social influences in decision-making, and better ways of motivating and training people so they are effective in the face of these ongoing attacks.

Reader Comments

The single common thread to the above (and most) breaches is the failure to architecturally separate data. Everything in a lump with access rules is more convenient than designing systems that put what is needed where it is needed. Significant breaches come when someone (or system) has access to more data than they need. Rules can be broken, architectural separation is harder—you have to hop domains, not just abuse the rules. It is still possible, it is just a lot more work.

—James Byrd



Greg Linden
“Embracing Noise
or Why Computer
Scientists Should Stop
Worrying and Learn
to Love the Errors”

<http://cacm.acm.org/blogs/blog-cacm/107864>

April 28, 2011

When people talk about differences between computers and the brain, they often emphasize the massively parallel processing of the brain. There are hundreds of billions of neurons in the brain firing at a relatively slow rate compared to just a few but very fast processors in a modern computer.

There is another very important difference between the brain and computers: tolerance of error. Computers are derailed if even a couple bits are flipped during a computation. Computers expect everything to be perfect. Error recovery is an afterthought, a pro-

tection against what is expected to be a rare event of a data read error, and perfection is assumed.

By contrast, our wetware embraces error. The brain is a cacophony, a battle of competing patterns, from which understanding emerges. It is a system built on noise, one that expects constant error and recovery from error. The brain abandons precision and thrives on approximation.

There are examples of embracing noise and error in computing. Machine translation has made great strides by building likely translation rules from patterns in large amounts of data, then expecting, welcoming, and correcting frequent errors in the translated text. Recommender systems make no assumptions that they are perfect, but only that they might be helpful, and only expect to be judged by how useful they are compared to alternatives for lists of movies or books you might like. Web search spreads a query across hundreds or thousands of machines and simply ignores those that fail to respond quickly, relying on getting a good enough answer out quickly rather than getting a complete answer out slowly.

Often we find estimates are good enough. Recent work on data mining in large clusters, such as Hadoop, has found orders of magnitude speed-ups can be gained if only an approximate answer—such as +1% with 95% probability—is required. Some computer security researchers are embracing the idea that classifying what is malware is not black or white, but a gray area, and are starting to use techniques, such as slowing suspicious processes, rather than expecting to be able to identify and kill all wrongdoers.

Of the lessons we take from biological systems, one should be that wetware is sloppy. And that is okay. Precision is not required in everything or even most things. Failures are best handled by expecting them all the time, not treating them as exceptions. We should expect errors and handle them routinely. We should stop worrying and learn to love the noise. 

Jason Hong is an associate professor of computer science at Carnegie Mellon University. Greg Linden is the founder of Geeky Ventures.

© 2012 ACM 0001-0782/12/06 \$10.00



DOI:10.1145/2184319.2184323

Scott E. Delman

Don't Throw the Baby Out with the Bathwater

NEARLY EVERY DAY I receive an email message that includes a link to at least one new article related to the Open Access movement. Some of these articles are written by members of the scientific community, some are written by professional journalists who at times combine fact with fiction to tell a compelling story, and some are statements of policies being established in academia in relation to faculty members, posting policies, and scholarly publishers—of which ACM is one. Some of these articles are well written and make perfectly good sense to me, while others not so much. I read every one of these articles because I take the issue of open access very seriously and I am a firm believer in the ideal of open access as well as the viability of certain OA models put forth by the community. But, above all else, I want to understand the different perspectives on the issue to become more informed, so that I can help contribute to the solution and not the problem.

I am also a scholarly publisher and a member of the ACM staff. I understand firsthand what is involved in building a publication like *Communications* or a publication platform like the ACM Digital Library, the complexity of the infrastructure that exists, what long-term investments are necessary to develop the ACM brand, where those investments come from, and all of the good that ACM does for the computer science community as a forum for the exchange of ideas and as a sustainable central repository for knowledge for the field. When articles are written that minimize or dismiss the value that scholarly publishers

bring to the scholarly communication process or suggest the scientific community would be better off simply posting articles freely on the Web without publishers involved in the process, this shows a clear lack of understanding and appreciation for how peer-reviewed information is created, distributed, and consumed. These articles also strike me as pure idealism without the input of data, facts, and practicality.

Much of what ACM does relies on the existing subscription model and while this may change over time, there are many risks associated with a transition to a new model and especially to a model that is to date not proven

Revenues generated by ACM publications are pumped back into the community and the community itself benefits, including faculty members and students at the same institutions working to change the system they helped create.

to be sustainable for the computer science community. While I do believe that many of the so-called “Gold OA” models (models typically based on grant funding instead of library funding) show a great deal of promise, there is no clear path of transition for the hundreds of small- to mid-sized scholarly publishers and societies serving millions of students, researchers, and practitioners around the world who have come to rely so heavily on the publications and services provided by those organizations.

Creating this clear path takes time and must be a collaboration involving the key players in the publication value chain (authors, end users, universities, librarians, and publishers), not the result of a one-sided mandate by the largest and most powerful academic institutions to boycott all subscription-based publishers because some of them have benefited more than others. To be certain, some of these organizations have benefited financially from the existing model more than others, but ACM is not one of them. Revenues generated by our publications are pumped back into the community and the community itself benefits, including faculty members and students at those same institutions working to change the system they helped create.

Change can be good and I am a firm believer that ACM, its members, and the computer science community can benefit from implementing open access in a well thought out and sustainable way, but this will take time and require collaboration and patience, not mandates and threats.

Scott E. Delman, PUBLISHER

Analyzing Medical Data

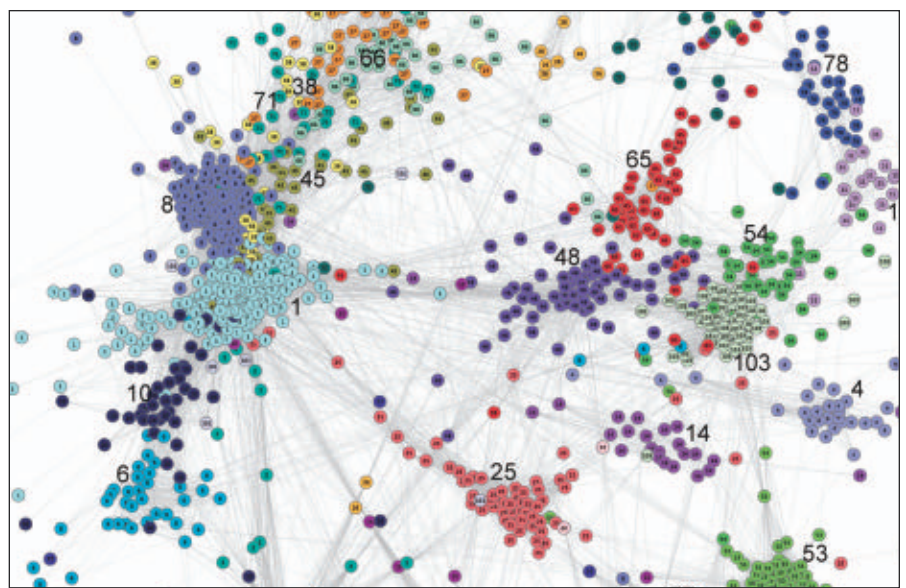
Electronic patient records contain a treasure trove of data, and researchers are using natural language processing technology to mine the structured data and free text.

ONE OF THE technological ironies in health care is the disconnect between the advanced state of clinical technology, such as non-confining open imaging technologies, the variety of smartphone health apps, surgical robots, and the backward state of electronic patient records.

Until the passage of the Health Information Technology for Economic and Clinical Health Act in the U.S. three years ago, only an estimated 20% of U.S.-based physicians used electronic patient records. That percentage is rapidly increasing due to the law's financial incentives, but the new attention is also awakening researchers to the limitations of the structured data that is often exchanged between physicians, insurance companies, and organizations interested in compiling and repurposing discrete patient records to conduct population-based medical research.

Søren Brunak, director of the Center for Biological Sequence Analysis at the Technical University of Denmark, is one of these researchers who is using natural language processing (NLP) technology to mine not only structured data such as standardized disease codes, but also free text.

In a recent paper, "Using Electronic Patient Records to Discover Disease



A patient cohort network in Søren Brunak et al.'s *Computational Biology* paper: Nodes represent patients, edges are correlations between patients, and node color denotes cluster membership.

Correlations and Stratify Patient Cohorts," published in *Computational Biology*, Brunak and his colleagues showed that combining free-text analysis with structured disease definition codes can help researchers discover unexpected connections between diseases, such as a link between migraine headaches and alopecia (hair loss), but also the lack of expected comorbidities, such as those between diseases coded as "mental and be-

havioral disorders" and those coded under the "drug abuse, liver disease, HIV" clusters.

"Not only our paper, but also work from many other groups, show that text mining and mining of unstructured data can lead to very interesting results," says Brunak, who laments the condition of many patient records. "My priority is that the patient record becomes as information-rich as possible. That maximizes the data min-

ing opportunities, and the success of Google and other technologies that successfully analyze unstructured texts are examples and evidence that the old-fashioned ideas that only structure is the way to go forth is wrong.”

A Lack of Flexibility

The medical community has long agreed on the basic data fields for the description of diseases. This schema, the International Classification of Diseases (ICD), uses a hierarchical system to pinpoint specific medical conditions. Two versions of ICD are concurrently in use: ICD-9, which is reaching the end of its viable usefulness, and ICD-10, its successor. ICD-9’s 13,000 available diagnostic codes use three to five alphanumeric characters per disease or medical condition but do not distinguish between injuries to the left or right sides, whereas the more granular ICD-10 diagnoses comprise 68,000 codes and include laterality.

ICD is currently the lingua franca of disease classification upon which many researchers rely to supply a structured data element when analyzing comorbidities. However, leading bioinformatics researchers, including Brunak and Christopher Chute, M.D., professor of bioinformatics at the Mayo Clinic and chairman of the World Health Organization’s steering group responsible for the next ICD iteration, ICD-11, say the terminology is not really suitable for serving as a baseline.

“ICD is not intended to be an exhaustive catalog of clinical concepts

that may be encountered or enumerated,” Chute says. “It is a high-level aggregation of diseases, and that is its purpose. It was originally a public health response, and it is also used for reimbursement, but to treat it as a catalog of disease is incorrect.”

Brunak concurs. “There is no doubt ICD-10 is not the best text-mining vocabulary for spotting things in records. It’s very difficult to say something general about which ontology or which system is best because there’s a different signal-to-noise ratio in different types of records, and one type does not fit all.”

Yet without some type of universally accepted thesaurus, Chute says the prospect for significant advances in the natural language processing of clinical data is uncertain. Chute says the ICD-11 working group has been operating in concert with the International Health Terminology Standards Organization, which oversees the development of the Systematized Nomenclature of Medicine (SNOMED), a hierarchical semantic network of more than 300,000 medical concepts and their relationships. The semantic nature of SNOMED allows for more than seven million relationships descending from the top three hierarchical classifications of “finding,” “disease,” and “procedure.”

Chute says the ICD-11 and SNOMED terms will be harmonized, yielding interoperability between the two dominant clinical data schema. “That’s what we ultimately need, and even SNOMED in its current incarnation doesn’t fully capture the spectrum of clinical concepts you’d like to catalog for natural language processing.”

S. Trent Rosenbloom, M.D., associate professor of biomedical informatics at Vanderbilt University, says the lack of an overriding priority in developing health-care data formats—a patient record must satisfy clinicians’ needs to describe a course of treatment for their patient as well as provide documentation for legal safeguards, plus serve as a billing document that satisfies insurance companies—has led to the difficulty in aligning analytical capabilities. In “Data From Clinical Notes: A Perspective on the Tension Between Structure and Flexible Documentation,” which was recently

published in *Journal of the American Medical Informatics Association*, Rosenbloom and colleagues note that “the flexibility of a computer-based documentation method to allow healthcare providers freedom and ensure accuracy can directly conflict with a desire to produce structured data to support reuse of the information in [electronic health record] systems.”

Rosenbloom’s colleague, Joshua C. Denny, M.D., assistant professor of biomedical informatics at Vanderbilt, says the holy grail of analyzing extremely large volumes of health records to deliver “personalized medicine” to a single patient will depend not only on perfecting NLP capabilities confined within clinical walls, but also in expanding the concept of what belongs in a medical record, and who should be authorized to provide data. For example, a 50-year-old man who runs every day may paradoxically have high levels of both good high-density lipoprotein (HDL) cholesterol, which helps to clear the arteries—high amounts of exercise can elevate it—and of bad low-density lipoprotein (LDL) cholesterol, which is a risk factor for coronary disease. Following conventional medical wisdom, the man’s physician may want to prescribe medication to lower the LDL levels without actually knowing if it is necessary because there is not a current capability to pull population-wide data on such a relatively small cohort. Patient-curated data may be able to help discern what, if any, treatment would be appropriate for such patients.

“Studies can’t get that population well,” Denny says. “That sort of patient has an interest in this and is going to document that. Maybe there is detail there that allows us to mine, on the back end, the health status of the small percentage of people who run every day and have this HDL and this LDL level. The number of combinations of variables such as that is huge. You’re never going to get that amount of data in a study. For that you need population-based data.”

Perfecting NLP Capabilities

While standards groups and policy committees address the issues surrounding format harmonization, as well as the legal issues surrounding pa-

“My priority is that the patient record becomes as information-rich as possible,” says Søren Brunak. “That maximizes the data mining opportunities.”

tient privacy and who should be able to contribute to health records, bioinformaticists are researching how to perfect NLP capabilities.

One NLP issue, says Denny, is that the majority of natural language processing technology has been created for dictated speech, but electronic health records are often typed.

“Typed documents are more ambiguous because there are more abbreviations and acronyms, and the documents usually contain more misspellings,” says Denny. “You don’t describe things in as much detail, so that hinders a little bit the richness of what can be done with natural language processing. Maybe speech-recognition technology will move forward fast enough that we’ll move toward documents that look like dictated documents, and the problems won’t be as big.”

William Cohen, research professor of machine learning at Carnegie Mellon University, is currently researching the creation of a domain-specific version of the Never Ending Language Learner (NELL), refining its general-purpose Web-crawling algorithm for use in published biomedical literature called BioNELL. While BioNELL is focused on published biomedical literature rather than on-the-fly clinical notes, Cohen says such domain-specific, rank-and-learn principles might be useful for organizations’ in-house lexicons while wider standards are being crafted.

“The exciting thing I find about

One NLP issue, says Joshua C. Denny, is that the majority of NLP technology has been created for dictated speech, but electronic health records are often typed.

BioNELL is taking the existing structured databases and using them to kick-start an NLP system,” Cohen says. “Symmetrically, you might want to take a natural language corpus and use that to kick-start understanding a sensor database.”

Brunak says that while the attention paid to his group’s *Computational Biology* paper about unlikely comorbidities was interesting, the broader message should be that examining more hitherto machine-unreadable data could alter the practice of medicine.

“Real patients have more than one disease,” says Brunak, “and the patient records give us an opportunity to discover comorbidities and disease correlations—not only those that co-

occur but also disease trajectories, that is, those that come before others. The message should be that we can start disease profiles of real patients instead of doing what medicine has done for hundreds of years, studying people disease by disease.” □

Further Reading

Barrett, N. and Weber-Jahnke, J.H. Applying natural language processing toolkits to electronic health records—an experience report, *Studies in Health Technology and Informatics*, 143, 2009.

Denny, J.C., et al.

PheWAS: demonstrating the feasibility of a phenome-wide scan to discover gene-disease associations, *Bioinformatics* 26, 9, May 1, 2010.

Murff, H.J., et al.

Automated identification of postoperative complications within an electronic medical record using natural language processing, *Journal of the American Medical Association* 306, 8, August 24, 2011.

Rosenbloom S.T., Stead W.W., Giuse, D., Lorenzi N.M., Brown S.H., and Johnson K.B. Generating clinical notes for electronic health record systems, *Applied Clinical Informatics* 1, 3, Jan. 1, 2010.

Savova, G.K., et al.

Discovering peripheral arterial disease cases from radiology notes using natural language processing, *Proceedings of the Annual symposium of the American Medical Informatics Association*, Washington, D.C., Nov. 13–17, 2010.

Gregory Goth is an Oakville, CT-based writer who specializes in science and technology.

© 2012 ACM 0001-0782/12/06 \$10.00

Milestones

Computer Science Awards

The American Academy of Arts and Sciences, John Simon Guggenheim Memorial Foundation, National Science Foundation, and the Franklin Institute recently honored leading computer scientists.

AMERICAN ACADEMY MEMBERS

The American Academy of Arts and Sciences named seven new members in the section of computer sciences (including artificial intelligence and information technologies). They are Arvind, Massachusetts Institute of Technology; Robert

P. Colwell, R&E Colwell & Associates, Inc.; Irene Greif, IBM; M. Frans Kaashoek, Massachusetts Institute of Technology; Michael Kearns, University of Pennsylvania; Judea Pearl, University of California, Los Angeles; and Jeffrey D. Ullman, Stanford University.

GUGGENHEIM FELLOW

John Simon Guggenheim Memorial Foundation selected Susan Landau, a visiting scholar at Harvard University, as a 2012 Fellow in the category of computer science.

ALAN T. WATERMAN AWARD

The National Science Foundation (NSF) named Robert Wood, an associate professor in Harvard University’s School of Engineering and Applied Sciences, and Scott Aaronson, an associate professor of electrical engineering and computer science at the Massachusetts Institute of Technology, to receive the 2012 Alan T. Waterman Award, which “recognizes an outstanding researcher under the age of 35 in any field of science or engineering NSF supports.”

BENJAMIN FRANKLIN MEDAL

The Franklin Institute awarded the Benjamin Franklin Medal in Computer and Cognitive Science to Vladimir Vapnik, who is a professor of computer science and statistics at Royal Holloway, University of London, and holds a professorship in computer science at Columbia University. Vapnik was recognized for “his fundamental contributions to our understanding of machine learning” and “his invention of widely used machine learning techniques.”

—Jack Rosenberger

Smarter Photography

Improvements in camera hardware, image processing, camera-photographer interfaces, and image viewing are advancing the state of the art in digital photography.

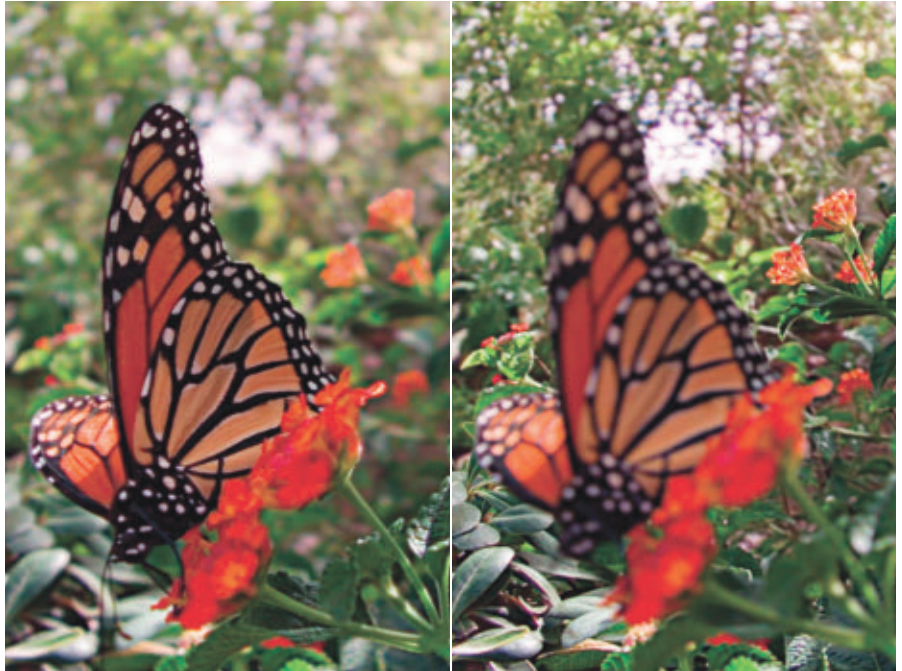
FOR MOST OF its first 10 years, digital photography largely mimicked film photography. Of course, Photoshop and other software made image processing easier, but digital photographers worked in much the same way as before and were often beset by similar problems. Now, however, digital photography is advancing to a fundamentally new level of capability, aided by technology that would have been unimaginable a decade ago.

For example, in the days of film photography, a photographer finding an unwanted object in a picture had few options. He or she might have cropped it out, “burned” the unwanted object in the darkroom to make it less conspicuous, or tried to take a better photo the next time. Soon after 2000, however, a digital photographer possessed a new option with photo-editing software’s clone tool, which could copy over the unwanted object with a more desirable part of the image.

Going one step further, computer scientists at Carnegie Mellon University have demonstrated prototype software that searches millions of photos on the Internet, locates one similar to yours, and downloads an image patch to paste into your photo. “The whole idea seems a little ludicrous,” says Alexei Efros, leader of the team that invented the technique. “It doesn’t seem like this should work.”

Advancements in computational photography are dramatically increasing both the ease of taking pictures and the quality of those pictures. The advances can be classified as those occurring in camera hardware, in post-processing software, in prompts and automated assists to the user taking a picture, and in powerful tools for users retrieving and viewing digital images.

Lytro Inc.’s recently introduced light field camera uses a unique new



Lytro Inc.’s Light Field Engine software travels with an image, thereby enabling users to selectively focus on any part of an image by clicking on it.

sensor and software to, in essence, capture everything in sharp focus. The magic is that, unlike a conventional sensor that records only the color and intensity of light, the Lytro’s image sensor also records the direction, or angle, at which a ray of light strikes it. This additional data enables algorithms in the camera’s software to reconstruct a sharp view of any part of the photograph. User software, called the Light Field Engine, travels with the image so viewers can selectively focus by clicking on any part of the image.

“Being able to snap the shutter without picking a thing of interest, or focal point, and being able to refocus it later has huge potential,” says Richard Koci-Hernandez, assistant professor at the University of California, Berkeley’s Graduate School of Journalism and a beta user of the Lytro camera. Koci-Hernandez recalls his days as a film photographer at the *San Jose Mercury News*, when he often had to make split-

second decisions about where in the viewfinder to focus. This is not necessary with the Lytro camera, he notes.

While the Lytro camera greatly expands the in-focus depth of field of an image, a similar concept expands the dynamic range of exposure in a photo. Many cellphone cameras can now create these high dynamic range images by combining the best-exposed portions of multiple images, which are taken in split seconds without user awareness.

A different approach to capturing useful information with an image is taken by Shree Nayar, chairman of the computer science department at Columbia University and a pioneer in computational imaging, who combines algorithms and innovative optics to produce images with more than one billion pixels. Nayar and his students have developed computational cameras that can achieve splendidly pixel-rich images without the very large

and expensive lenses that such images would normally require. The difficulty in attaining such pixel-rich images lies not in the sensor as gigapixel sensors are available today. Rather, the difficulty is in the limits in resolving power, due to geometric aberrations, inherent in conventional optics.

Nayar's lenses, which are spherical in shape, produce coded output that is manipulated by algorithms to remove any aberrations. The most-advanced lens design has a number of smaller "relay lenses" positioned on the surface of a ball lens to focus small portions of the overall image onto megapixel sensors positioned in a hemispherical array around the lenses. Compared with other gigapixel cameras, this camera is small, less than 10 cm. in diameter, and relatively inexpensive to produce, Nayar says.

Nayar's work is partly funded by the U.S. Defense Advanced Research Projects Agency, and has obvious defense and security applications. "If I can recognize your face from a mile away, that's a game changer," says Nayar.

Post-Processing Software

Adobe Systems, manufacturer of the flagship Photoshop image-processing software, has conducted substantial research in plenoptics, in which multiple images or perspectives are generated with the single push of the shutter button and then combined. Plenoptics enables the Lytro light field camera's variable focusing, high dynamic range's variable brightness or exposure, and variable perspective. "All of these can be done after the fact in software," says Bryan O'Neil Hughes, senior product manager for Photoshop.

Although Adobe does not sell cameras, it developed a lens, similar to an insect eye with multiple lenses, which consists of hundreds of glued-together micro-lenses, each of which can sample incoming light from a different perspective. With an array of new algorithms and a 60 megapixel sensor, the Adobe lens was able to achieve variable focus after the fact, as the Lytro camera does.

Adobe has also developed experimental, computationally intensive software that can provide "deblurring" capabilities by working on traditional images. Part of Adobe's trick is to distin-

guish blur caused by camera movement from blur caused by subject movement. Unlike the plenoptic research involving multiple images, Adobe's deblurring works with a single image.

Meanwhile, scientists at Microsoft Research have used both hardware and software to fix the blurring caused by camera movement. Their experimental system, which operates in-camera, uses inexpensive inertial sensors, like those in many cell phones, and to estimate a point-spread function (PSF) from a single image capture. The PSF, essentially a representation of the amount and direction of blur, goes through a process of deconvolution that creates a blur-free image.

Software alone can produce some deblurring, as Adobe has demonstrated, but adding the inertial data helps the software do a better job, says Richard Szeliski, director of the Interactive Visual Media Group at Microsoft Research. "This is multisensor fusion—the inertial sensors plus the visual [optical] sensors."

Szeliski's method works with a single image, but he says a trend for future cameras will be very rapidly capturing multiple images of the same scene, then merging them in different ways to achieve various objectives. That will require very short exposures, which are individually noisy but can be scrubbed of noise through averaging. "All this requires faster silicon and faster digital signal processors," he says.

Helping Photographers

Sensors, insect-eye lenses, lightning-fast exposures, and smart algorithms have provided significant advantages, but why not help photographers avoid some of the age-old pitfalls in shooting? That is the approach taken by computer scientist Stephen Brewster and colleagues at the University of Glasgow. Their work centers on the cameras in Android smartphones because they are more advanced than conventional digital cameras in many ways and because of their open architecture, fast processors, and increasing popularity.

One of Brewster's experimental cameras uses a smartphone's accelerometer to warn the photographer that the camera is moving too much to get a sharp picture. It does this by displaying a warning in the camera's image dis-

play or, for people who are holding the phone away from their head, by an audible or vibrational cue. And for users who do not understand the luminance histogram that appears on the rear of many digital cameras, Brewster's team has created a camera that emits a low tone when an image is underexposed and a high tone when it is overexposed. Another set of techniques use the face-detection software built into Android smartphones to help the photographer improve composition.

For the user who may regard these multiple warnings as information overload, the University of Glasgow researchers have taken all of the camera's measurements about exposure, subject, camera movement, and composition and merged them into a summary Picture Quality Index. The Picture Quality Index is updated once per second when the shutter button is half-depressed and helpfully displays its evaluation—red for poor, amber for fair, and green for good—in the viewfinder image.

Image Viewing and Retrieval

The Lytro light field camera is about getting objects in focus, but it is also about giving photographers multiple choices long after the picture is taken. Microsoft and others are taking that trend further, partly to overcome limitations in current displays. The average computer monitor lacks the contrast ratio and resolution to properly view images with billions of pixels and a very large range of luminance.

Microsoft's HD View software allows smooth panning and zooming across gigapixel images, including panoramas. It also adjusts the dynamic range of the portion of the photo being viewed by the user to the much more limited luminance range of his or her monitor so that, for example, the viewer can see good detail when looking at a bright sky, but also when looking into the dark shadow under a tree, when both details appear in the same photograph.

Empowering users after a photo is taken is at the core of the research done by Carnegie Mellon's Alexei Efros. His remote cloning technique works surprisingly well because photographers tend to shoot similar things over and over, he says. But,



ACM's *interactions* magazine explores critical relationships between experiences, people, and technology, showcasing emerging innovations and industry leaders from around the world across important applications of design thinking and the broadening field of the interaction design. Our readers represent a growing community of practice that is of increasing and vital global importance.

interactions
<http://www.acm.org/subscribe>



Alexei Efros' remote image-cloning technique employs a relatively new approach to modeling in which simple machine-learning techniques are applied to huge databases.

more important, the remote cloning works because it employs a relatively new approach to modeling in which very simple machine-learning techniques are applied to huge databases. Large amounts of data can overcome weaknesses in algorithms, Efros says. "The standard view in computer science has been that the most important thing is the algorithm, then you have the representation, then you find some data," he says. "But it's actually the other way around: The most important thing is the data. Then it's the representation, and only then comes the algorithm."

Efros and his colleagues are applying this principle to a new technique that finds visually similar images even if they are quite different at the pixel level and are not effectively matched by conventional techniques. It uses a statistical technique called support vector machine to estimate the relative importance of different features in a query image. "Our approach shows good performance on a number of difficult cross-domain visual tasks by, for example, matching paintings or sketches to real photographs," he says.

The approach by Efros' team could create "a new age in visual expression," says Columbia's Shree Nayar. Existing image-editing tools use classical image processing techniques, he says, "but now you have the opportunity to use lots of data and machine learning."

Efros says his ultimate goal is to

create a "visual memex"—a model for linking images not by categories, such as car, person, or city, but by zeroing in on what is unusual or unique in an image. These visual characteristics would become, in effect, hyperlinks. In a YouTube video titled "Data-driven Visual Similarity for Cross-domain Image Matching," Efros downloads 200 images from Flickr based on a simple keyword search for "Medici Fountain Paris." From them he builds a "visual memex graph" whose nodes are images and parts of images and whose edges are various types of associations, such as visual similarity and context. He goes on to show how his algorithms use the graph to find far better matches to a test image of the fountain than traditional searching techniques. By zeroing in on relatively small but unique elements in his test image, the technique avoids the superficial but incorrect matches based on similar skies or foregrounds that occupy large portions of the pictures but which are irrelevant to the fountain.

Where might these type of techniques lead? "My dream is to make kind of a World Wide Web for visual data," says Efros. "That's a 10- to 15-year project." **C**

Further Reading

Cossairt, O., Miao, D., and Nayar, S. Gigapixel computational imaging, *IEEE International Conference on Computational Photography*, Pittsburgh, PA, April 8–10, 2011.

Kee, E., Paris, S., Chen, S., and Wang, J. Modeling and removing spatially-varying optical blur, *IEEE International Conference on Computational Photography*, Pittsburgh, PA, April 8–10, 2011.

Nack, J. Adobe demos refocusable images, <http://blogs.adobe.com/jnack/2010/09/adobe-demos-refocusable-images.html>, Sept. 25, 2010.

Srivastava, A., Malisiewicz, T., Gupta, A., and Efros, A. Data-driven visual similarity for cross-domain image matching, *Proceedings of the 2011 SIGGRAPH Asia Conference*, Hong Kong, Dec. 12–15, 2011.

Szeliski, R. *Computer Vision: Algorithms and Applications*, 2011. Springer, New York, NY, 2011.

Gary Anthes is a technology writer, editor, and photographer based in Arlington, VA.

© 2012 ACM 0001-0782/12/06 \$10.00

Data Mining Meets City Hall

Local and national governments are turning to open data to cut their costs, increase transparency and efficiency, and respond to the needs of citizens.

BY NOW, THE refrain is familiar to many New Yorkers: “In God we trust. Everyone else, bring data.” It is a phrase—and a mind-set—the city’s mayor, Michael Bloomberg, tirelessly promotes. Over the course of his 10-year term, Bloomberg has transformed a domain where statistical analyses are confined to opinion polls into one where data-driven transparency is hailed as a means of cutting costs, increasing government efficiency, and engaging often-jaded citizens.

The Bloomberg administration has tracked its performance through indicators like infant mortality rates, which have fallen by nearly 20% since Bloomberg took office in 2001, and responses to reported emergencies, which took a minute longer in 2011 than in 2001. It has also used predictive analytics to investigate potential fire risks. Locations flagged as high risk are investigated within 48 hours, and six months after the program’s implementation, fire department personnel and buildings inspectors found seriously hazardous



New York City’s chief digital officer Rachel Sterne, right, at the opening press breakfast during the first day of Internet Week NY, which occurred in June 2011.

Open data “democratizes the exchange of information and services,” says Rachel Sterne. “It empowers citizens to collaboratively create solutions.”

living conditions in more than 75% of their inspections, an increase of 400% over the previous two years. And last year the Bloomberg administration created NYC Digital, an office whose goal is to “realize New York City’s potential as the world’s leading digital city.” Led by the city’s first chief digital officer, Rachel Sterne, NYC Digital has organized hackathons, app competitions, and other initiatives to encourage the use of municipal data.

“It democratizes the exchange of information and services,” Sterne explained at the 2011 O’Reilly Strata Conference in New York. “It empowers citizens to collaboratively create solutions. It’s not just the consumption but the coproduction of government services and democracy.”

Increasingly, New York is not alone in its commitment to open data and transparency. North American cities

like Chicago, Portland, San Francisco, and Vancouver have begun to release their datasets. The U.S. government has also done so, as have the governments of Australia, Estonia, New Zealand, Norway, and the United Kingdom, along with global organizations like the United Nations and the World Bank. The catalysts for their change are varied, but recession-squeezed budgets often play a significant role.

“It wasn’t really an innovation-centric approach at first,” admits Dustin Haisler, the former Chief Information Officer for the City of Manor, TX, who won plaudits for his use of QR codes to distribute information on city projects and historic buildings. “We were boxed into a corner financially, and it grew out of a search for alternative ways to solve problems.”

To that end, many participants see open data as not just a way for govern-

ments to track their own progress, but as a way to enable citizens to work together to improve it.

Much of the excitement revolves around municipal initiatives. “Cities are where most people understand their government and interact with it,” says Jen Pahlka, the founder and executive director of Code for America, a nonprofit organization that connects developers with municipal governments through a year-long fellowship program, a “brigade” of volunteer professionals, and a soon-to-be-launched start-up incubator. Thus far, Code for America has leveraged public data to help citizens to keep up with legislation in Philadelphia, browse public school information in Boston, and locate retailers that accept food stamps in every city in the U.S.

Data Transparency

Organizations like Code for America are part of an increasingly vibrant ecosystem that works to promote data transparency. Typically, data advocates balance basic work to open new datasets and encourage the adoption of global standards with purpose-driven projects like creating apps and application programming interfaces (APIs). The Washington, D.C.-based Sunlight Foundation, for example, employs a team of investigative journalists to obtain data, such as information about political junkets and contributions, through Freedom of Information Act requests. It also houses a policy team that engages in direct lobbying, a team that organizes volunteers, and a lab whose 20-odd developers focus on making data more accessible.

Sunlight’s first API, launched in 2007, leveraged work by Joshua Tauberer—a software developer who, in 2004, launched GovTrack.us to aggregate federal legislative data—to create a site that helps people find and contact their congressional representatives. “It’s now evolved into something that provides better results than official government sites since we offer geographical lookups based on latitude and longitude,” asserts Tom Lee, director of Sunlight Labs. More recently, Sunlight has released APIs that track the development of legislation and can detect, for example, when legislative language moves be-

tween different statehouses or from interest groups into law.

At the federal level, activity has focused on aggregating the data published by various agencies and trying to make it useful to developers and the public. The Obama administration has been a vocal supporter of open data, and in 2009, Vivek Kundra, then federal chief information officer, launched Data.gov, a site whose goal is “to improve access to federal data and expand creative use of those data beyond the walls of government.” At the time of its launch, Data.gov had 47 datasets; at press time, the site boasted nearly 400,000 datasets.

In recent years, the Data.gov team has facilitated the organization of issues-based communities such as people with an interest in energy or health. “Then, we can put out a call to the community and find out what they need, and go to the relevant agency and try to get it for them,” explains Jeanne Holm, who joined Data.gov as a data evangelist in 2010. “Each community can spark the publication of hundreds of thousands of datasets that wouldn’t have been published otherwise, and bring people together around those conversations.”

Communities can also sound the alert when datasets do not line up. Sunlight wrote scripts to reconcile the data at USASpending.gov with the Catalog of Federal Domestic Assistance and discovered a discrepancy of \$1.3 trillion. It

**Sunlight Foundation
APIs can track
the development
of legislation
and detect when
legislative language
moves between
different statehouses
or from interest
groups into law.**

then contacted agencies, like Health & Human Services, whose data was inaccurate. “We brought them all this negative attention, but by and large, people in government are trying to do a ton with very limited resources, and what they need are better tools,” says Lee.

The British government’s counterpart to Data.gov was launched in January 2010 at Data.gov.uk. Built under the direction of Nigel Shadbolt, a professor of artificial intelligence at the University of Southampton, and Sir Tim Berners-Lee, Data.gov.uk now offers more than 5,400 datasets, including detailed crime statistics, government spending data, and health indicators like hospital infection rates.

To date, dozens of Data.gov.uk-powered apps have been created that enable citizens to do everything from researching and comparing nursing-care facilities to viewing scheduled road closures across the country. Shadbolt and Berners-Lee, director of the World Wide Web Consortium, are now leveraging their work on the site to create a new government research center, the Open Data Institute. “The mission is to extract not just social value or efficiency for the public sector, but actual economic value in terms of businesses that might develop and promote and use data,” says Shadbolt, who is crafting the Institute’s implementation plan.

Indeed, getting companies and citizens actively involved is one of the biggest promises of open data, according to advocates. The discussion is often framed in the larger context of what is known as Gov 2.0—going beyond data publication to crowdsourced information solutions. “There’s an interesting question about whether we can build enough capability and infrastructure to allow people to be creative around these assets,” says Shadbolt. SeeClickFix, a Web site and mobile app that enables U.S. citizens to report non-emergency neighborhood issues like potholes and streetlight outages, is a popular example. Borne of Web designer Ben Berkowitz’s frustration with the graffiti on his house in New Haven, CT, it has since grown into an 11-person start-up that works with city governments across the country to manage quality-of-life service requests and get usage analytics.

“We can turn on the hydrants of data, and that’s great for developers and great for a lot of companies, but for the average person on the street, it impacts them when they see an application that is of utility to them,” says Shadbolt, who cites the example of a transit app that tells commuters when the next bus will arrive. “So I have information that has delivered an immediate decision-making benefit, and that’s what people understand. It’s data that’s turned into information that, in turn, becomes knowledge because you’re using it to take an action. It’s a classic computer science view.”

Outreach remains a challenge, however, and critics often charge that Gov 2.0 empowers the already empowered, such as the tech-savvy people who do not need to be convinced that open data can improve government. Yet data advocates try to engage underserved populations.

“There’s sometimes the perception that if you do a mobile app, you’re hitting the wealthier members of the community,” says Code for America’s Pahlka. “But mobile is an incredibly important strategy if you’re looking at low-income communities.”

Pahlka cites a study by New York City’s Department of Social Services, which found that more than 80% of the people who visited its facilities were regular cellphone users, and that 35% of them owned smartphones. SMS-based apps are another way to broaden accessibility and adoption as are targeted outreach campaigns. “It’s not about a broad advertising campaign for users that are already in the know,” says Pahlka. “It’s about partnering with cities to reach the people who need these services. If you’re targeting users of social services, advertise to them in the department during the transactions.”

In spite of the progress that has been made, changing government culture also remains a challenge. Due to procurement processes and well-established power structures, the efficient release and management of data can be difficult. “The instinct to protect data generally comes from a place of public service,” asserts Pahlka. “What we try to do is illustrate that things have changed, and that the benefits outweigh the risks.”

Data.gov.uk now offers more than 5,400 datasets, including detailed crime statistics, government spending data, and health indicators like hospital infection rates.

In Boston, Code for America’s fellows tried in vain to persuade the city to release data collected by public school buses’ GPS devices. When winter hit and multiple snowstorms stranded many school buses, concerned parents overloaded schools with phone calls asking about their children. “And the fellows said, ‘It’s not hard for us to write an app that puts this data on the Web so parents can look it up themselves,’” says Pahlka. “Sometimes it’s not enough to explain why you want the data. You have to make it work for the people in the system.”

Further Reading

Howard, A.

Gov 2.0 goes local, *O’Reilly Radar*, Oct. 15, 2010.

Lohr, S.

The age of big data, *The New York Times*, Feb. 11, 2012.

Say, M.

Government plans open data push, *The Guardian*, Nov. 28, 2011.

Sterne, R.

Data-driven innovation: How open government is transforming New York City, O’Reilly Strata Conference, New York, NY, Sept. 22–3, 2011, <http://www.youtube.com/watch?v=NGyCLMwIld0>.

Van Buskirk, E.

Sneak peek: Obama administration’s redesigned Data.gov, *Wired Epicenter*, May 19, 2010.

Leah Hoffmann is a technology writer based in Brooklyn, NY.

© 2012 ACM 0001-0782/12/06 \$10.00

ACM Member News

ACM HELPS LAUNCH TURING AWARD LAUREATES FORUM

ACM is entering into an agreement with the Klaus Tschira Foundation and the Heidelberg Institute for Theoretical Studies to join and support a new event focused on winners of the Abel Prize, the Fields Medal, and the ACM Turing Award. The event is being modeled after the annual meeting of Nobel laureates in Lindau, Germany. For more than 60 years, the Lindau meeting has brought Nobel laureates into close contact with young researchers from all over the world.

The ACM-supported meeting will adopt the Lindau format, but will focus on the highest scientific awards in mathematics and computer science. The event will be called the Heidelberg Laureates Forum: Abel-, Fields- and Turing Laureates Meet the Next Generation. “In addition to the opportunity for interaction between young researchers in computer science and distinguished Abel, Fields, and Turing laureates, such an annual event will also help raise public awareness of the ACM Turing Award—particularly in Europe,” says ACM CEO John White. “We are quite excited about this new event, particularly the dimension that aims to support the careers of young researchers in computer science by bringing them in direct contact with winners of the ACM Turing Award.”

The thematic focus of the Heidelberg Laureates Forum will alternate each year between mathematics and computer science, but all recipients of the Abel Prize, Fields Medal, and Turing Award will be invited to attend. The first meeting will take place in September 2013, and the second is planned for September 2014.

The meeting will be supported by a newly established foundation, The Abel-Fields-Turing Foundation. ACM will serve on the Board of this new foundation and help shape the program each year.

—Jack Rosenberger



Game Changer

Judea Pearl's passionate advocacy of the importance of probability and causality helped revolutionize artificial intelligence.

WHEN JUDEA PEARL spoke about reasoning with probability at the first conference on Uncertainty in Artificial Intelligence in Los Angeles in 1985, the attendees who shared his outlook were seen as being outside the mainstream of artificial intelligence, recalls Eric Horvitz, distinguished researcher at Microsoft Research. But Pearl, winner of the 2011 ACM A.M. Turing Award, turned out to be one of the leaders of a revolution that “literally has changed the very nature of the discipline of artificial intelligence and computer science more generally,” says Horvitz. (An interview with Pearl, “A Sure Thing,” appears on p. 136.)

In the early 1980s, expert systems were based on sets of rules, but such reasoning systems were brittle, says Pearl, and when they provided an incorrect answer, it was not easy to figure out which of the rules was the culprit. Pearl felt strongly that humans reasoned using a probabilistic engine in their minds, but probability had fallen out of favor because applying a standard probability distribution to the number of variables involved made it exponentially complex. Yet probability was not too complex for humans. “If we do it simply,” says Pearl, “computers ought to be able to do it simply, which means there ought to be a practical approximation.”

Taking a cue from psychological theories of how children learn to read, Pearl developed Bayesian networks, graphical models of how beliefs can propagate in response to new observations. The algorithms he developed combined graph theory and probability theory to make the system simple enough that computers could reason probabilistically. “He basically used graph theory to characterize how you could represent conditional independence,” says Joseph Halpern, a professor of computer science at Cornell University.

“When you see a phenomenon exhibited by humans, it must be that computers are able to simulate it.”

However, Pearl soon believed that humans do not have a “probability engine” in their minds, but instead reason based on an understanding of cause and effect. That leads to the ability to ask the question “What if?” If you do *A*, the result should be *B*, and you can consider what the likelihood of that outcome is. “We have a causal engine in our minds, and we decorate it with probability,” he says. “This is how scientists store scientific knowledge. What leads to what?”

Also, causal reasoning provides the ability to reason about counterfactuals. If there had not been *A*, what would have happened, which gives the ability to ask why. If you take an aspirin and you get a stomachache, you can decide that you should not have taken it. “You go back and say, ‘I would have been better off had I not done what I did,’” he explains.

Philosophers have pondered causality for hundreds of years, Halpern says, and everyone from epidemiologists to lawyers worries about it. But when Pearl developed ways of representing it mathematically, he changed the discussion. “I think it’s fair to say he reinvigorated the field of causality,” says Halpern, who collaborated with him on a study of cause and

explanations. “It was sort of a new way of looking at things.”

Pearl’s work has implications well beyond computer science, in areas such as economics, epidemiology, and disease diagnosis. It is also contributing to areas of computing such as machine learning and natural language processing.

If a machine were to reason causally, making decisions about desired effects, it might have the illusion of free will, Pearl argues, suggesting that our own sense of choice may also be a useful illusion. “Imagine we have robots that communicate as if they had free will, blaming each other, praising each other, saying that you could have done better,” he says. “We’re going to learn quite a lot about ourselves if we manage to get a robot community to communicate in this manner.” And it could prove very useful to give robots this sensation of urgency. “I feel if evolution equipped us with this illusion, it has some merit,” says Pearl.

He urges computer scientists to be undaunted when they are told that a machine cannot emulate a human ability. “Don’t take no for an answer. When you see a phenomenon exhibited by humans, it must be that computers are able to simulate it,” he says. “We may not reach human level in computers, it may be just an aspiration, but aspiration leads to positive outcomes.”

Pearl has not yet decided what to do with the Turing Award’s \$250,000 prize, but says he would like to devote part of the money to overcoming skepticism about his theories in some conservative scientific circles. Or he might provide a prize for students who vie against conventional wisdom, he says, “some sort of incentive to get young people to circumvent their professors.”

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

© 2012 ACM 0001-0782/12/06 \$10.00

An Influential Theoretician

Sanjeev Arora, winner of the 2011 ACM-Infosys Award, discusses his pivotal role in theoretical computer science.

AT THE VERY heart of computer science is the question of which problems, such as the P versus NP question, can be solved efficiently.

Sanjeev Arora, the Charles C. Fitzmorris Professor of Computer Science at Princeton University, has been at the center of major developments in complexity theory during the last several decades, first gaining fame for his contributions toward proving the PCP Theorem in 1992 and the hardness of computing approximate solutions to NP -hard optimization problems.

The PCP Theorem states that every mathematical proof, of any length, can be efficiently converted into a special format in which correctness can be verified with extremely high probability by reading only a few random bits of the proof. This strikingly counterintuitive statement goes against the normal understanding of proofs where a single small error can ruin correctness. How can a small random sample find all such possible bugs? Technically, the proof of the PCP Theorem is a tour de force, making this one of the most difficult theorems in computer science.

Since then Arora has also contributed fundamental new algorithms to compute approximate solutions to the traveling salesman problem, as well as to graph partitioning problems. And his textbook, *Computational Complexity*, co-authored with Boaz Barak, has become a standard text in many universities for advanced undergraduate or introductory graduate courses on this central area of theoretical computer science.

"We are recognizing the contributions of one of the century's greatest theoreticians, who has played a pivotal role in some of the deepest and most influential results in theoretical computer science and its applications,"



Sanjeev Arora has been responsible for many significant advances in complexity theory during the last several decades.

says Henry Kautz, chair of the department of computer science at the University of Rochester and chair of the 2011 ACM-Infosys Foundation Award in the Computing Sciences.

The 2011 ACM-Infosys Award, which carries a \$175,000 prize and recognizes work by young computer scientists and systems developers, was awarded to Arora for "contributions to computational complexity, algorithms, and optimization that have helped reshape our understanding of computation."

Arora, 44, says he is currently working on several important open problems in approximation, such as graph coloring, sparsest cut, and the unique games conjecture. "I have also become interested in bringing greater mathematical rigor to machine learning," he says. "Currently, that field relies on heuristic approaches—i.e., algorithms whose performance we are currently unable to prove mathematically. I am

trying to design algorithms with provable bounds and performance."

Regarding his working style, Arora says he likes "to work on many different areas of theoretical computer science and explore new ideas and research topics. Fresh insights often arise from ideas jumping from one area to another."

While the ACM-Infosys Award is for research accomplishments, Arora's Princeton colleagues describe him as "a great mentor and advisor, with many students and postdocs, some already leaders in their own right."

Arora expresses gratitude toward the "students over the years who have been willing to accompany me on new explorations. They have been my teachers as surely as I have been theirs."

He is very active in the Center for Computational Intractability at Princeton, of which he is the founding director. Supported by funding from the National Science Foundation, the center is devoted to understanding, coping with, and benefitting from computational intractability.

"We are hard-pressed to think of another theoretical computer scientist under 50 whose work has exhibited the sustained depth over a couple of decades, and who has had such a large impact on so many areas," says Bernard Chazelle, the Eugene Higgins Professor of Computer Science at Princeton.

Arora hails from Kota, India, and moved to the U.S. at age 20. "It is fair to say that growing up in a small city in India, I could not have imagined the fun, interesting, and satisfying life—personal and professional—that I have ended up having." 

Paul Hyman is a science and technology writer based in Great Neck, NY.

© 2012 ACM 0001-0782/12/06 \$10.00



2013

TOKYO/JAPAN



8th ACM/IEEE International Conference on
Human-Robot Interaction (HRI 2013)
March 3-6, 2013, Tokyo, Japan

<http://humanrobotinteraction.org/2013/>



General Chairs

Hideaki Kuzuoka and Vanessa Evers

Program Chairs

Michita Imai and Jodi Forlizzi

Local Arrangement Chairs

Fumihide Tanaka and Hirotaka Osawa

Finance Chairs

Masahiro Shiomi and Leila Takayama

Sponsors



Supporter



Robots as Holistic Systems

Robotic solutions are increasingly applied to real world problems such as our ageing society, renewable energy, climate control, emergency response, education and exploration. These societal problems require a holistic approach to the design and development of robots that meet human needs, address technical challenges, and foster acceptance in everyday settings. The eighth ACM/IEEE International Conference on Human-Robot Interaction, HRI2013, is devoted to exploring the theme of Holistic Human-Robot development.

Important Dates

10 September 2012:

Submission of full papers and workshop/tutorial proposals

19 November 2012:

Notification of acceptance for full papers

3 December 2012:

Submission of videos and late-breaking reports

3-6 March 2013:

Conference

Conference Site

National Museum of Emerging Science and Innovation
(Miraihan, Tokyo, Japan)

<http://www.miraikan.jst.go.jp/en/>

Full Paper Submission

Authors are invited to submit manuscripts; eight camera-ready pages including figures are allowed for each full paper. Accepted papers will be published in the conference proceedings and archived in the ACM Digital Library.

Late Breaking Reports

Authors are encouraged to submit their late-breaking results as short two page papers. Accepted short papers will be presented as a poster and published in the conference proceedings.

Video Session

We invite videos related to all aspects of HRI. In addition to the importance of the lessons learned and the novelty of the situation, the entertainment value will be judged. The video itself must be self-explanatory for the audience. Accepted videos will be published in the conference proceedings and archived in the ACM Digital Library.

Tutorials and Workshops

Proposals are sought from those wishing to organize a tutorial or a workshop on a HRI-related theme. Tutorials and workshops will be held on one day before the main technical sessions.

Exhibitions

There will be an exhibition site at the conference. Financial supporters are encouraged to display state-of-the-art products and services in all areas of robotics and human-robot interaction.



DOI:10.1145/2184319.2184329

Phillip G. Armour

The Business of Software A Measure of Control

Some limitations on measurements in software.

YOU CAN'T CONTROL what you can't measure." So says Tom DeMarco at the beginning of his seminal book *Controlling Software Projects*.³ DeMarco was echoing Lord Kelvin who, a century or so earlier, had said "If you cannot measure it, you cannot improve it."

These statements seem rather intuitive, and both quotes have been used over the years to justify the development and application of metrics programs in the business of software. Certainly it would be difficult to determine if you have actually improved something if there were no way to quantify the change—but there are other considerations in measurement.

Watts, on First

In the mid-1990s I attended a presentation by the late Watts Humphrey given to an audience of telecommunications executives where he gently chided them for investing so heavily in a measurement program that was, to his mind, rather ineffectual. Humphrey's point was that this particular company was, at the time, an SEI CMM Level 1 organization. Such an organization was characterized by the Software Engineering Institute's Capability Maturity Model as Initial or

Should the control chicken come before the measurement egg or should it be the other way around?

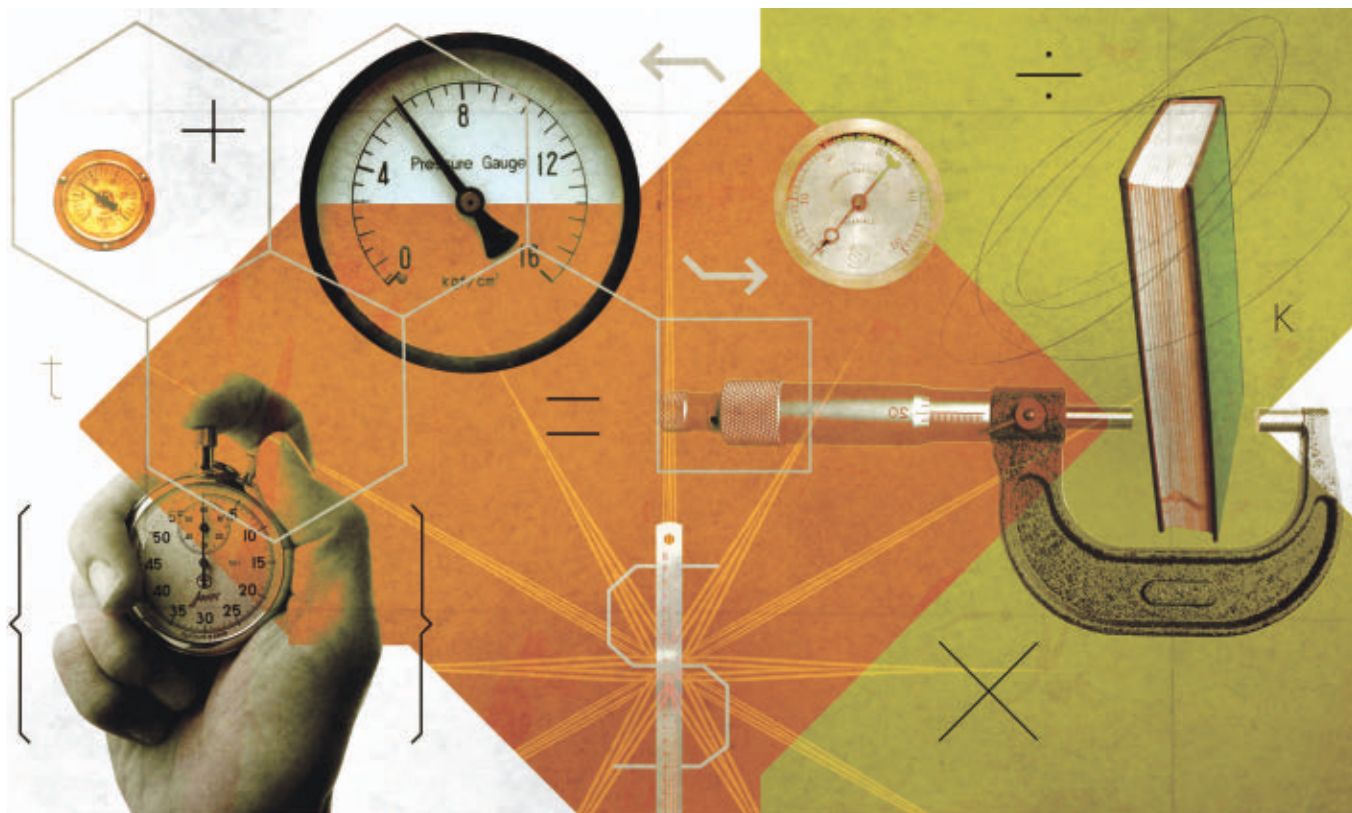
"ad hoc" meaning the processes used to build systems were recreated each time for each system and each project. In Humphrey's view this meant that, to some extent, this company's development process was out of control; or perhaps more appropriately, nothing could be inferred from measurement on this project since the next project would likely use a quite different process. As Humphrey put it, the only thing measurement would truly show was that the process was out of control and they already knew that. Therefore, he reasoned, the company should invest first in stabilizing their processes before they expended much effort in developing measurement programs.

This message received a cold response. The executives and the company had already invested heavily in their metrics program and they were proud of it. They believed it was instrumental in their technological success irrespective of their SEI level. So who was right? Should the control chicken come before the measurement egg or should it be the other way around?

Hawthorne Would

In purely metric terms, Humphrey had a point—if the engine in your car keeps surging uncontrollably it is more important to fix it than to install a new speedometer. However, this company had a long track record of using measurements to *generate* change. They intentionally employed a form of *Hawthorne Effect*^a; measurement was sometimes intended to catalyze improvement rather than to measure. At one point this company instituted a large metrics program that collected data from regular feed-

^a Named after a study done at Western Electric's plant in Cicero, Illinois between 1924 and 1932 where significant productivity increases were measured after apparently minor or placebo changes in the workplace environment. The reasoning was that simply measuring work activities tends to improve them.



back sessions between employees and supervisors simply to force managers and their direct reports to engage in regular feedback sessions and to develop feedback skills. Though measurement was touted as the purpose of the program, in reality it was much less significant than the process.

Core Metrics

At the core of systems development, there are some truly basic attributes that we need to measure. These attributes were summed up by Larry Putnam, Sr. and Ware Myers⁴ as:

- **Size:** usually of the delivered system in some useful metric. While few people actually care how “big” a system is, size is actually a proxy for the *knowledge content* that is itself a proxy for the customer-realized *value* of a system.

- **Productivity:** how effective the organization is in turning resources of time and effort/cost into valuable software products.

- **Time:** how much calendar time is required to build the system/deliver the value.

- **Effort/Cost:** the amount of work and money required to deliver the value.

- **Reliability/Quality:** the relative amount of functioning (read “correct”

knowledge) system versus non-functioning (“incorrect” knowledge) system.

As simple as they are, there are challenges to measurement even for these basic measures.

Size: Measuring Knowledge

At the core of all software size metrics is this uncomfortable truth: what we really want to measure is the *knowledge content* of the delivered software but there is no way to do that. There is no empirical way of quantifying knowledge. There is no unit for knowledge. There is not even a consistent definition of what knowledge is. However, there are *proxies* for the knowledge content that are quite measurable. They are always

At the core of systems development, there are some truly basic attributes that we need to measure.

related to the substrate or medium on which knowledge is deposited and we inevitably end up measuring the physical properties of this substrate. All else being equal, a system that is twice as “big” as another system will contain more knowledge—approximately twice as much knowledge in fact. This size might be counted using many different units: requirements, stories, use cases, and so forth. Each of these units has itself some average knowledge content or “size” coupled with some variability or uncertainty in that knowledge content. The size of the unit times the number of units indicates the size of the system. The uncertainty in the size of the unit times the uncertainty in the count of the unit indicates the uncertainty in the size of the system.²

Productivity: Measuring Effectiveness

This is a loaded term—it sounds like a manufacturing units-per-hour metric but in software it really references the team’s or organization’s cooperative ability to acquire knowledge—to learn. It is this factor, more than any other, which determines how effectively we can build systems. We can usually see when a project team is effective or inef-

fective, but proactively ensuring such effectiveness is difficult and many business practices (such as continually moving people on and off projects) make us less effective though it is hard to quantify just how much.

Time: Measuring Duration

One would think this would be very easy—just look at the clock or the calendar. But some organizations do not even record and retain information on when their projects start and even when they do what they measure may be quite ambiguous. Imagine a project against whose time recording system someone logged one hour of requirements activity two years ago, while the project team came on board yesterday. Did the project “start” two years ago or did it start yesterday? In this case the answer is pretty clear. But suppose a year ago, five people recorded a month’s worth of work, then six months ago 10 person-months were recorded, and three months ago 20 person months were recorded: now when did the project start? Such a slow ramp-up is not unusual and it means there simply was no one discrete point in time when the project “started.” And if we cannot define when a project started, clearly we cannot say how long it took to complete.

Effort: Measuring Cost

Organizations routinely play with both the effort recorded (declining to record overtime, for instance) and with the effort-to-cost ratio by not paying for this overtime or by hiring cheaper offshore resources. These practices introduce significant variance in such measures, even when they are actually taken.

What we really want to measure is the knowledge content of the delivered software but there is no way to do that.

Reliability: Measuring Quality

This is a perennial challenge in our business. Since defects are mostly deemed to be “bad” there are personal and organizational pressures that operate against consistent measurement. Since a “defect” is simply the identification of knowledge that was not gained, measuring defects suffers from some of the same challenges as measuring system size. In some ways, we can view defects as key and valuable indicators of the knowledge acquisition process.¹ But we still have a lot of variability in how we define and measure them.

Intrinsic and Artificial Variability

In measurement of software processes and products, there are two types of variability: intrinsic and artificial. Intrinsic variability simply exists in the situation whereas artificial variability is a function of the artifice of measurement. For example: Intrinsic variability occurs in predicting the size of a finished system in order to estimate a project—before we build a system we simply cannot know exactly how big it will be. We can make an educated guess, we can compare against similar system history, we can extrapolate from what we have seen before, but we do not *know*. Artificial variability occurs due to variation in the format of metrics and how we collect them. If we count system use cases as a size metric and one person writes highly detailed requirements while another dashes off vague and perfunctory products, what “use case” means is quite different. The knowledge content of one person’s use case can vary a lot from the knowledge content of another person’s use case. When recording time, if one project counts its start as the very first moment any time is recorded for any activity, while another waits until everyone is on board the meaning of “start” will be quite variable.

We do not have control over the intrinsic variability—we cannot definitively know precisely how big a system will be before we build it, for instance—but we can and should manage the artificial variability. Good metrics programs do this: they apply structure to requirements document formats and density, they define usable and practical measures of time

We can usually see when a project team is effective or ineffective, but proactively ensuring such effectiveness is difficult.

and effort, they carefully define what a defect is and how severe it might be. And they embrace the intrinsic uncertainty as something to be identified and quantified rather than denied and hidden away.

Get Control

The fact that some things are not particularly measurable does not mean measurements are not useful. The fact that there is uncertainty all around us does not mean we should pretend things are not uncertain. There is a lot we can do to remove or reduce artificial variability and one of our primary reasons to do this is to expose the irremovable intrinsic uncertainty so we can measure it and make important decisions based upon it. To make metrics more usable and useful we have to strip off this artificial variability and we do this by looking at the metrics and the metrics collection process and by applying some, well, control.

To reverse engineer the statements of Lord Kelvin and Tom DeMarco: if you do not improve it, you cannot measure it very well and, to some extent, we cannot measure what we do not control. **G**

References

1. Armour, P.G. Not defect: The mature discipline of testing. *Commun. ACM* 47, 10 (Oct. 2004), 15–18.
2. Armour, P.G. The conservation of uncertainty. *Commun. ACM* 50, 9 (Sept. 2007), 25–28.
3. DeMarco, T. *Controlling Software Projects*. Prentice Hall, Upper Saddle River, NJ, 1986.
4. Putnam, L.H. and Myers, W. *Five Core Metrics*. Dorset House, New York, NY, 2003, 33.

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

Copyright held by author.

Inside Risks

The Cybersecurity Risk

Increased attention to cybersecurity has not resulted in improved cybersecurity.

THE RISK OF being “hacked”—whatever that expression actually means—is at the heart of our civilization’s chronic cybersecurity problem. Despite decades of computer security research, billions spent on secure operations, and growing training requirements, we seem incapable of operating computers securely.

There are weekly reports of penetrations and data thefts at some of the world’s most sensitive, important, and heavily guarded computer systems. There is good evidence that global interconnectedness combined with the proliferation of hacker tools means that today’s computer systems are actually *less secure* than equivalent systems a decade ago. Numerous breakthroughs in cryptography, secure coding, and formal methods notwithstanding, cybersecurity is getting worse as we watch.

So why the downward spiral? One reason is that cybersecurity’s goal of reducing successful hacks creates a large target to defend. Attackers have the luxury of choice. They can focus their efforts on the way our computers represent data, the applications that process the data, the operating systems on which those applications run, the networks by which those applications communicate, or any other area that is possibly subverted. And faced with a system that is beyond one’s technical hacking skills, an attacker can go around the security perimeter and use a range of other techniques, including social engineering, supply-chain insertion, or even kidnapping and extortion.



It may be that cybersecurity appears to be getting worse simply because society as a whole is becoming much more dependent upon computers. Even if the vulnerability were not increasing, the successful hacks can have significantly more reach today than a decade ago.

Views of Cybersecurity

The breadth of the domain means many different approaches are being proposed for solving the cybersecurity problem:

- Cybersecurity can be viewed solely as an *insider problem*. What is needed, say advocates, are systems that prevent

ACM Transactions on Reconfigurable Technology and Systems



This quarterly publication is a peer-reviewed and archival journal that covers reconfigurable technology, systems, and applications on reconfigurable computers. Topics include all levels of reconfigurable system abstractions and all aspects of reconfigurable technology including platforms, programming environments and application successes.

www.acm.org/trets
www.acm.org/subscribe



Association for
Computing Machinery

The possibility of an active, malicious adversary is what distinguishes security from other computer science problems.

authorized users from acting improperly. Such technology would simultaneously prevent attacks from non-malicious (but improperly trained) insiders, disgruntled employees, and malware that had compromised the accounts of the loyalists. The problem with this approach is that we fundamentally do not know how to address the insider threat.

► Given that operating systems have become too complicated to make any assurances about their correct or intended operation, many cybersecurity practitioners focus on the promise of network security as a kind of silver-bullet solution. But as Iran's experience with the Stuxnet computer worm demonstrated, even systems that are thought to be isolated can be compromised by outside adversaries. Even if network security were perfect—and it is not—we would still need to secure the hosts.

► Recently, there has been an effort to frame cybersecurity as an economic problem—convincing companies to spend resources on defense and training consistent with the risk they face. This formulation assumes spending more money actually increases security, but there is no evidence to support the assumption. Indeed, one of the persistent problems with framing security as an economic problem is that there are no reliable techniques that can be used to examine a system and measure the size of its vulnerabilities and the likelihood of compromise. Such attempts to measure security inherently risk focusing attention on what can be measured, instead of what matters.

► Others see security as a holistic process that encompasses all elements of an organization's IT and HR opera-

tions. Such a broad formulation seems to have some benefits—Microsoft's Security Initiative, started in 2002, dramatically improved the security of the company's products. But most organizations lack both the technical and financial ability to make information assurance a primary goal, and even an effort the size of Microsoft's did not create unhackable software.

The possibility of an active, malicious adversary is what distinguishes security from other computer science problems. A compiler designer worries that an optimizer bug might result in an incorrect calculation or undefined behavior. A security professional knows an adversary can analyze the compiler, the buggy output, and large amounts of code to find that single instance where the buggy executable and a specific input can be used to exploit a system.¹

The adversary makes security more difficult than other computer science problems, because the adversary adapts to our best defenses and finds ways around them. Fortunately, adversaries are *not* all-powerful: they too are governed by economics, attention spans, and other external factors. There may be limits on the adversary's knowledge of the target systems. In many cases we can decrease the risk of a successful attack.

Making progress on cybersecurity requires that we address a myriad of both technical and nontechnical factors that work to prevent governments, corporations, and even individuals from securing their systems. We have real solutions to many security problems, but many are unwilling to adopt them.

Technical factors that comprise the cybersecurity problem dominate the discussion among both technologists and policymakers. These factors include language and operating system choices, security architectures, usability, and training. What is frustrating is that many of the techniques and technologies for developing secure systems that have been shown to reduce security-critical defects, such as Microsoft's Security Development Lifecycle (<http://microsoft.com/sdl>), have not been widely adopted. There is a huge gap between general practice and best practice.

Nontechnical factors impacting cybersecurity reflect deep political, social, and economic divisions within our society. These problems include shortened development cycles; the inability to attract and retain the best workers; and the general failure of our schools at early science, technology, engineering, and math (STEM) education. While it is certainly possible that the need to secure our computers will force us to find solution to these other problems, such Pollyannaish hopes seem unlikely to be realized.

In recent years there has been an effort to liken cybersecurity to a public health problem. Just as hand washing and coughing on our sleeves can help halt the spread of influenza, advocates say, good “cyber hygiene” such as running up-to-date anti-virus software and only going to clean Web sites run by reputable organizations can help stop the spread of malware and the growth of malicious botnets.

A more accurate public health metaphor might be obesity. Just as there are companies in the U.S. that benefit from the production and consumption of excess calories, while others make money treating the medical conditions that result, there are companies in the U.S. that benefit from poor security practices, while others are benefiting from mitigating the resulting problems.

Preventing security snafus is difficult and frequently thankless. It is commonly reported that chief security officers are denied resources by management because they cannot quantify the risk their organizations face or how the requested expenditures will improve the security posture. We would like to demonstrate that

It frequently feels like many organizations are implicitly relying on hackers for their security testing.

security has a significant return-on-investment, but security is frequently just a cost. Chief security officers that deploy technology are sometimes criticized for wasting money when new systems are purchased and no attack materializes. For senior managers, the risk to one’s career of being innovative is frequently higher than the risk of maintaining the same poor practices of one’s peers.

The Isolation Fallacy

One of the simplest solutions proposed for the cybersecurity problem is to run systems in secure enclaves that are disconnected from the Internet. While the idea may sound attractive, execution is impossible in practice.

Even a so-called “stand-alone computer” has a bidirectional connection to the Internet. All of the software on these machines is typically downloaded from the Internet (or from media created on machines that were connected to the Internet). The documents produced on stand-alone machines are either burned to DVD or printed—after which they are often scanned and sent by email or fax to their final destination. A completely isolated system would have very limited utility.

Just as all computers are connected, so too are all humans connected to computers. Human activities as disparate as genetic engineering and subsistence farming rely on computers and communications systems to manipulate data and send it vast distances. An attacker can cause a lot of damage by modifying a message, no matter if the data is a genetic code or a coded SMS message. Millions of people live downstream from dams with floodgates that are controlled by computers.

Over the past 30 years security researchers have developed a toolbox of techniques for mitigating many kinds of cyber attacks. Those techniques include workable public key cryptography (RSA with certificates to distribute public keys); fast symmetric cryptography (AES); fast public key cryptography (elliptic curves); easy-to-use cryptography (SSL/TLS); sandboxing (Java, C#, and virtualization); firewalls; BAN logic; and fuzzing. These breakthroughs have resulted in countless papers, successful tenure cases, billions of dollars in created

Most companies see information security as a cost or a product rather than as an enabling technology.

wealth, and several Turing awards. In spite of all this progress, cyberspace is not secure.

Some have argued that because today’s cyber infrastructure was designed without attention to security, the proper solution is redesign. Such proposals, when realized, frequently result in systems having the same kinds of problems we experience today. For example, some have proposed adding a kind of “authentication layer” to the Internet.³ Such a layer would increase the value of stolen credentials, proxy-based attacks, and implanted malware—problems that already bedevil today’s authentication layers.

We frequently discover that what was once regarded as a breakthrough security technology is really nothing more than an incremental advance. For example, considerable effort was expended over the past decade to deploy non-executable stacks and address space layout randomization on consumer operating systems. As a result, Microsoft’s 64-bit Windows 7 is not vulnerable to much of the malware that can infect 32-bit Windows XP systems. Yet even without an executable stack, Windows 7 applications can still fall victim to so-called “return-oriented programming”⁵ in which the attacker’s malicious code is created from the exploited program and a series of specially constructed stack frames, each frame executing a few instructions in the program before “returning” to the next sequence.

The Fault is Both in Our Bytes, and in Our Selves

While it is tempting to focus on technical factors impacting the cybersecurity problem, I believe nontechnical fac-

tors dominate the variety of risks we face today. Shortened development cycles and increased competition mean much of the software and configurations that are deployed have not been adequately validated. It frequently feels like many organizations are implicitly relying on hackers for their security testing.

At the same time, obsolete, bug-ridden vulnerable systems never seem to get retired. In February 2012 approximately 30% of the computers on the Internet were still running Windows XP (down from 31% the previous month), according to W3Schools.⁴ Yes, Windows 7 has vulnerabilities, but Windows XP is dramatically less secure: it should be banned from today's cyber infrastructure.

Other factors increasing the cybersecurity risk include our difficulty attracting and retaining enough software engineers, and the failure of our schools to promote technology education from an early age.

It is important to realize that cybersecurity, despite its importance, represents only a tiny part of computer science research as a whole. Security professionals rightfully point out that a single flaw in practically any program can result in a devastating security compromise. This is troubling, because most computer professionals receive little if any training in security, most CS professors and software engineers try to ignore it, and there are few security specialists. This argues for better training and the creation of a licensing or certification process.

Some people blame pay scales. While science and engineering jobs pay better than average jobs in the

U.S., they do not pay better than careers in medicine, law, and business, says Lindsay Lowell, director of policy studies at Georgetown University's Institute for the Study of International Migration. Testifying in 2011 before the House Subcommittee on Immigration Policy and Enforcement, Lowell said it is the lower salary paid to science and technology professionals that is responsible for the large number of non-U.S. students enrolled in U.S. graduate science and engineering programs.⁶

For generations educators have recognized that one of the primary purposes of schooling is to teach students how to write. As a result today's high school graduates have had at least 10 year's worth of writing instruction (and many say their writing still leaves much to be desired).

The situation is worse when it comes to technology education. Think what you want about so-called "digital natives," but experience with Facebook and video games does not translate into algorithmic thinking. Computers are part of early education in many communities, but the courses invariably teach how to be users, and not how to understand the underlying technology. Most college graduates have essentially no ability to perform even simple office automation tasks, and the typical CS graduate has less than six years' experience writing software. A real risk of our current educational system is that most graduates simply lack the experience to write security-critical software because they did not start programming in middle school.

We may be increasingly an information society, but most companies see information technology, and especially information security, as a *cost* or a *product* rather than as an enabling technology. Organizations balance their security against other competing requirements. A 2011 Bloomberg Government Survey of 172 Fortune 500 companies found they were collectively spending \$5.3 billion per year on cybersecurity and stopping just 69% of all attacks. The organizations told Bloomberg they could increase the effectiveness of their defenses over the next 12 to 18 months such that they could stop 84% of cyber at-

tacks; to do so they would need to increase their annual spending to \$10.2 billion. Stopping 95% of cyber attacks, which Bloomberg claimed would be the "highest attainable level" of security, would increase spending to \$46.6 billion per year.²

Stopping 95% of cyber attacks means one in 20 still gets through—far too many when the result of even a single successful attack can be devastating. The situation is similar to cancer treatment: if chemotherapy leaves a single surviving cancer cell, the patient frequently dies. With cybersecurity it is clear we cannot cure the patient. We must learn to live with the disease.

Live with Cyberinsecurity

There is no obvious solution to the problem of cybersecurity. While we depend on our computers, we seem incapable of making or operating them in a trustworthy manner. Much is known about how to build secure systems, but few of the people building and deploying systems today are versed in the literature or the techniques. We should be designing society so that we can survive the failure of our machines, but it is more cost-effective to create systems without redundancy or resiliency.

Reducing our cyber risk requires progress on both technical and political fronts. But despite the newfound attention that cybersecurity increasingly commands, our systems seem to be growing more vulnerable every year. ■

References

1. C compilers may silently discard some wraparound checks. US-CERT Vulnerability Note VU#162289, April 4, 2008.
2. Domenici, H. and Bari, A. The Price of Cybersecurity: Big Investments, Small Improvements. A. Holmes, Ed., Bloomberg Government Survey (Jan. 31, 2012).
3. Landwehr, C. A national goal for cyberspace: Create an open, accountable Internet. *IEEE Security and Privacy* 7, 3 (May 2009).
4. OS Platform Statistics, w3schools.com; http://www.w3schools.com/browsers/browsers_os.asp.
5. Roemer, R. Buchanan, E., Shacham, H., and Savage, S. Return-oriented programming: Systems, languages, and applications. *ACM Trans. Info. Syst. Secur.* 5, 1, Article 2 (Mar. 2012).
6. "STEM" The Tide: Should America Try to Prevent and Exodus of Foreign Graduates of U.S. Universities with Advanced Science Degrees. Hearing Before the Subcommittee on Immigration Policy and Enforcement of the Committee on the Judiciary of House of Representatives (Oct. 5, 2011), 112–164.

Simson L. Garfinkel (slgarfin@nps.edu) is an associate professor at the U.S. Naval Postgraduate School in Monterey, CA.

Copyright held by author.

While we depend on our computers, we seem incapable of making or operating them in a trustworthy manner.



Kode Vicious Scale Failure

Using a tool for the wrong job is OK until the day when it isn't.

Dear KV,

I have been digging into a network-based logging system at work because, from time to time, the system jams up, even when there seems to be no good reason for it to do so. What I found would be funny, if only it were not my job to fix it: the central dispatcher for the entire logging system is a simple for loop around a pair of read and write calls; the for loop takes input from one of a set of file descriptors and sends output to one of another set of file descriptors. The system works fine as long as none of the remote readers or writers ever blocks, and normally that is not a problem. The problem has come about because what was once handling fewer than 10 machines is now handling 40, some of which are remote across a wide area network. The obvious fix is to make the code nonblocking, but what I am surprised about is that anyone would write code this way. It is obvious from the first time you look at the code that it cannot scale.

Blocked and Loopy

Dear Loopy,

I would like to say that I am sure the original author of the code you are looking at was not trying to torture you; but after seeing many similar pieces of code, it is difficult for me to continue to accept this particular bit of make-believe. What you are probably looking at is “throwaway” or “prototype” code that got away. The person who wrote the code probably had a boss pop



into his cubical one day with a “great idea” to improve the logging system by using the network and a central dispatcher, and then asked the programmer to code up something simple to toss around. That something simple is what you now see. In my mind, I see the programmer getting the code running, and—since programmers are optimists—being excited when it ran and considering it done.

The next thing I imagine is that once the code was deployed, people found a use for it. Code that people do not find a use for rarely causes problems, because it rarely gets executed. From 10 clients, it went to 20, and then on to

the point where it broke and someone asked you to look at it.

If I were you, I would count your blessings. Taking a single, simple, read/write loop and converting it into a reasonably robust, nonblocking piece of code, while not trivial, is not a massive undertaking. Of course, while you are at it, you are going to add code to report when your clients are slow, or disconnect, or cause problems, right? Right! You could easily spend days hacking around and polishing a system like this, but I would suggest that you just add enough code and hooks so that when the system goes to 100 nodes you can split your dispatcher and run more than one of them simultaneously on separate nodes, because that is the next thing you will have to do for scalability. If you do not do this correctly, then your successor will be writing me a letter—exactly like this one.

KV

Dear KV,

My employer recently deployed a system on its network that is very sensitive to variations in network traffic. Although our team let people know that the amount of load on our network might cause problems with this particular application, it was decided to deploy the software anyway and see what happened in production. As you can imagine, most of the time things work pretty well; but occasionally, often because of random misconfigurations or because another application abuses the network

The Ultimate Online Resource for Computing Professionals & Students



<http://www.acm.org/dl>



Association for
Computing Machinery

Advancing Computing as a Science & Profession

resources, our shiny software fails completely, resulting in angry email threads and finger pointing. At this point, there is no way to turn back, and we now live in fear of the next time someone adds a new application in the network. There are ways to work around these issues, but people seem unwilling to do the necessary work and are only interested in our group “just fixing the code.” Of course we can patch and hack the code to work around temporary problems in the network, but that does not really address the problem. Why is it so difficult for people to understand when they are using a tool the wrong way?

Wrong Way Round

Dear Wrong Way,

Whenever I see people taking one tool and using it—usually poorly—for the wrong job, I am always reminded of screwdrivers. You can use a screwdriver to drive screws, yes, but you can also turn the screwdriver around and use the handle as a hammer to drive nails. Of course, doing this means you are at risk of poking your eye out, but, you say, “I need to drive only this one nail, I am sure it will be OK.” And it is OK, until the day when it isn’t. Software, being far more malleable than a screwdriver, is subject to this extension problem far more often than physical tools.

There are a couple of ways to make your point in these situations. One is simply to let the code break and watch people suffer. I recommend against developing an evil laugh or learning to cackle, as that will give you away. While this is an enjoyable fantasy, it is not very practical in a work environment. There is probably a good reason for your company to use the code you are

Code that people do not find a use for rarely causes problems, because it rarely gets executed.

There is probably a good reason for your company to use the code you are complaining about.

complaining about, and it behooves you to do what you can to help your company use it correctly.

Instead of screaming, or cackling, or pulling your hair out, you can try to explain to one person, rather than to a group, how the software works and its limitations. If you can find one other person who understands the problem, that can help you in two ways. First, it will make you feel less crazy—there is nothing worse than being the only person who sees or understands a problem. Second, it will help convince others of the correctness of your position. If you can get momentum behind your idea, then maybe you can convince the powers that be to use the system correctly and within its design parameters. Failing that, at least you will have someone to commiserate with when the system collapses again.

Like so many problems in computing, the screwdriver problem is a human problem and not a technical one, and thus it requires a human solution.

KV

Related articles on queue.acm.org

A License to Kode

George Neville-Neil

<http://queue.acm.org/detail.cfm?id=1217262>

Coding Smart: People vs. Tools

Donn M. Seeley

<http://queue.acm.org/detail.cfm?id=945135>

Building Scalable Web Services

Tom Killalea

<http://queue.acm.org/detail.cfm?id=1466447>

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

Copyright held by author.

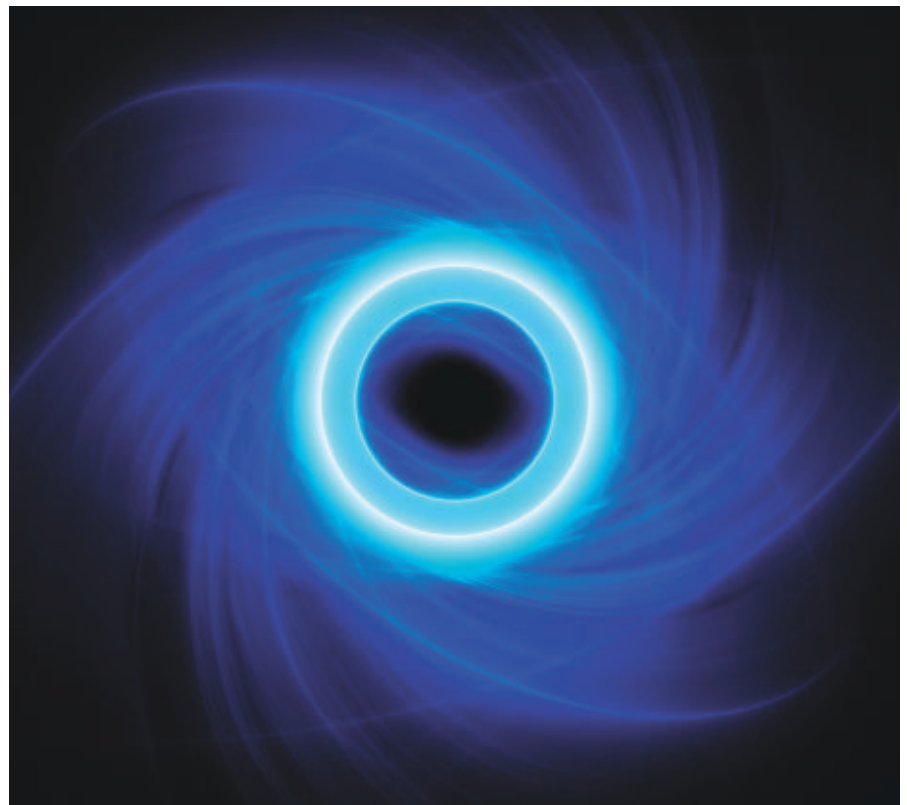
Privacy and Security

Security of the Internet and the Known Unknowns

Seeking answers to questions about Internet vulnerabilities.

THE INTERNET IS not secure. The fundamental mechanics of the Internet have well-known and long-standing security issues that occasionally disrupt it. This is usually an accident and mostly caused by human error, but it is speculated that by exploiting the openness and trust on which the Internet is built, “bad people” could build a “cyber nuke” to take down the network. Serious as these security issues appear to be, they have not apparently created any serious problems thus far. Does this mean we have simply been lucky, or are the issues more theoretical than actual?

When we think about the Internet, we tend to think about the equipment and the protocols—the technology that makes it all work. But the Internet is a many-layered system that depends not only on technology, but also on people, coffee, and money. The equipment and protocols are the network layer. Each network in the Internet has a Network Operation Center (NOC) that monitors its own network and its connections to other networks, responds to incidents when they occur, and continually strives to maintain acceptable levels of service and reliability, at an acceptable cost. Each NOC acts independently and interacts with other NOCs, collectively forming the operational layer—people and coffee. Every network operator makes their own commercial decisions, forming the commercial and economic layer(s), driven



by money. Finally, there is the policy or regulatory layer, which sets the context in which all networks operate. Each layer is a complex system in its own right, and each one plays its part in the security and reliability of the Internet. Consider, for example, the Border Gateway Protocol (BGP), a key component of the network layer.

Layers of Complexity

The essence of the Internet is that every network is able to reach every

other network. The routing information that each network needs is distributed across the Internet by BGP. Unfortunately, BGP is not secure. BGP will happily distribute invalid routing information, and offers no way to distinguish valid from invalid routes. Every now and then, some network administrator somewhere makes a small mistake that generates bogus routing information that BGP blindly accepts and relays across the Internet. The effect of such a “route leak” is that data

is diverted from its intended destination, usually ending up in a black hole from which there is no return. Since that can be achieved by accident, who can say what might be achievable with malice aforethought?

The Border Gateway Protocol has been with us for a long time—the latest version (version 4) is nearly 18 years old. So why have these issues not been addressed? First, what is remarkable about BGP is not that it is imperfect. Rather, given the exponential growth of the Internet since BGP was designed, what is truly remarkable is that it works at all. Second, it turns out that the validity of routing information is difficult to establish. Third, a more secure BGP may be either more brittle or less flexible, or both. This would make it more vulnerable or less able to adapt in a crisis, and hence not necessarily an improvement.

Making BGP more secure is difficult and the extra equipment and running costs are significant. The most recent effort, BGPSEC, is a work in progress in the IETF.^a Assuming BGPSEC is adopted, the IETF working group envisages a five- to 10-year timescale for deployment, and is rather depending on Moore's Law reducing the cost of the required extra memory capacity and processing power in the meantime. Even so, as currently defined BGPSEC does not address the issue of route leaks.

The insecurity of BGP leads to failure at the network layer. But securing the Internet is not solely a technology issue: the operational layer detects and responds quickly to route leaks and the like. The China Telecom incident that occurred in April 2010, in which approximately 15% of Internet addresses were disrupted for perhaps 18 minutes,^b is not only an example of the insecurity of the technology but also a testament to the effectiveness of the operational layer.

The continuing insecurity of BGP can be seen as a failure of the commercial and economic layer. The cost

to each network, in equipment and in effort, of a more secure form of BGP would be high. What is worse, the benefit to each network would be limited until or unless others did the same. A more secure Internet would be a common good, but the incentive to create or maintain that common good is missing. On the other hand, given how effectively the operational layer compensates for the insecurity of BGP, perhaps the commercial/economic system has not failed. Perhaps the invisible hand has guided us to a cost-effective balance between security and the ability to deal with the occasional failure. Intervention to increase security would be a failure of the policy layer if, in the name of safety, unnecessary costs were forced on the system.

The great strength of the Internet is that it harnesses the independent efforts of tens of thousands of separate organizations. They cannot all connect directly to each other, so the Internet is a vast mesh of connections over and above the component networks, moderated by BGP. Each network monitors and manages its own small part of the system, but nobody has oversight over the entire mesh—there is no NOC for the Internet as a whole! A key function of a NOC is to monitor network performance; but for the Internet there is little data on how well it performs normally, let alone how well it actually copes with failure.

Another key function of a NOC is to ensure there is spare capacity in the network, available for use when things go wrong; but for the Internet we do

A more secure Internet would be a common good, but the incentive to create or maintain that common good is missing.

not know how much spare capacity there is, or where it is—so we cannot even guess whether there is enough capacity in the right places, should something unprecedented happen.

Nobody would advocate a central committee for the Internet. However, the network's reliability is of vital interest, so it is remarkable how little we know about how well it works. Hard information is difficult to obtain. Much is anecdotal and incomplete, while some is speculative or simply apocryphal. The report *The Internet Under Crisis Conditions: Learning from September 11*,² is an exception and a model of clarity; the authors warn: "...While the committee is confident in its assessment that the events of September 11 had little effect on the Internet as a whole...the precision with which analysts can measure the impact of such events is limited by a lack of relevant data."

Effects of Internet Vulnerabilities

What is the realistic, likely impact of Internet vulnerabilities? What can be done to cost-effectively mitigate the risks? Is the longevity of these issues a symptom of a market failure, and if so, should government act to secure this critical Infrastructure? These are all good questions but, sadly, we do not have good answers and we are hampered by a lack of good data.

Monitoring Internet performance could provide the relevant data. Unfortunately, it would be difficult to implement such monitoring: the sheer scale of the Internet is an obvious problem, and then each layer has its own issues. First, there are many technical problems: what data should be collected, where and how to collect it, how to store it in a usable form, how to process it to extract useful information, and so on. Some of these are engineering issues; others are research topics. Second, such a system would be a common good, with the usual issues of incentives to create and maintain—or, more bluntly, who would pay for it? Third, Internet networks compete with each other and some of the data would be deemed commercially sensitive—so, there could be some self-interested resistance to creating the common good. Fourth, there is a fine line between monitoring and control, and a fine line be-

a The Internet Engineering Task Force—see <http://www.ietf.org> for more about the IETF in general, and <http://datatracker.ietf.org/wg/sidr/charter/> for more about the BGPSEC initiative in particular.

b <http://www.renesys.com/blog/2010/11/chinas-18-minute-mystery.shtml>.

An accident investigation board for the Internet would be a modest and minimally intrusive step, though not without its own challenges.

tween monitoring performance and monitoring content—lines that different jurisdictions might take quite different views on.

All day, every day, the Internet copes with circuit and equipment failures, and almost nobody notices. Occasionally an event—a route leak, the cutting of critical undersea cables, the discovery of a software problem—has a more significant impact, but the Internet just keeps rolling along. Many in the industry will argue this is evidence for the inherent reliability of the Internet, and, if improvement is required, it is best left to the industry. Any outside interference is generally deemed likely to be clueless and destructive.

On the other hand, every incident is a small experiment in how well the system responds. But it is nobody's job to investigate, assess the impact, and find the lessons to be learned. So, nobody does it, or at least nobody does it thoroughly or authoritatively. In some cases, those closest to an incident do not care to publish the exact causes or the entire history. This can be for commercial reasons, or for security reasons, but often simply because they have better things to do with their time. Perhaps we can learn from the airline industry. Air travel has become ever safer, not by trying to gain perfect knowledge of all possible combinations of weather, equipment failure, human error, and so forth, but by learning the lessons of every incident. This common good is funded communally—nobody ex-

pects the industry to do this by itself. Further, air accident investigators have found ways to deal with the commercial and other issues, and to operate in a global industry.

An accident investigation board for the Internet would be a modest and minimally intrusive step, though not without its own challenges. It would need to develop and refine ways to assess the impact of incidents. It would need to establish what day-to-day data gathering is most useful and most easily achieved. Unlike comprehensive monitoring, which looks like a step too far, this would be a step-by-step way of approaching a 20% solution that might provide 80% of what we need.

Conclusion

We do not know whether a more secure Border Gateway Protocol would materially improve the reliability of the Internet, let alone whether the improvement would be cost justified. We do not know whether it would be more effective to improve the operational layer, so that it would cope better with the known vulnerabilities, and would be better prepared to deal with as-yet unknown ones. Assuming we can work out how best to improve things, we do not know how to construct a system of incentives to implement those improvements. And the insecurity of BGP is not the only threat to the Internet's reliability.

If we are to improve the security and reliability of the Internet, we really need better data on which to base policy and engineering decisions...and that is a common good to which government could usefully contribute. C

References

1. Hall, C., Clayton, R., Anderson, R. and Ouzounis, E. *Inter-X: Resilience of the Internet Interconnection Ecosystem*. P. Trimintzios, Ed. ENISA Report; <http://www.enisa.europa.eu/activities/Resilience-and-CIIP/networks-and-services-resilience/inter-x/interx-report/interx-report/>.
2. Partridge, C. et al. *The Internet Under Crisis Conditions: Learning from September 11*. The National Academies Press, Washington, D.C., 2002.

Chris Hall (chris.hall@highwayman.com) is a senior engineer and manager with Highwayman Associates in Leatherhead, U.K., and the founding executive director of the Communications Innovation Institute (now called the Communications Research Network) at the Cambridge University Computer Laboratory. Recently, he has been working on the Quagga open-source BGP daemon (sponsored by Euro-IX), and co-wrote the report¹ from which this column is derived.

Copyright held by author.

Calendar of Events

June 15–16

International Symposium on Memory Management, Beijing, China, Sponsored: SIGPLAN, Contact: Vitek Jan, Email: jv@cs.purdue.edu

June 18–20

International Conference on Engineering, Technology, and Innovation, Munich, Germany, Contact: Klaus Sailer, Email: klaus.sailer@sce.web.de

June 18–22

The 21st International Symposium on High-Performance Parallel and Distributed Computing, Delft, Netherlands, Contact: Dick Eperna, Email: d.h.j.eperna@tudelft.nl

June 20–22

17th ACM Symposium on Access Control Models and Technologies, Newark, NJ, Sponsored: SIGSAC, Contact: Vijayalakshmi Atluri, Email: atluri@andromeda.rutgers.edu

June 25–27

High Performance Graphics, Paris, France, Sponsored: SIGGRAPH, Contact: Michael C. Doggett, Email: mike@cs.lth.se

June 25–28

NASA/ESA Conference on Adaptive Hardware and Systems, Nuremberg, Germany, Contact: Dr. Umesh D. Patel, Email: umeshkumar.d.patel@nasa.gov

June 25–28

Federated Events on Component-Based Software Engineering and Software Architecture, Bertinoro, Italy, Sponsored: SIGSOFT, Contact: Raffaella Mirandola, Email: mirandola@elet.polimi.it

June 25–28

23rd ACM Conference on Hypertext and Social Media, Milwaukee, WI, Sponsored: SIGWEB, Contact: Ethan V Munson, Email: munson@uwm.edu

The Profession of IT

The Myth of the Elevator Pitch

Instead of pitching, listen and offer.

BEHOLD, ON THE elevator you just boarded is a key executive or leader you have long wished to meet. You have approximately half a minute to say something about your project that will engage that person and get that person's help. What a tremendous boost to your project that would be! Could you do that?

One of the most common pieces of professional wisdom is to prepare and practice a short speech that you can launch whenever you need to make a quick summary of your project, your company, or your idea. You cannot predict when the circumstances will arise where it would be valuable to do this.

The elevator pitch, sometimes known as the elevator speech, is a short summary that quickly defines a product or service and its value proposition. A successful pitch induces the listener to make a decision sought by the speaker. The pitch is usually approximately 30 seconds, never more than two minutes.

There is certainly great value in being able to give a clear, concise, memorable summary of your work. However, the elevator pitch has been enshrined in mythology that greatly inflates its importance and promises an outcome it cannot deliver. The success of your venture depends on much more than a short pitch. We will examine the elevator pitch claim and offer guidelines on how you can make a short summary be a useful part of your repertoire.



Pitch Pervasiveness

The lofty exemplar of elevator pitches is the TV reality show “Shark Tank,” which features entrepreneurs pitching to a panel of investors. Investors can do anything from rejecting pitched proposals to opening negotiations or offering outright purchases. The show offers great entertainment, drama, and suspense. We can only imagine the fight-to-the-death competition of elevator pitches in the auditions for the show.

A recent celebration of the pitch appeared in the popular *Sherman's Lagoon* cartoon by Jim Toomey. One character, about to enter an elevator pitch contest, tells his friend that an elevator pitch “is a fast-and-furious business proposal I give you while we ride in an elevator together. By the end of the ride, you like me enough to give me money.” His friend responds, “There isn't a building tall enough.”

Elevator pitches are a hot topic on the Web. Hundreds of Web sites offer

courses, videos, and books to teach this valuable skill. A typical checklist appeared in an article by Pincus⁴:

- Know your purpose
- Know your target
- Focus on your customer
- Be authentic
- Be specific
- Be prepared
- Be concise
- Solve a problem
- Show your passion
- Practice

There is even a Web site that offers 10,000 examples of elevator pitches in case you have a mental block (<http://www.speech-topics-help.com>). The quintessential pitches stick in listeners' minds.² Public speaking organizations such as Toastmasters have offered practice sessions for elevator pitches for many years.

Do Elevator Pitches Really Make A Difference?

The advice to prepare an elevator pitch is commonly presumed to be valuable for many professionals, including project managers, sales engineers, visionaries, and policy makers. Entrepreneurs preparing to pitch to venture capitalists are often sternly warned to have a great pitch because, it is said, the VCs judge the quality of the idea and the team by the quality of the pitch.

With all this enthusiasm, you might be surprised to learn that most elevator pitches do not lead to the final outcomes sought by their speakers. For example, innovators use pitches but suffer a failure rate of about 96%.¹ It is like the lottery—your pitch is almost certain to be a losing ticket, but you buy it because it is cheap and there is a

A successful pitch induces the listener to make a decision sought by the speaker.

The elevator pitch has been enshrined in mythology that greatly inflates its importance and promises an outcome it cannot deliver.

chance that your 30-second pitch could change *everything*.

While there have been no rigorous studies of pitch outcomes, there have been several studies about the influence of business plans, on which many pitches are based. If the business plan makes a big difference to investor decisions, then there is a reasonable chance that its synopsis (the elevator pitch) might have a big impact too.

However, until recently most of the studies of business plans were plagued with issues such as small and unrepresentative samples and weak statistical methodologies, which led to unreliable and contradictory claims about whether business plans affect investor choices.

Finally, in 2009, two Stanford-trained economic historians published a definitive study that looked carefully at the whether business plans actually convey any information to VCs.³ This study was a breakthrough because it used a much larger sample (762 plans) than used in previous studies, the plans were representative of those typically screened by VCs, and the study used rigorous statistical methods to achieve its results.

The authors unequivocally ruled out the idea that business plans communicate anything to VCs: "We find that neither the presence of business planning documents nor their content serve a communicative role for venture capitalists...[O]ur findings are consistent with the view held by VC practitioners who dismiss efforts to systematically evaluate the role of business planning documents."

In other words, VC investors do not—perhaps cannot—infer much about likely success from business plans, let alone their elevator cousins.

So, the data does not support the conventional wisdom that the elevator pitch is a key to success with innovations. The convention is a myth. Is this an example of a sticky, untrue idea?²

The Bigger Picture

In escaping a myth, we want to preserve its nugget of truth while defining new and more effective actions. To do this, we must step back and look at the bigger picture.

The desired outcome of a project is to produce a result—for example, a system or product—that brings value to a group of customers; they adopt it as part of their practice.

The innovation process producing this outcome has many pieces. One summary of the pieces is the Eight Practices framework.¹ The first five practices—sensing, envisioning, offering, adopting, and sustaining—are the main elements of the process. The last three—executing, leading, embodying—create an environment of success for the first five.

At first glance, it appears that the elevator pitch is a component of the envisioning practice. The purpose of that practice is to tell a compelling story (a "vision") of how the world would be if the innovation idea were incorporated into it. The story makes it obvious what problem is solved and why the innovation proposal is likely to solve it. The elevator pitch might appear to be a précis of this story. And another business construct, the motto, may seem like a précis of the elevator pitch. Thus we have a hierarchy: envisioning story; elevator pitch; and motto.

Here are two examples:

► The John Deere Company has a vision about farm machines improving productivity by factors of 10 or more. In their pitch, they envision a line of tractors and tools that perform impeccably and never wear out, so that farmers get the most out of their investment. They ask their investors to join them in making this happen. They summarize their vision with the motto, "Nothing runs like a Deere."

► NASA has a vision of a space elevator structure that can lift payloads

to orbit, 100,000 kilometers up, at a cost per kilogram less than 1/100 of costs with current rocket technologies. They organize competitions to try out different lift mechanisms, materials, payloads, and energy recovery systems. They invite investors to join them in making this dream become a real technology. Although they have not selected a motto, it might be “Lifting your loads to the stars.”

But the notion that a pitch is a précis of an envisioning story is not quite right. A closer examination reveals that a pitch is actually a combination of a précis of the envisioning story and an offer. The offer to make the vision happen is the heart of the third practice. In the example(s) of pitches cited above, you can see not only a summary of the idea but an invitation to investors to help make it happen.

The standard idea of a pitch is that it is a communication—a presentation transmitting information from a speaker to a listener. In contrast, the core idea in the Eight Practices of innovation is that each practice is a conversation that generates a commitment to produce an outcome essential to the innovation.¹ The envisioning story is a conversation to elicit the listener’s commitment to imagine how the world would be with the new idea incorporated into it. The offer is a conversation to elicit a listener’s commitment to join the speaker in making that world happen.

The problem with the communication idea is that communications do not elicit commitments. Conversations elicit commitments. Commitments produce action.

We can now define a pitch as a short conversation that seeks a commitment to listen to an offer conversation. It is transitional between the envisioning and offering conversations. The pur-

The advice to prepare an elevator pitch is commonly presumed to be valuable for many professionals.

We can now define a pitch as a short conversation that seeks a commitment to listen to an offer conversation.

pose of a pitch is to engage the other person into a conversation with you about your idea. This is a sharp contrast with the conventional idea that a pitch is a transmission of information.

Peter once assisted his coauthor Bob Dunham in leading a workshop with small-business CEOs. Bob formed everyone into groups of three and asked them to “state their offer” to their two listeners, then receive constructive feedback on how engaging their offer is. He specifically said to think of it as a version of the elevator pitch. In all the groups, the feedback was largely negative. The elevator pitches were not engaging. Bob knew that something was amiss because all these CEOs were successful in their businesses. Why would a warm-up exercise about what their business offers be so unsuccessful? I conferred with Bob and we concluded that the CEOs were working from information-transmission notion of their pitch. They decided to try an experiment in which Peter would break that paradigm in his group. Bob ran the exercise again with the same instructions but different groups. As before the first two speakers in Peter’s group presented their “offers” and got lukewarm feedback. They then looked at Peter for his turn. This little segment of conversation took place:

Wendy: Your turn.

Peter: Before we start ... are you interested in innovation?

Wendy: You bet. If my business does not come up with an innovative idea soon, we’ll be gone.

Peter: I’m writing a book on how to succeed at innovation. Would you be interested in talking about it at lunch?

Wendy: Sure thing! But let’s get back to our work. What is your offer?

Peter: I just made it.

The point that the whole group understood after this was that the purpose was to gain a commitment from the other person to have more conversation with you about what you can offer. If you think about the elevator pitch as an offer to discuss how to solve the other person’s problem or get a job done for them, you will approach the conversation in a different way from a presentation pitch.

Conclusion

We call the elevator pitch a myth because it is a widely shared belief that a short fast-paced presentation is essential to the success of a project or innovation. The data show almost no correlation between these pitches or their supporting business plans and the final desired outcomes.

The useful nugget of the myth is that a short compelling summary of your idea is worth developing. It will help you in completing the envisioning practice, which is essential to success. It will also help you in making connections with other people—such as those whom you might serendipitously encounter in an elevator—and who might be helpful to you in the future.

To make this work, you need to reinterpret the pitch. It is not a transmission of information but an offer to have a conversation. It is often much easier to ask someone to join you in a conversation than it is to present a polished, sticky, commercial-grade presentation. A conversational pitch will get you closer to your idea being adopted. ■

References

1. Denning, P. and Dunham, R. *The Innovator’s Way*. MIT Press, 2010.
2. Heath, C. and Heath, D. *Made to Stick*. Random House, 2007.
3. Kirsch, D., Goldfarb, B., and Gera, A. Form or substance: The role of business plans in venture capital decision making. *Strategic Management Journal* 30 (2009), 487–515.
4. Pincus, A. The perfect (elevator) pitch. *Bloomberg Businessweek* (June 18, 2007); http://www.businessweek.com/careers/content/jun2007/ca20070618_134959.htm

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA, is Editor of ACM Ubiquity, and is a past president of ACM.

Nicholas Dew (ndew@nps.edu) is Associate Professor of Strategic Management at the Naval Postgraduate School in Monterey, CA. He researches entrepreneurship and is a co-author of the recently published textbook *Effectual Entrepreneurship*.

Copyright held by author.

Viewpoint

Why Computer Scientists Should Care About Cyber Conflict and U.S. National Security Policy

Cybersecurity and policy issues for computer scientists.

IN THE LAST several years, cyber security has been the focus of a great deal of media attention that has reflected public policy concerns worldwide about this topic. For example, the *Sunday Times of London* quoted Israeli Prime Minister Binyamin Netanyahu as saying that “Israel must turn into a global cyber superpower.”² Through the National Computer Network Emergency Response Coordination Center of China, the Chinese government reported China experienced nearly a half-million cyber attacks in 2010, about half of which originated abroad.³

Of particular note is the May 2011 publication of the White House’s International Strategy for Cyberspace, in which it asserted: “when warranted, the United States will respond to hostile acts in cyberspace as we would to any other threat to our country... We reserve the right to use all necessary means—diplomatic, informational, military, and economic—as appropriate and consistent with applicable international law, in order to defend our Nation, our allies, our partners, and our interests.”⁷ Amplifying these sentiments, a number of news stories subsequently appeared reporting the U.S. believed a cyber attack on the U.S. could be regarded as an act of war, and that the



The U.S. Department of Homeland Security logo is reflected in the eyeglasses of a cybersecurity analyst in the watch and warning center at the Department of Homeland Security’s cyber defense facility at Idaho National Laboratory.

U.S. could respond to such an act using traditional military force (for example, see Gorman¹).

Some have been skeptical of such statements. One argument is that it is impossible to attribute a cyber attack to any specific nation-state, so against what party would any nation retaliate in the wake of a devastating cyber attack on it? Another argument is that the strategy assumes state sponsorship is necessary for such an attack. What prevents a non-state actor from conducting such an attack? Indeed, the necessary technology is available

at big-box stores, and much of the necessary knowledge is freely available on the Internet.

But such arguments oversimplify both the threat and the strategy.

Threat and Strategy

Consider first the threat. Most of what is discussed in the popular media as “cyber attacks” is really espionage of various kinds. What is “lost” is information: technical documents, political memos, credit card numbers, Social Security numbers, money in bank accounts, business plans, and so on. As

a http://news.xinhuanet.com/english2010/china/2011-08/09/c_131038851.htm.

most computer scientists know, these are breaches of confidentiality—the legitimate owner still has the information, but someone else has it as well, someone who should not have it and who might be able to use it against the legitimate owner.

These acts are undeniably unfriendly—but do they amount to “acts of war”? Espionage is not traditionally regarded as a violation of international law—primarily because all nations do it. They *do* violate domestic law, which is why such acts are (properly) regarded as criminal acts—appropriate for investigation and prosecution by law-enforcement authorities.

But the fact that what we have mostly seen to date is cyber espionage should not blind us to the fact that the range of possibilities for something bad to happen to us is not at all limited to the loss of confidentiality for sensitive information. Computers manage electric power generation, airplanes, cars, heating and cooling systems, flood control gates, sewage systems, and so on. They are also central to a U.S. defense strategy that relies on having an information superiority on the battlefield that enormously increases the capabilities of existing forces.

A number of examples of actual cyber attacks—actions taken to destroy, disrupt, or degrade computers—are known publicly. It is alleged that in 1984, the U.S. modified software that

The attribution of any kind of cyber operation, whether for attack or exploitation, is technically difficult.

was subsequently obtained by the Soviet Union in its efforts to obtain U.S. technology. Ostensibly designed to operate oil and gas pipelines, the Soviets used this software to operate a natural gas pipeline in Siberia. After a period in which all appeared normal, the software allegedly caused the machinery it controlled to operate outside its safety margins, at which point a large explosion occurred.⁶ And, in 2010, the Stuxnet worm disrupted industrial control systems in the Iranian infrastructure for enriching uranium, apparently destroying centrifuges by ordering them to operate at unsafe speeds.³

Compared to acts of espionage, such actions are closer to the boundary between peace and war because they achieved effects that could have been achieved through the use of tra-

ditional kinetic weaponry such as bombs. Do these acts amount to acts of war? To date, the international community has not made such a determination. But this fact does not suggest there is no possible cyber attack that would cross the line. Indeed, given the increasing dependence of much of an advanced nation’s critical infrastructure on computers for safe and efficient operation, the *possibility* of a catastrophic cyber attack on an advanced nation cannot be ruled out—widespread power outages affecting hundreds of millions of people, a hacked air traffic control system causing airplanes to crash, military forces unable to deploy, and so on.

Are any of these catastrophic scenarios likely? Much of the public debate about such matters makes it seem these scenarios are imminent and they are easy to do, for example, by some lone teenage hacker/terrorist working in a basement in a far-off land. Nonsense. A long-lasting catastrophic effect on the U.S. through a cyber attack would be very difficult even for a major nation-state to achieve. Still, policymakers are paid to make contingency plans even for unlikely events—and the policy question is this: *If* a catastrophic cyber attack against the U.S. such as I have described did occur, *should* the U.S. regard it as an act of war? For *some* sufficiently high level of damage and destruction to the U.S., surely the answer is yes. The 9/11 terrorists committed crimes against the U.S.—but the international community supported the U.S. call for treating the events of 9/11 as an armed attack warranting a forceful military response.

What about attribution? What does an act of war mean if you cannot identify the responsible party? There is no question that the attribution of any kind of cyber operation, whether for attack or for exploitation, is technically difficult. If the particular techniques of any given operation have never before been seen, and if the perpetrator has concealed his tracks perfectly, and if no one else knows he is responsible for the operation, and if there are no circumstances to suggest he might be behind the operation, then attribution may well be impossible. And indeed all of these condi-



Senator John Rockefeller spoke before the Senate Homeland Security Committee in February 2012 on the urgent need to pass comprehensive cybersecurity legislation.

PHOTOGRAPH COURTESY OF JOHN D. (JAY) ROCKEFELLER IV

tions do hold for many of the acts of cyber espionage and cyber attack we have seen to date.

Would these conditions be true for a cyber attack that might plausibly be regarded as an act of war? Perhaps, but perhaps not. For example, cyber attackers may make technical mistakes that leave behind clues about their identity on some of the systems they have compromised. They may use a technique that has been seen before. They may have discussed their plans on a bulletin board that is being monitored. An intelligence official who provided to the perpetrators information useful for conducting the attack may discuss his actions on an intercepted phone call. Political circumstances (such as international tensions) may suggest a particular national actor that might gain a significant advantage from conducting such an attack. Although they do not guarantee it, all of these possibilities increase the likelihood attribution could be established.

Policymakers have some experience with many kinds of crises, but their understanding of the cyber world is, with some exceptions, sketchy and incomplete. Nevertheless, in the event of cyber crisis, they will make decisions with whatever information and knowledge they have. Computer scientists today are in a position to play an important role in helping to shape national and international policies regarding cyber conflict.

What might such a role entail? The issues are complex, from both technical and policy standpoints, and are worthy of serious intellectual atten-

Computer scientists are in a position to play an important role in helping to shape national and international policies regarding cyber conflict.

We are in the earliest stages of an ongoing policy debate about matters of war and peace in cyberspace.

tion. Some interesting issues include the following:

- Attack assessment. Knowing that a nation or even a particular facility is under serious cyber attack is highly problematic given the background noise of ongoing cyber attacks occurring all the time. What information would have to be collected, from what sources should that information be collected, and how should it be integrated to make such a determination?

- Geolocation of computers. Given that computers are physical objects, every computer is in some physical location. Knowledge of that location may be important in assessing the political impact of any given cyber attack.

- Techniques for limiting the scope of a cyber attack. Associated with any bomb is a lethal radius outside of which a given type of target is likely to be unharmed—knowledge of a bomb's lethal radius helps military planners minimize collateral damage. What, if any, is the cyber analog of “lethal radius” for cyber weapons?

- How could a penetration of an adversary's computer system be conducted so that the adversary knows the penetration is an exploitation rather than an attack?

- Given a continuing and noisy background of criminal and hacker cyber attacks, how would two nations that agreed to a “cyber cease-fire” know the other side was abiding by the terms of the agreement?

- How might catalytic cyber conflict between two nations be avoided? (Catalytic conflict refers to conflict between two parties initiated by a third party, perhaps by impersonating one of the two parties.)

- How can small conflicts in cyberspace between political/military adversaries be kept from growing into larger ones?

The first four listed items address technical issues that are important to parties on either side of a cyber attack or exploitation (that is, as victim or as perpetrator).⁵ The last three items are some of the 50-plus policy-related questions relevant to conflict in cyberspace described in a 2010 report of the U.S. National Research Council,⁴ and the reader should notice that all of them transcend the individual concerns of any particular nation and require a degree of familiarity with concepts of computer science and information technology that is second nature to most *Communications* readers.

Conclusion

We are in the earliest stages of an ongoing policy debate about matters of war and peace in cyberspace, and the voice of professional computer scientists should be heard in that debate. Whatever one's views on the topic, dialog and discussion within the computer science community about this matter can help policymakers make more informed decisions in this area. □

References

1. Gorman, S. Cyber combat: Act of war. *Wall Street Journal* (May 31, 2011); <http://online.wsj.com/article/SB10001424052702304563104576355623135782718.html#zzz1O9BF3ADz>.
2. Mahnaïmi, U. Israeli military plots to cripple Iran in cyberspace. *Sunday Times* (Aug. 7, 2011).
3. Markoff, J. A code for chaos. *New York Times* (Oct. 2, 2010); <http://www.nytimes.com/2010/10/03/weekinreview/03markoff.html?ref=stuxnet>.
4. National Research Council. *Proceedings of a Workshop on Detering Cyberattacks: Informing Strategies and Developing Options for U.S. Policy*, National Academies Press, 2010; http://www.nap.edu/openbook.php?record_id=12886&page=1.
5. National Research Council. *Technology, Policy, Law, and Ethics Regarding U.S. Acquisition and Use of Cyberattack Capabilities*, W.A. Owens, K.W. Dam, and H.S. Lin, Eds., National Academies Press, 2009; http://www.nap.edu/catalog.php?record_id=12651.
6. Reed, T.C. *At the Abyss: An Insider's History of the Cold War*. Ballantine Books, New York, 2004.
7. White House. *International Strategy for Cyberspace*, May 2011; http://www.whitehouse.gov/sites/default/files/rss_viewer/international_strategy_for_cyberspace.pdf.

Herbert Lin (hlin@nas.edu) is chief scientist at the Computer Science and Telecommunications Board of the National Academies, where he has been study director of major projects on public policy and information technology including the 2009 NRC study *Technology, Policy, Law, and Ethics Regarding U.S. Acquisition and Use of Cyberattack Capabilities* and the 2010 NRC *Proceedings of a Workshop on Detering Cyberattacks: Informing Strategies and Developing Options for U.S. Policy*. Prior to his NRC service, he was a professional staff member and staff scientist for the House Armed Services Committee (1986–1990).

Copyright held by author.

Article development led by **acmqueue**
queue.acm.org

A good user experience depends on predictable performance within the data-center network.

BY DENNIS ABTS AND BOB FELDERMAN

A Guided Tour of Data-Center Networking

THE MAGIC OF the cloud is that it is always on and always available from anywhere. Users have come to expect that services are there when they need them. A data center (or warehouse-scale computer) is the nexus from which all the services flow. It is often housed in a nondescript warehouse-sized building bearing no indication of what lies inside. Amidst the whirring fans and refrigerator-sized computer racks is a tapestry of electrical cables and fiber optics weaving everything together—the data-center network. This article provides a “guided tour” through the principles and central ideas surrounding the network at the heart of a data center—the modern-day loom that weaves the digital fabric of the Internet.

Large-scale parallel computers are grounded in HPC (high-performance computing) where kilo-processor

systems were available 15 years ago. HPC systems rely on fast (low-latency) and efficient interconnection networks capable of providing both high bandwidth and efficient messaging for fine-grained (for example, cache-line size) communication. This zealous attention to performance and low latency migrated to financial enterprise systems where a fraction of a microsecond can make a difference in the value of a transaction.

In recent years, Ethernet networks have made significant inroads into bridging the performance and scalability gap between capacity-oriented clusters built using COTS (commodity-off-the-shelf) components and purpose-built custom system architectures. This is evident from the growth of Ethernet as a cluster interconnect on the Top500 list of most powerful computers (top500.org). A decade ago high-performance networks were mostly custom and proprietary interconnects, and Ethernet was used by only 2% of the Top500 systems. Today, however, more than 42% of the most powerful computers are using Gigabit Ethernet, according to the November 2011 list of Top500 computers. A close second place is InfiniBand, which is used by about 40% of the systems. These standards-based interconnects combined with economies of scale provide the genetic material of a modern data-center network.

A modern data center,^{13,17,24} as shown in Figure 1, is home to tens of thousands of hosts, each consisting of one or more processors, memory, network interface, and local high-speed I/O (disk or flash). Compute resources are packaged into racks and allocated as *clusters* consisting of thousands of hosts that are tightly connected with a high-bandwidth network. While the network plays a central role in the overall system performance, it typically represents only 10%–15% of the cluster cost. Be careful not to confuse cost with value—the network is to a cluster computer what the central nervous system is to the human body.



Figure 1. An example data center and warehouse-scale computer.

Each cluster is homogeneous in both the processor type and speed. The thousands of hosts are orchestrated to exploit thread-level parallelism central to many Internet workloads as they divide incoming requests into parallel subtasks and weave together results from many subtasks across thousands of cores. In general, in order for the request to complete, all parallel subtasks must complete. As a result, the maximum response time of any one subtask will dictate the overall response time.²⁵ Even in the presence of abundant thread-level parallelism, the communication overhead imposed by the network and protocol stack can ultimately limit application performance as the effects of Amdahl's Law² creep in.

The high-level system architecture and programming model shape both the programmer's conceptual view and application usage. The latency and bandwidth "cost" of local (DRAM) and remote (network) memory references

are often baked into the application as programming trade-offs are made to optimize code for the underlying system architecture. In this way, an application organically grows within the confines of the system architecture.

The cluster-application usage model, either dedicated or shared among multiple applications, has a significant impact on SLAs (service-level agreements) and application performance. HPC applications typically use the system in a dedicated fashion to avoid contention from multiple applications and reduce the resulting variation in application performance. On the other hand, Web applications often rely on services sourced from multiple clusters, where each cluster may have several applications simultaneously running to increase overall system utilization. As a result, a data-center cluster may use virtualization for both performance and fault isolation, and Web applications are programmed with this sharing in mind.

Web applications such as search, email, and document collaboration are scheduled resources and run within a cluster.^{4,8} User-facing applications have soft real-time latency guarantees or SLAs that the application must meet. In this model, an application has approximately tens of milliseconds to reply to the user's request, which is subdivided and dispatched to worker threads within the cluster. The worker threads generate replies that are aggregated and returned to the user. Unfortunately, if a portion of the workflow does not execute in a timely manner, then it may exceed a specified timeout delay—as a result of network congestion, for example—and consequently some portion of the coalesced results will be unavailable and thus ignored. This needlessly wastes both CPU cycles and network bandwidth, and may adversely impact the computed result.

To reduce the likelihood of congestion, the network can be overprovi-

sioned by providing ample bandwidth for even antagonistic traffic patterns. Overprovisioning within large-scale networks is prohibitively expensive. Alternatively, implementing QoS (quality of service) policies to segregate traffic into distinct classes and provide performance isolation and high-level traffic engineering is a step toward ensuring application-level SLAs are satisfied. Most QoS implementations are implemented by switch and NIC (network interface controller) hardware where traffic is segregated based on priority explicitly marked by routers and hosts or implicitly steered using port ranges. The goal is the same: a high-performance network that provides predictable latency and bandwidth characteristics across varying traffic patterns.

Data-Center Traffic

Traffic within a data-center network is often measured and characterized according to *flows*, which are sequences of packets from a source to destination host. When referring to Internet protocols, a flow is further refined to include a specific source and destination port number and transport type—UDP or TCP, for example. Traffic is asymmetric with client-to-server requests being abundant but generally small. Server-to-client responses, however, tend to be larger flows; of course, this, too,

depends on the application. From the purview of the cluster, Internet traffic becomes highly aggregated, and as a result the *mean* of traffic flows says very little because aggregated traffic exhibits a high degree of variability and is non-Gaussian.¹⁶

As a result, a network that is only 10% utilized can see lots of packet discards when running a Web search. To understand individual flow characteristics better, applications are instrumented to “sample” messages and derive a distribution of traffic flows; this knowledge allows you to infer a taxonomy of network traffic and classify individual flows. The most common classification is bimodal, using the so-called “elephant” and “mice” classes.

Elephant flows have a large number of packets and are generally long lived; they exhibit “bursty” behavior with a large number of packets injected in the network over a short time. Traffic within a flow is generally ordered, which means elephant flows can create a set of “hotspot” links that can lead to tree saturation or discarded packets in networks that use lossy flow control. The performance impact from elephant flows can be significant. Despite the relatively low number of flows—say less than 1%—they can account for more than half the data volume on the network.

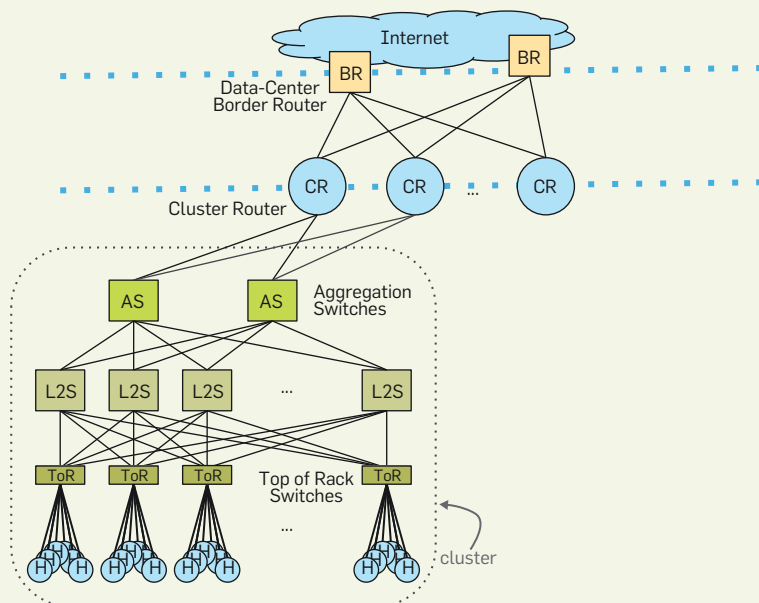
The transient load imbalance induced by elephant flows can adversely affect any innocent-bystander flows that are patiently waiting for a heavily utilized link common to both routes. For example, an elephant flow from A to B might share a common link with a flow from C to D. Any long-lived contention for the shared link increases the likelihood of discarding a packet from the C-to-D flow. Any packet discards will result in an unacknowledged packet at the sender’s transport layer and be retransmitted when the timeout period expires. Since the timeout period is generally one or two orders of magnitude more than the network’s round-trip time, this additional latency²² is a significant source of performance variation.³

Today’s typical multitiered data-center network²³ has a significant amount of *oversubscription*, where the hosts attached to the rack switch (that is, first tier) have significantly more—say an order of magnitude more—provisioned bandwidth between one another than they do with hosts in *other* racks. This *rack affinity* is necessary to reduce network cost and improve utilization. The traffic intensity emitted by each host fluctuates over time, and the transient load imbalance that results from this varying load can create contention and ultimately result in discarded packets for flow control. Traffic *between* clusters is typically less time critical and as a result can be staged and scheduled. Inter-cluster traffic is less orchestrated and consists of much larger payloads, whereas intra-cluster traffic is often fine-grained with bursty behavior. At the next level, between data centers, bandwidth is often very expensive over vast distances with highly regulated traffic streams and patterns so that expensive links are highly utilized. When congestion occurs the most important traffic gets access to the links. Understanding the granularity and distribution of network flows is essential to capacity planning and traffic engineering.

Data-Center Network Architecture

The network topology describes precisely how switches and hosts are interconnected. This is commonly represented as a graph in which vertices represent switches or hosts, and links

Figure 2. A conventional tree-like data-center network topology.



are the edges that connect them. The topology is central to both the performance and cost of the network. The topology affects a number of design trade-offs, including performance, system packaging, path diversity, and redundancy, which, in turn, affect the network's resilience to faults, average and maximum cable length, and, of course, cost.¹² The *Cisco Data Center Infrastructure 3.0 Design Guide*⁶ describes common practices based on a tree-like topology¹⁵ resembling early telephony networks proposed by Charles Clos,⁷ with bandwidth aggregation at different levels of the network.

A fat-tree or folded-Clos topology, similar to that shown in Figure 2, has an aggregate bandwidth that grows in proportion to the number of host ports in the system. A *scalable* network is one in which increasing the number of ports in the network should linearly increase the delivered bisection bandwidth. Scalability and reliability are inseparable since growing to large system size requires a robust network.

Network addressing. A host's *address* is how endpoints are identified in the network. Endpoints are distinguished from intermediate switching elements traversed en route since messages are created by and delivered to an endpoint. In the simplest terms, the address can be thought of as the numerical equivalent of a host name similar to that reported by the Unix `hostname` command.

An address is unique and must be represented in a canonical form that can be used by the *routing function* to determine where to route a packet. The switch inspects the packet header corresponding to the layer in which routing is performed—for example, IP address from layer 3 or Ethernet address from layer 2. Switching over Ethernet involves ARP (address resolution protocol) and RARP (reverse address resolution protocol) that broadcast messages on the layer 2 network to update local caches mapping layer 2 to layer 3 addresses and vice versa. Routing at layer 3 requires each switch to maintain a subnet *mask* and assign IP addresses statically or disseminate host addresses using DHCP (dynamic host configuration protocol), for example.

The layer 2 routing tables are automatically populated when a switch is



The high-level system architecture and programming model shape both the programmer's conceptual view and application usage.



plugged in and learns its identity and exchanges route information with its peers; however, the capacity of the packet-forwarding tables is limited to, say, 64K entries. Further, each layer 2 switch will participate in an STP (spanning tree protocol) or use the TRILL (transparent interconnect of lots of links) link-state protocol to exchange routing information and avoid transient routing loops that may arise while the link state is exchanged among peers. Neither layer 2 nor layer 3 routing is perfectly suited to data-center networks, so to overcome these limitations many new routing algorithms have been proposed (for example, Portland^{1,18} and VL2¹¹).

Routing. The *routing* algorithm determines the path a packet traverses through the network. A packet's route, or path, through the network can be asserted when the message is created, called source routing, or may be asserted hop by hop in a distributed manner as a packet visits intermediate switches. Source routing requires that every endpoint know the prescribed path to reach all other endpoints, and each source-routed packet carries the full information to determine the set of port/link traversals from source to destination endpoint. As a result of this overhead and inflexible fault handling, source-routed packets are generally used only for topology discovery and network initialization, or during fault recovery when the state of a switch is unknown. A more flexible method of routing uses distributed *lookup tables* at each switch, as shown in Figure 3.

For example, consider a typical Ethernet switch. When a packet arrives at a switch input port, it uses fields from the packet header to index into a lookup table and determine the next hop, or egress port, from the current switch.

A good topology will have abundant *path diversity* in which multiple possible egress ports may exist, with each one leading to a distinct path. Path diversity in the topology may yield ECMP (equal-cost multipath) routing; in that case the routing algorithm attempts to load-balance the traffic flowing across the links by spreading traffic uniformly. To accomplish this *uniform spreading*, the routing function in the switch will *hash* several fields of the packet header to produce a deterministic egress port.

In the event of a link or switch failure, the routing algorithm will take advantage of path diversity in the network to find another path.

A path through the network is said to be *minimal* if no shorter (that is, fewer hops) path exists; of course, there may be multiple minimal paths. A fat-tree topology,¹⁵ for example, has multiple minimal paths between any two hosts, but a butterfly topology⁹ has only a single minimal path between any two hosts. Sometimes selecting a *non-minimal* path is advantageous—for example, to avoid congestion or to route around a fault. The length of a non-minimal path can range from $\text{min}+1$ up to the length of a Hamiltonian path visiting each switch exactly once. In general, the routing algorithm might consider non-minimal paths of a length that is one more than a minimal path, since considering *all* non-minimal paths would be prohibitively expensive.

Network Performance

Here, we discuss the etiquette for sharing the network resources—specifically, the physical links and buffer spaces are resources that require *flow control* to share them efficiently. Flow control is carried out at different levels of the network stack: data-link, network, transport layer, and possibly within the application itself for explicit coordination of resources. Flow control that occurs at lower levels of the communication stack is transparent to applications.

Flow control. Network-level flow control dictates how the input buffers at each switch or NIC are managed: store-and-forward, virtual cut-through,¹⁴ or wormhole,¹⁹ for example. To understand the performance implications of flow control better, you must first understand the total delay, T , a packet incurs:

$$T = H(t_r + Lt_p) + t_s$$

H is the number of *hops* the packet takes through the network; t_r is the fall-through latency of the switch, measured from the time the first flit (flow-control unit) arrives to when the first flit exits; and t_p is the propagation delay through average cable length L . For short links—say, fewer than 10

meters—electrical signaling is cost-effective. Longer links, however, require fiber optics to communicate over the longer distances. Signal propagation in electrical signaling (5 nanoseconds per meter) is faster than it is in fiber (6 nanoseconds per meter).

Propagation delay through electrical cables occurs at subluminal speeds because of a frequency-dependent component at the surface of the conductor, or “skin effect,” in the cable. This limits the signal velocity to about three-quarters the speed of light in a vacuum. Signal propagation in optical fibers is even slower because of dielectric waveguides used to alter the refractive index profile so that higher-velocity components of the signal (such as shorter wavelengths) will travel longer distances and arrive at the same time as lower-velocity components, limiting the signal velocity to about two-thirds the speed of light in a vacuum. Optical signaling must also account for the time necessary to perform electrical-to-optical signal conversion, and vice versa.

The *average* cable length, L , is largely determined by the topology and the physical placement of system racks within the data center. The packet’s serialization latency, t_s , is the time necessary to squeeze the packet onto a narrow serial channel and is determined by the *bit rate* of the channel. For example, a 1,500-byte Ethernet packet (frame) requires more than $12\mu\text{s}$ (ignoring any interframe gap time) to be squeezed onto a 1Gb/s link. With store-and-forward flow control, as its name suggests, a packet is buffered at each hop before the switch does anything with it:

$$T_{\text{sf}} = H(t_r + Lt_p + t_s)$$

As a result, the serialization delay, t_s , is incurred at *each* hop, instead of just at the destination endpoint as is the case with virtual cut-through and wormhole flow control. This can potentially add on the order of $100\mu\text{s}$ to the round-trip network delay in a data-center network.

A *stable* network monotonically delivers messages as shown by a characteristic throughput-load curve in Figure 4. In the absence of end-to-end flow control, however, the network

can become unstable, as illustrated by the dotted line in the figure, when the offered load exceeds the saturation point, α . The saturation point is the offered load beyond which the network is said to be *congested*. In response to this congestion, packets may be discarded to avoid overflowing an input buffer. This *lossy* flow control is commonplace in Ethernet networks.

Discarding packets, while conceptually simple and easy to implement, puts an onus on transport-level mechanisms to detect and retransmit lost packets. Note, packets that are lost or corrupted during transmission are handled by the same transport-level reliable delivery protocol. When the offered load is low (less than α), packet loss as a result of corruption is rare, so paying the relatively large penalty for transport-level retransmission is generally tolerable. Increased traffic (greater than α) and adversarial traffic patterns will cause packet discards after the switch’s input queue is exhausted. The resulting retransmission will only further exacerbate an already congested network, yielding an unstable network that performs poorly, as shown by the dotted line in Figure 4. Alternatively, with *lossless* flow control, when congestion arises packets may be blocked or held at the source until resources are available.

A *global congestion control* mechanism prevents the network from operating in the post-saturation region. Most networks use *end-to-end flow control*, such as TCP,⁵ which uses a windowing mechanism between pairs of source and sink in an attempt to match the source’s injection rate with the sink’s acceptance rate. TCP, however, is designed for reliable packet delivery, not necessarily timely packet delivery, and as a result, requires tuning (TCP congestion-control algorithms will auto-tune to find the right rate) to balance performance and avoid unnecessary packet duplication from eagerly retransmitting packets under heavy load.

Improving the network stack. Several decades ago the network designers of early workstations made trade-offs that led to a single TCP/IP/Ethernet network stack, whether communicating over a few meters or a few kilometers. As processor speed and density improved, the cost of network communication grew

relative to processor cycles, exposing the network stack as a critical latency bottleneck.²² This is, in part, the result of a user-kernel context switch in the TCP/IP/Ethernet stack—and possibly additional work to copy the message from the application buffer into the kernel buffer and back again at the receiver. A two-pronged hardware/software approach tackled this latency penalty: OS bypass, and zero copy, both of which are aimed at eliminating the user-kernel switch for every message and avoiding a redundant memory copy by allowing the network transport to grab the message payload directly from the user application buffers.

To ameliorate the performance impact of a user/kernel switch, OS bypass can be used to deposit a message directly into a user-application buffer. The application participates in the messaging protocol by spin-waiting on a doorbell memory location. Upon arrival, the NIC deposits the message contents in the user-application buffer, and then “rings” the doorbell to indicate message arrival by writing the offset into the buffer where the new message can be found. When the user thread detects the updated value, the incoming message is processed entirely from user space.

Zero-copy message-passing protocols avoid this additional memory copy from user to kernel space, and vice versa at the recipient. An interrupt signals the arrival of a message, and an interrupt handler services the new message and returns control to the user application. The interrupt latency—the time from when the interrupt is raised until control is handed to the interrupt handler—can be significant, especially if interrupt coalescing is used to amortize the latency penalty across multiple interrupts. Unfortunately, while interrupt coalescing improves message efficiency (that is, increased effective bandwidth), it does so at the cost of both increased message latency and latency variance.

Scalable, Manageable, And Flexible

In general, *cloud computing* requires two types of services: user-facing computation (for example, serving Web pages) and inward computation (for example, indexing, search, and map/reduce). Outward-facing functionality

can sometimes be done at the “border” of the Internet where commonly requested pages are cached and serviced by edge servers, while inward computation is generally carried out by a cluster in a data center with tightly coupled, orchestrated communication. User demand is diurnal for a geographic region; thus, multiple data centers are positioned around the globe to accommodate the varying demand. When possible, demand may be spread across nearby data centers to load-balance the traffic.

The sheer enormity of this computing infrastructure makes nimble deployment very challenging. Each cluster is built up rack by rack and tested as units (rack, top-of-rack switch, among others), as well as in its entirety with production-level workloads and traffic intensity.

The cluster ecosystem undergoes organic growth over its life span, propelled by the rapid evolution of soft-

ware—both applications and, to a lesser extent, the operating system. The fluid-like software demands of Web applications often consume the cluster resources that contain them, making *flexibility* a top priority in such a fluid system. For example, adding 10% additional storage capacity should mean adding no more than 10% more servers to the cluster. This linear growth function is critical to the *scalability* of the system—adding fractionally more servers results in a commensurate growth in the overall cluster capacity. Another aspect of this flexibility is the *granularity* of resource additions, which is often tied to the cluster packaging constraints. For example, adding another rack to a cluster, with, say, 100 new servers, is more manageable than adding a whole row, with tens of racks, on the data-center floor.

Even a modest-sized cluster will have several kilometers of fiber-optic cable acting as a vast highway inter-

Figure 3. Example packet routing through a switch chip.

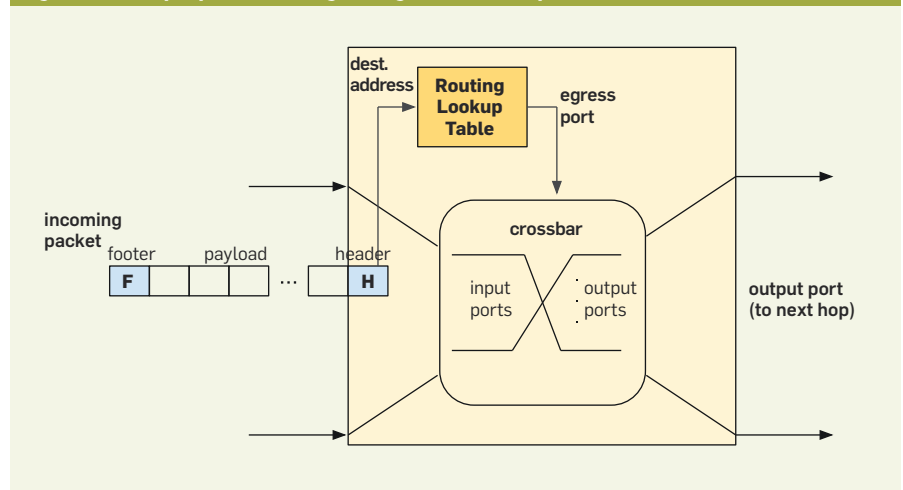
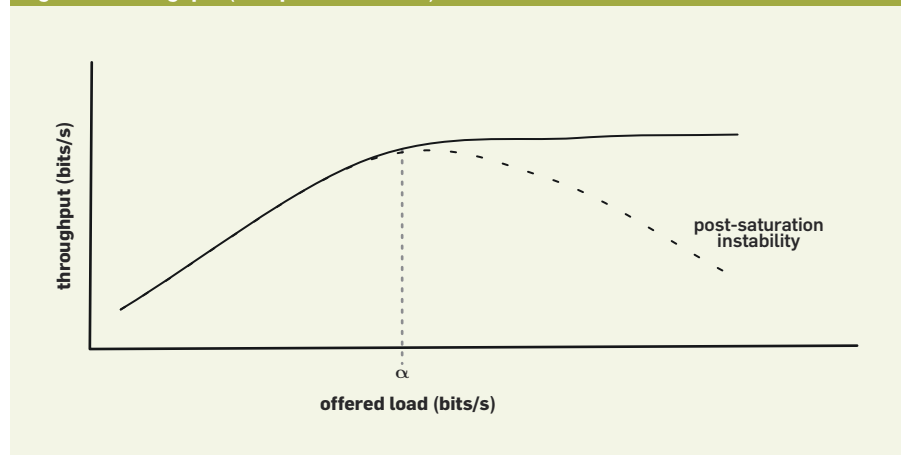


Figure 4. Throughput (accepted bandwidth) as load varies.



connecting racks of servers organized as multiple rows on the data-center floor. The data-center network topology and resulting cable complexity is “baked in” and remains a rigid fixture of the cluster. Managing cable complexity is nontrivial, which is immediately evident from the intricately woven tapestry of fiber-optic cabling laced throughout the data center. It is not uncommon to run additional fiber for redundancy, in the event of a cable failure in a “bundle” of fiber or for planned bandwidth growth. Fiber cables are carefully measured to allow some slack and to satisfy the cable’s bend radius, and they are meticulously labeled to make troubleshooting less of a needle-in-a-haystack exercise.

Reliable and Available

Abstraction is the Archimedes lever that lifts many disciplines within computer science and is used extensively in both computer system design and software engineering. Like an array of nested Russian dolls, the network-programming model provides abstraction between successive layers of the networking stack, enabling platform-independent access to both data and system management. One such example of this type of abstraction is the *protocol buffer*,²¹ which provides a structured message-passing interface for Web applications written in C++, Java, or Python.

Perhaps the most common abstraction used in networking is the notion of a *communication channel* as a virtual resource connecting two hosts. The TCP communication model provides this abstraction to the programmer in the form of a file descriptor, for example, where reads and writes performed on the socket result in the corresponding network transactions, which are hidden from the user application. In much the same way, the InfiniBand QP (queue-pair) verb model provides an abstraction for the underlying send/receive hardware queues in the NIC. Besides providing a more intuitive programming interface, abstraction also serves as a protective sheath around software when faults arise, depositing layers of software sediment to insulate it from critical faults (for example, memory corruption or, worse, host power-supply failure).

The data-center network serves as a “central nervous system” for information exchange between cooperating tasks.

Bad things happen to good software. Web applications must be designed to be fault aware and, to the extent possible, resilient in the presence of a variety of failure scenarios.¹⁰ The network is responsible for the majority of the unavailability budget for a modern cluster. Whether it is a rogue gamma ray causing a soft error in memory or an inattentive worker accidentally unearthing a fiber-optic line, the operating system and underlying hardware substrate work in concert to foster a robust ecosystem for Web applications.

The data-center network serves as a “central nervous system” for information exchange between cooperating tasks. The network’s functionality is commonly divided into *control* and *data* planes. The control plane provides an ancillary network juxtaposed with the data network and tasked with “command and control” for the primary data plane. The control plane is an autonomic system for configuration, fault detection and repair, and monitoring of the data plane. The control plane is typically implemented as an embedded system within each switch component and is tasked with fault detection, notification, and repair when possible.

For example, when a network link fails or has an uncharacteristically high number of transmission errors, the control plane will *reroute* the network to avoid the faulty link. This entails recomputing the routes according to the *routing algorithm* and emplacing new entries in the routing tables of the affected switches. Of course, the effects of this patchwork are not instantaneous. Once the routing algorithm computes new routes, taking into consideration the newfound faulty links, it must disseminate the routes to the affected switches. The time needed for this information exchange is referred to as *convergence time*, and a primary goal of the routing protocol is to ensure it is optimally confined to a small epoch.

Fault recovery is a very complicated subject and confounds all but the simplest of data-center networks. Among the complicating factors are *marginal* links that cause “flapping” by transitioning between active and inactive (that is, up and down), repeatedly creat-

ing a deluge of error notifications and resulting route recomputation based on fluctuating and inconsistent link status. Some link-layer protocols allow the link speed to be adjusted downward in hopes of improving the link quality. Of course, lowering the link speed results in a reduced bandwidth link, which in turn may limit the overall bandwidth of the network or at the very least will create load imbalance as a result of increased contention across the slow link. Because of these complicating factors, it is often better to logically excise the faulty link from the routing algorithm until the physical link can be replaced and validated.

Conclusion

The data-center network is generally regarded as a critical design element in the system architecture and the skeletal structure upon which processor, memory, and I/O devices are dynamically shared. The evolution from 1G to 10G Ethernet and the emerging 40G Ethernet has exposed performance bottlenecks in the communication stack that require better hardware-software coordination for efficient communication. Other approaches by Solarflare, Myricom, and InfiniBand, among others, have sought to reshape the conventional sockets programming model with more efficient abstractions. Internet sockets, however, remain the dominant programming interface for data-center networks.

Network performance and reliability are key design goals, but they are tempered by cost and serviceability constraints. Deploying a large cluster computer is done incrementally and is often limited by the power capacity of the building, with power being distributed across the cluster network so that a power failure impacts only a small fraction—say, less than 10%—of the hosts in the cluster. When hardware fails, as is to be expected, the operating system running on the host coordinates with a higher-level hypervisor or cluster operating system to allow failures to be replaced in situ without draining traffic in the cluster. Scalable Web applications are designed to expect occasional hardware failures, and the resulting software is by necessity resilient.

A good user experience relies on *predictable* performance, with the data-center network delivering predictable latency and bandwidth characteristics across varying traffic patterns. With single-thread performance plateauing, microprocessors are providing more cores to keep pace with the relentless march of Moore's Law. As a result, applications are looking for increasing thread-level parallelism and scaling to a large core count with a commensurate increase in communication among cores. This trend is motivating *communication-centric* cluster computing with tens of thousands of cores in unison, reminiscent of a flock darting seamlessly amidst the clouds. **□**

References

1. Al-Fares, M., Loukissas, A. and Vahdat, A. A scalable, commodity data-center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication* (2008), 63–74; <http://doi.acm.org/10.1145/1402958.1402967>.
2. Amdahl's Law; http://en.wikipedia.org/wiki/Amdahl's_Law.
3. Ballani, H., Costa, P., Karagiannis, T. and Rowstron, A. Towards predictable data-center networks. In *Proceedings of the ACM SIGCOMM 2011 Conference* (2011), 242–253; <http://doi.acm.org/10.1145/2018436.2018465>.
4. Barroso, L.A., Dean, J. and Holze, U. Web search for a planet: The Google cluster architecture. *IEEE Micro* 23, 2 (2003), 22–28; <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1196112&isnumber=26907>.
5. Cerf, V. and Icahn R.E. A protocol for packet network intercommunication. *SIGCOMM Computer Communication Review* 35, 2 (2005), 71–82; <http://doi.acm.org/10.1145/1064413.1064423>.
6. Cisco Data Center Infrastructure 3.0 Design Guide. Data Center Design—IP Network Infrastructure; http://www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/DC_3_0/DC-3_0_IPInfra.html.
7. Clos, C. A Study of non-blocking switching networks. *The Bell System Technical Journal* 32, 2 (1953), 406–424.
8. Fitzpatrick, B. Distributed caching with Memcached. *Linux Journal* 2004; <http://www.linuxjournal.com/article/7451>.
9. Dally, W. and Towles, B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, San Francisco, CA, 2003.
10. Gill, P., Jain, N. and Nagappan, N. Understanding network failures in data centers: measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 Conference* (2011), 350–361; <http://doi.acm.org/10.1145/2018436.2018477>.
11. Greenberg, A., Hamilton, J.R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P. and Sengupta, S. VL2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication* (2009): 51–62; <http://doi.acm.org/10.1145/1592568.1592576>.
12. Greenberg, A., Hamilton, J., Maltz, D.A. and Patel, P. The cost of a cloud: Research problems in data center networks. *SIGCOMM Computer Communications Review* 39, 1 (2008), 68–73; <http://doi.acm.org/10.1145/1496091.1496103>.
13. Hoelzle, U. and Barroso, L. A. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines* (1st ed.). Morgan & Claypool Publishers, 2009.
14. Kermani, P. and Kleinrock, L. Virtual cut-through: A new computer communication switching technique. *Computer Networks* 3, 4 (1976), 267–286; <http://www.sciencedirect.com/science/article/pii/0376507579900321>.
15. Leiserson, C.E. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers* 34, 10 (1985), 892–901.

16. Mori, T., Uchida, M., Kawahara, R., Pan, J., and Goto, S. Identifying elephant flows through periodically sampled packets. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement* (2004); 115–120; <http://doi.acm.org/10.1145/1028788.1028803>.
17. Mudigonda, J., Yalagandula, P., Mogul, J., Stiekes, B., Pouffary, Y. NetLord: A scalable multi-tenant network architecture for virtualized datacenters. *SIGCOMM Computer Communication Review* 41, 4 (2011), 62–73; <http://doi.acm.org/10.1145/2043164.2018444>.
18. Mysore, R.N., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., Subramanya, V., Vahdat, A. PortLand: A scalable fault-tolerant layer 2 data center network fabric. *SIGCOMM Computer Communication Review* 39, 4 (2009), 39–50; <http://doi.acm.org/10.1145/1594977.1592575>.
19. Ni, L. M., McKinley, P. K. A survey of wormhole routing techniques in direct networks. *Computer* 26, 2 (1993), 62–76; <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=191995&isnumber=4947>.
20. Ousterhout, J., Agrawal, P., Erickson, D., Kozyrak, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Parulkar, G., Rosenblum, M., Rumble, S.M., Stratmann, E. and Stutsman, R. The case for RAMClouds: Scalable high-performance storage entirely in DRAM. *SIGOPS Operating Systems Review* 43, 4 (2010), 92–105; <http://doi.acm.org/10.1145/1713254.171327>.
21. Protocol buffers; <http://code.google.com/apis/protocolbuffers/>.
22. Rumble, S.M., Ongaro, D., Stutsman, R., Rosenblum, M., and Ousterhout, J.K. It's time for low latency. In *Proceedings of the 13th Usenix Conference on Hot Topics in Operating Systems* (2011).
23. Vahdat, A., Al-Fares, M., Farrington, N., Mysore, R.N., Porter, G., and Radhakrishnan, S. Scale-out networking in the data center. *IEEE Micro* 30, 4 (2010), 29–41; <http://dx.doi.org/10.1109/MM.2010.72>.
24. Vahdat, A., Liu, H., Zhao, X. and Johnson, C. The emerging optical data center. Presented at the *Optical Fiber Communication Conference*. OSA Technical Digest (2011); <http://www.opticsinfobase.org/abstract.cfm?URI=OFC-2011-OTuH2>.
25. Wilson, C., Ballani, H., Karagiannis, T., and Rowstron, A. Better never than late: Meeting deadlines in datacenter networks. In *Proceedings of the ACM SIGCOMM 2011 Conference* (2011), 50–61; <http://doi.acm.org/10.1145/2018436.2018443>.

Related articles on queue.acm.org

Enterprise Grid Computing

Paul Strang

<http://queue.acm.org/detail.cfm?id=1080877>

Cooling the Data Center

Andy Woods

<http://queue.acm.org/detail.cfm?id=1737963>

Improving Performance on the Internet

Tom Leighton

<http://queue.acm.org/detail.cfm?id=1466449>

Dennis Abts is a member of the technical staff at Google, where he is involved in the architecture and design of next-generation large-scale clusters. Prior to joining Google, Abts was a senior principal engineer and system architect at Cray Inc. He has numerous technical publications and patents in areas of interconnection networks, data-center networking, cache-coherence protocols, high-bandwidth memory systems, and supercomputing.

Bob Felderman spent time at both Princeton and UCLA before starting a short stint at Information Sciences Institute. He then helped found Myricom, which became a leader in cluster-computing networking technology. After seven years there, he moved to Packet Design where he applied high-performance computing ideas to the IP and Ethernet space. He later was a founder of Precision I/O. All of that experience eventually led him to Google where he is a principal engineer working on issues in data-center networking and general platforms system architecture.

© 2012 ACM 0001-0782/12/06 \$10.00

Article development led by [acmqueue](http://acmqueue.queue.acm.org)
queue.acm.org

Understanding the world from the sea of online photos.

BY DAVID CRANDALL AND NOAH SNAVELY

Modeling People and Places with Internet Photo Collections

COMPUTATIONAL PHOTOGRAPHY OFTEN considers sets of photos taken by a single user in a single setting, but the popularity of online social media sites has created a social aspect to photo collections as well. Photo-sharing sites such as Flickr and Facebook contain vast amounts of latent information about our world and human behavior. Our recent work has involved building automatic algorithms that analyze large collections of imagery in order to understand and model people and places at a global scale. Geotagged photographs can be used to identify the most photographed places on Earth, as well as to infer the names and visual representations of these places.

At a local scale, we can build detailed three-dimensional models of a scene by combining information from thousands of two-dimensional photographs taken by different people and from different vantage points. One key representation for many of these tasks is a network: a graph linking photos by visual similarity or other measures.

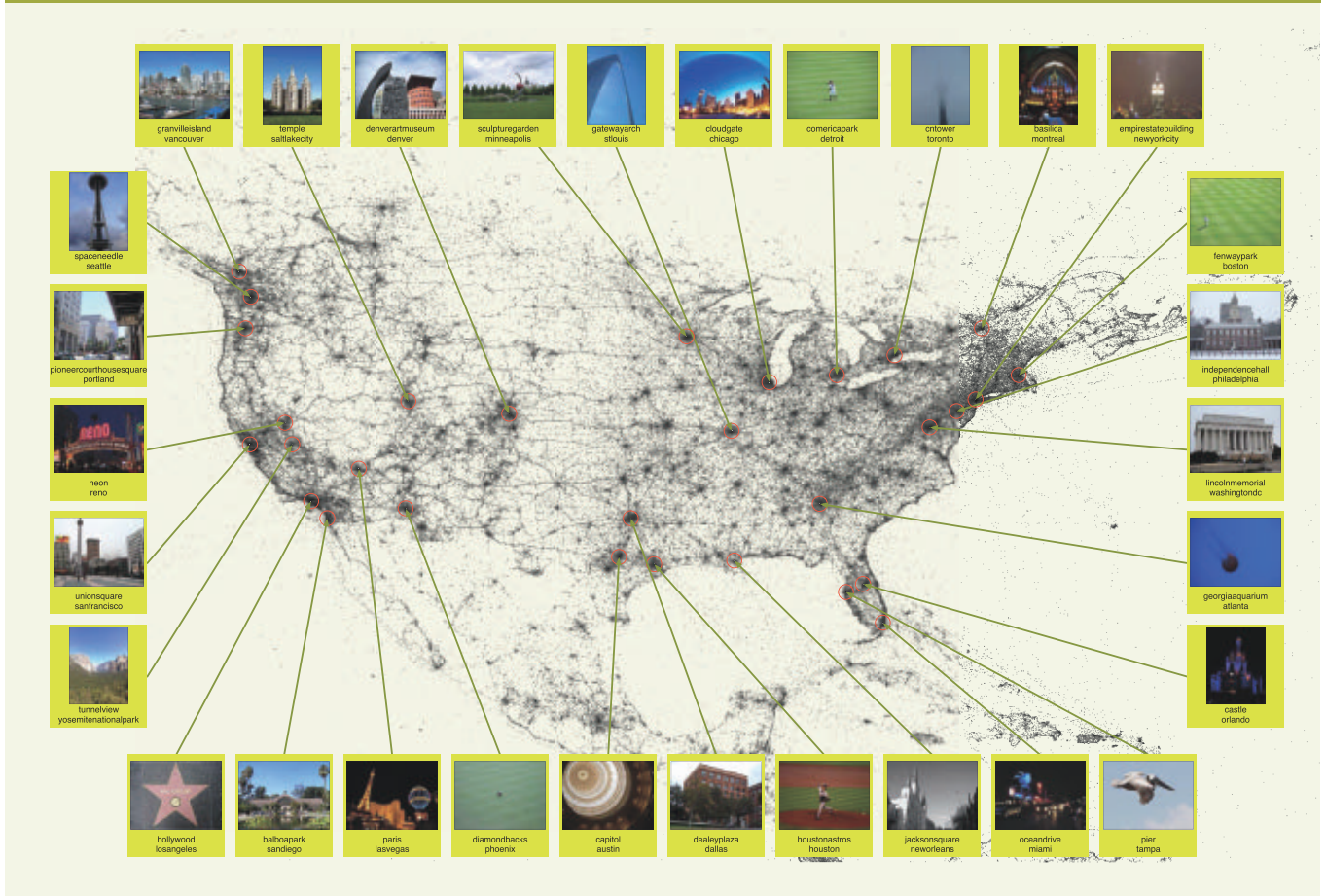
This article describes our work in using online photo collections to reconstruct information about the world and its inhabitants at both global and local scales. This work has been driven by the dramatic growth of social content-sharing Web sites, which have created immense online collections of user-generated visual data. Flickr.com alone currently hosts more than six billion images taken by more than 40 million unique users,¹¹ while Facebook.com has said it grows by nearly 250 million photos every day.²⁰

While users of these sites are primarily motivated by a desire to share photos with family and friends, collectively they are generating vast repositories of online information about the world and its people. Each of these photos is a visual observation of what a small part of the world looked like at a particular point in time and space. It is also a record of *where* a particular person (the photographer) was at a moment in time and *what* he or she was paying attention to.

In aggregate, and in combination with nonvisual metadata available on photo-sharing sites (including photo timestamps, geotags, captions, user profiles, and social contacts), these billions of photos present a rich source of information about the state of the world and the behavior of its people. We can thus imagine extending the domain of computational photography to encompass all of the world's photos, where the goal is to extract useful information about places and people from our collective image data.

We have recently demonstrated how to reconstruct information about the world at both global and local scales using collections such as those on

Figure 1. Annotated map of North America.



Flickr.^{7,9,21} Our algorithms for analyzing the world at a global scale can automatically create annotated world maps by finding the most photographed cities and landmarks, inferring place names from text tags, and analyzing the images themselves to identify “canonical” images to summarize each place. Figures 1 and 2 show examples of such maps. Figure 1 is an annotated map of North America, automatically generated by analyzing nearly 35 million photos from Flickr. For each of the top 30 most photographed cities, the map shows the name of the city inferred from tags, the name of the most photographed landmark, and a representative photo of the landmark. Figure 2 is an automatically generated annotated map of Europe.

This analysis can also generate statistics about places, such as ranking landmarks by their popularity or studying which kinds of users visit which sites. At a more local level, we can use automatic techniques from computer vision to produce strikingly accurate 3D models of a landmark, giv-

ing in a large number of 2D photos taken by many different users from many different vantage points. Figure 3 shows an example 3D reconstruction of the Colosseum created completely automatically from photos harvested from the Internet. This figure shows the 3D model itself, along with the position and orientation of the camera that took each photo. The reconstructed cameras are shown as black wireframe pyramids indicating where each photo was taken, and the Colosseum is reconstructed as a dense 3D point cloud, similar to what a laser scanner would capture—but in this case, reconstructed completely automatically from photos found on the Internet.

Figure 9 illustrates a similar “point cloud” reconstruction of the Old Town of Dubrovnik, created from more than 6,500 Flickr photos, as well as a reconstruction of the Forum from a larger 3D model of Rome.

This work follows an emerging trend in interdisciplinary research connecting computer science to other scientific disciplines. The recent ex-

plosion of publicly available data on the Internet—from Twitter streams, to Wikipedia edit logs, to scans of all of the world’s books¹⁶—is creating an opportunity to revolutionize research in the humanities and social sciences.¹³ This leads to two key research problems in computer science: extracting meaningful semantics from the raw data; and doing so efficiently. Compared with traditional techniques such as surveys and direct measurement, data collection from online social networking sources is of negligible cost and can be conducted at unprecedented scales.

The challenge is that online datasets are largely unstructured and thus require sophisticated algorithms that can organize and extract meaning from noisy data. In our case, this involves developing automated techniques that can find patterns across millions of images. Representing large image collections as networks or graphs, where each image is a node and is connected to related images, can form a useful representation for extracting many

Figure 2. Automatically generated annotated map of Europe.

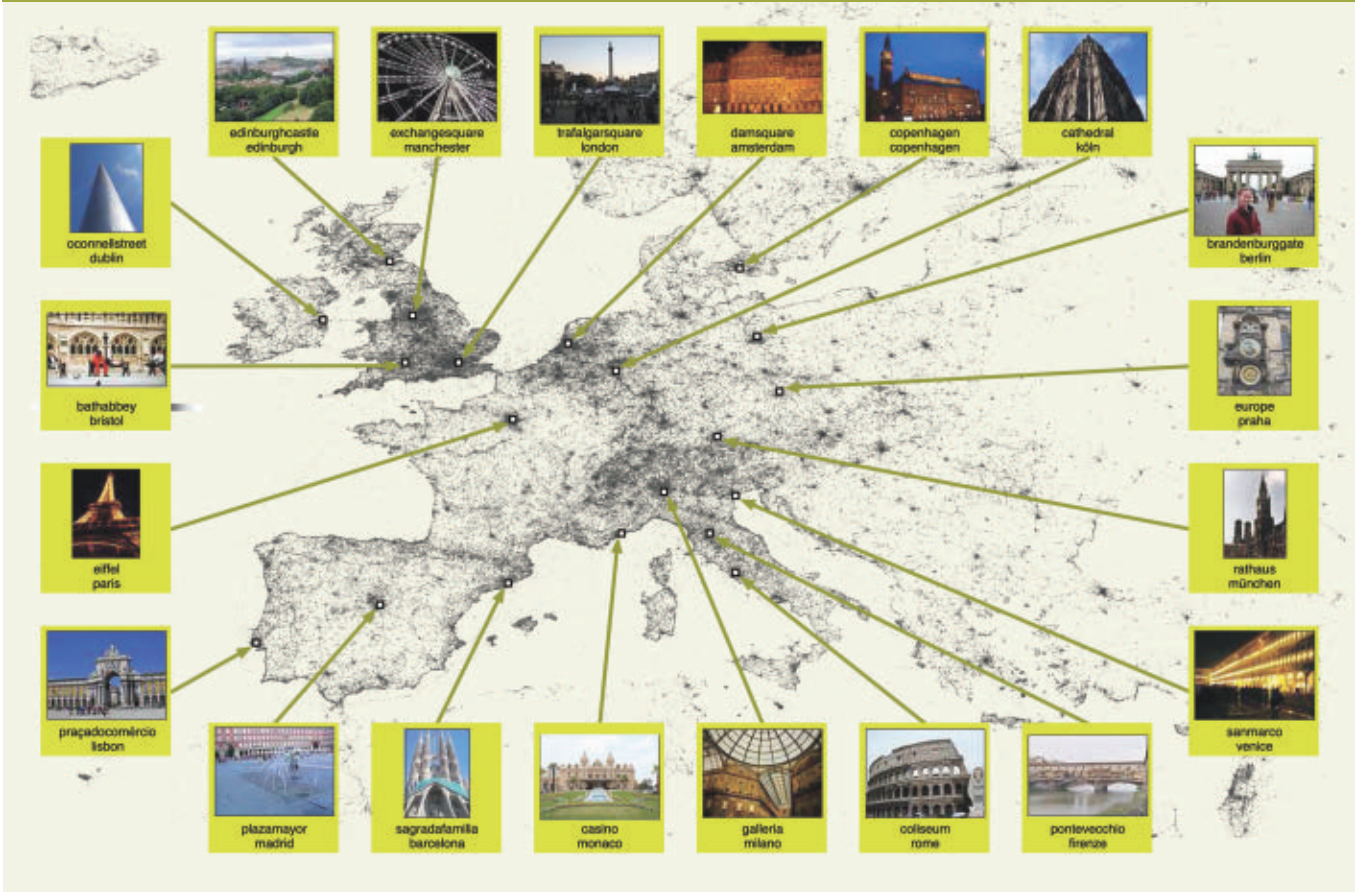


Figure 3. 3D model of a scene (right) generated from photographs (left).



Figure 4. Metadata for a photo downloaded from Flickr.



types of information such as 3D structures or representative views. This observation suggests interesting parallels between image collections and other domains where link structure appears, such as between people in social networks or between pages in the World Wide Web.

Figure 6 presents an example of a visual connectivity network for a set of images of Trafalgar Square. We compute a measure of visual similarity between every pair of images and connect those above a threshold. Many photos are not connected at all; they generally are images of people or objects and not of the square itself. A clustering algorithm finds tightly connected components of the network. This produces three groups of images, each corresponding to a different frequently photographed scene from the square (marked by a dotted blue line in the figure).

Mapping the World

In addition to the images themselves, modern photo-sharing sites such as Flickr collect a rich assortment of non-visual information about photos. Many

online photos have metadata specifying what a photo contains (text tags), as well as where (geotag), when (timestamp), and how (camera metadata such as exposure settings) the photo was taken. On social media sites photos are also accompanied by information generated as a result of sharing, such as text tags, comments, and ratings. The geotagging features of photo-sharing sites are especially useful in our work. These geotags record the latitude and longitude of the point on Earth where a photo was taken. This information either is entered manually by the photographer using a map-based interface or (increasingly) is automatically determined by a GPS (global positioning system) receiver in the camera or cellphone. Figure 4 offers example metadata for a photo downloaded from Flickr,¹⁵ including a geotag specifying latitude and longitude, textual tags, and camera information.

By aggregating this visual and non-visual information from the photographs of many millions of users, we can study what the world looks like in the collective consciousness of the world's photographers. To begin, we collected a dataset of more than 90 million geotagged photos using the Flickr public API.² As one might expect, more photographs are taken in some locations than others. A plot of the geotags in our database, shown in figures 1 and 2, illustrates this nonuniform distribution. This distribution contains significant information beyond the images themselves that is revealed only through analysis of large numbers of photos from many photographers. For example, photo-taking is dense in urban areas and quite sparse in most rural areas. Note that the continental boundaries in these maps are quite sharp, because beaches are such popular locales to take photos. Also note how roads are visible in these maps because people take photos as they travel. In Figure 1 the east-west interstate highways crossing the western U.S. are especially clear.

Given that photographic activity is highly nonuniform, we identify geographic concentrations of photos by using mean shift, a clustering algorithm for finding the peaks of a non-parametric distribution.⁵ We look for peaks at multiple scales (by applying

mean shift with kernels of different sizes), including both city (50km radius) and landmark (100m) scales. We can then rank cities and landmarks based on the number of photos or number of distinct photographers who have uploaded a photo from that place.

For example, the most photographed cities in the world, according to Flickr, are New York, London, San Francisco, Paris, and Los Angeles. The five most photographed landmarks are the Eiffel Tower, Trafalgar Square, Tate Modern, Big Ben, and Notre Dame (more detailed rankings are available online⁸). The techniques used to

produce these rankings are relatively simple, but they are an example of the kinds of analyses that are suddenly possible with the rise of photo-sharing sites. The list of top landmarks includes some surprises; the Apple Store in Manhattan, for example, ranks among the top five landmarks in New York City and is ranked 28th in the entire world.

For each of these highly photographed places, we can automatically infer its name by looking at the text tags that people assign to photographs taken in that place. Although most tags are at best weakly related to geog-

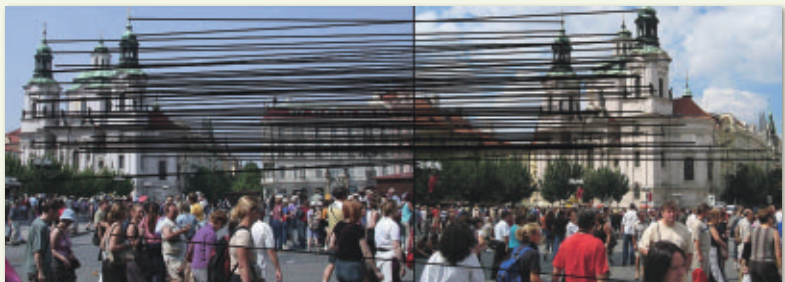
Figure 5. SIFT feature matching.



(a) input photograph.



(b) input photo with extracted SIFT features superimposed.



(c) a pair of overlapping images shown side by side with matching features connected by black lines.

raphy—flower, family, sunset, black and white, among others—we can find place names by looking across the photos of millions of users and finding tags that are used frequently in a particular place and infrequently outside of it. We can also generate a visual description of each place by finding a representative image that summarizes that place well. To do this, we deem each photo-

graph taken in a place as a vote for the most interesting viewpoint at that location. Intuitively, we then try to find the viewpoint that receives the most votes by looking for groups of photos that are visually similar and taken by many different users.

To implement this intuitive approach we construct a graph in which each image from the place is a node,

and we connect pairs of photos having a high degree of visual similarity. Then we apply a graph-clustering algorithm to find tightly connected components of the graph (that is, groups of nodes that are connected to many other nodes within the group but not to many nodes outside the group) and choose one of these photos as a representative image. A sample graph of this type is shown in Figure 6. To decide which nodes to connect, we measure visual similarity using an automated technique called SIFT (scale-invariant feature transform) feature matching,¹⁴ illustrated in Figure 5. Note that this summary image is not necessarily the *best* photo of a particular place—it will likely be a canonical tourist photo rather than a more unusual yet captivating viewpoint captured by a professional photographer.

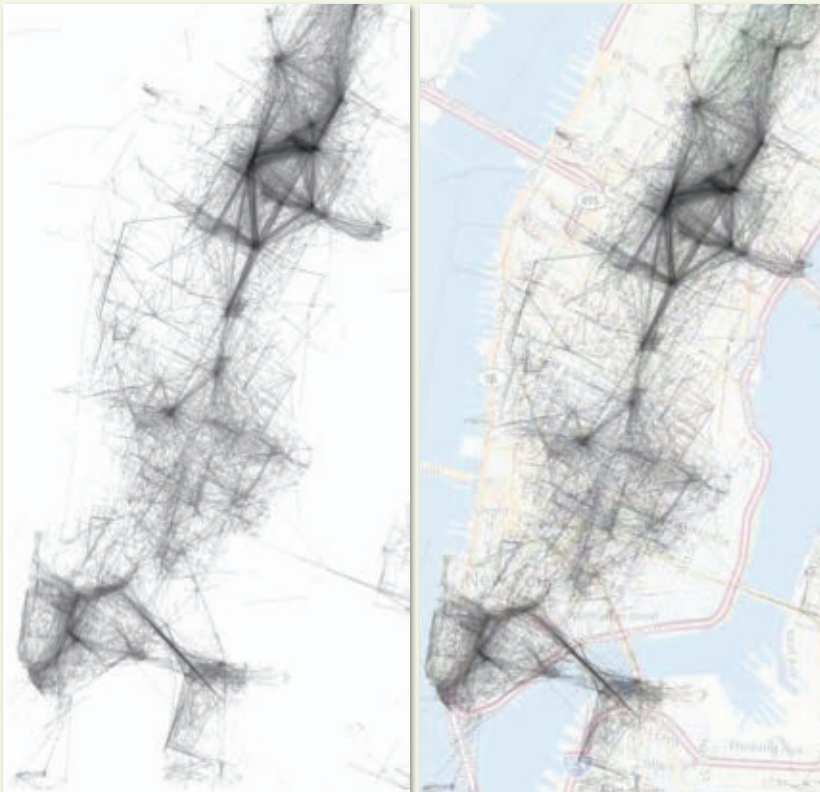
The map in Figure 1 was produced completely automatically using this analysis on tens of millions of images downloaded from Flickr. Starting with a blank slate, we plotted the raw photo geotags to produce the map in the background and then applied mean-shift clustering to locate the 30 most photographed cities on Earth. For each of those cities, we extracted the city's name by looking for distinctive text tags and found the name of the most photographed landmark within the city. Then we extracted a representative image for that landmark. While the analysis is not perfect—a human would have chosen a more appropriate image of Phoenix than a bird on a baseball field, for example—the result is a compelling summary of North America, produced automatically by analyzing the activity of millions of Flickr users. Maps for other continents, regions, and cities of the world are available at our project Web site.⁸

This analysis is reminiscent of sociologist Stanley Milgram's work during the 1970s studying people's "psychological maps"—their mental images of how the physical world is laid out.¹⁷ He asked Parisians to draw freehand maps of their city and then compared these maps with the factual geography. Milgram found that the maps were highly variable and largely inaccurate but that most people tended to anchor their maps around a few key landmarks such as the Seine River and Notre

Figure 6. Photo network for finding representative images.



Figure 7. Trails of human movement in Manhattan.



Dame Cathedral. He ranked landmarks by their degree of importance in the collective Parisian psychology by counting the number of times that each landmark was mentioned in the study. Our work is an analogous study, at a much larger scale. It is important to note that we are also dealing with much less controlled data, however, and our results are biased by the demographics of Flickr users.

Data from Flickr can also be used to study the behavior of human photographers, because each photo is an observation of what a particular user was doing at a particular moment in time. For example, studying sequences of geotagged, timestamped photos can track the paths that people take. Figure 7 shows an example of this analysis for Manhattan. Note that the grid structure of the streets and avenues is clearly visible, as are popular tourist paths such as the walk across the Brooklyn Bridge and the ferries leaving the southern tip of the island. We used this data to study the relationship between human mobility patterns and the social network defined on Flickr.⁶ We can infer a user's social network with startling accuracy based only on such patterns. After observing that two people were at about the same place at about the same time on five distinct occasions, for example, the probability that they are friends is nearly 60%.

This illustrates how data from online social-sharing systems can be used to study questions in sociology at a scale that has never before been possible. It also reveals a potential privacy concern with geotagging-enabled social-networking sites, in that users can reveal more information about themselves than they intend to, such as the identity of their friends.

Reconstructing Landmarks

Thus far, our visual representation of a landmark has simply been a single image that is visually similar to many other images taken at that site. For popular landmarks, however, thousands of online photos are taken by different users, each with a different composition and from a different viewpoint. Each of these photos is thus a slightly different 2D observation of a 3D scene. This leads to the idea of using computer vision algorithms to re-



Internet photo collections pose new challenges to computer vision because of their extreme scale and unstructured nature—they are taken by many different people with many different cameras, from largely unknown viewpoints.



cover 3D geometry from these photos completely automatically.

We developed a technique to reconstruct accurate 3D models of world landmarks from large collections of uncalibrated images on photo-sharing Web sites.^{1,9} The principle underlying this technique is similar to that used by stereopsis, which allows humans to perceive the world in 3D. Our two eyes view a scene from slightly different perspectives, and from these two views the brain can infer the depth of each scene point based on the difference between where the point appears in the two images. The corresponding computer vision problem of inferring depth given the input from two different cameras is the well-studied stereo problem.¹²

In the case of reconstructing landmarks using Flickr images, there are not two, but thousands, of images serving as independent views. The problem is much more difficult, however, because the precise positions and viewing directions of the cameras are not known. (The latitude-longitude coordinates in geotags are much too noisy for this purpose. Even the geotags produced by GPS receivers are very noisy because consumer GPS devices have an accuracy of about 10 meters.) Hence, both the structure of the scene and the positions of all of the cameras must be inferred simultaneously. This is known in computer vision as the *structure from motion* problem. While structure from motion has been studied for decades, Internet photo collections pose new challenges to computer vision because of their extreme scale and unstructured nature—they are taken by many different people with many different cameras, from largely unknown viewpoints. Moreover, the images on a site such as Flickr contain significant noise arising from mislabeled images, poor-quality photos, image occlusions, and transitory objects (such as people) appearing in the scene.

Solving this problem means first knowing which images of a given landmark have visual overlap. As with our technique for choosing representative views, we first perform SIFT feature matching between pairs of images to build an image network. Unrelated images, such as a closeup of a pigeon, are automatically discarded, as they will not be connected to other images that

actually feature the landmark. This matching algorithm is computationally expensive but is easily parallelized.⁹

Figure 5 illustrates SIFT feature matching in more detail. Given the input photo (a) on the top left of Figure 8, SIFT extracts a number of features,

consisting of salient locations and scales in the image, as well as a high-dimensional descriptor summarizing the appearance of each feature. A subset of detected feature locations, depicted as yellow circles, is superimposed on the image (b) on the top right. The image is

shown again on the bottom (c) next to an image from a similar viewpoint; we can match SIFT features to find a correspondence between these images. Because of the robustness of SIFT, most of these matches are correct.

Once we have the network of visual connectivity between images, we need to estimate the precise position and orientation of the camera used to capture each image—that is, exactly where the photographer was standing and in which direction he or she was pointing the camera—as well as the 3D coordinates of every SIFT point matched in the images. It turns out that this can be posed as an enormous optimization problem, in which the location of each scene point and the position of each camera are estimated given constraints induced by the same scene points appearing in multiple images. This optimization tries to find the camera and scene geometry that, when related to each other through perspective projection, most closely agree with the 2D SIFT matches found between the images. This optimization problem is difficult to solve, not only because of its size, but also because the objective function is highly nonlinear.

Information in the visual network, as well as absolute location information from geotags, however, can help with this reconstruction task. Consider a pair of visually overlapping images, such as the two photographs shown in the upper left of Figure 8. Using the computed SIFT matches and geometric reasoning algorithms, we can determine the geometric relationship between these two images—that image 2 is taken to the left of image 1 and rotated slightly clockwise, say. We can compute such *relative camera poses* for each edge in the network (such as the small network on the right of Figure 8). By computing many such relationships, we can build up a network of information on top of a set of images, shown to the right. We also have geotags for some images, shown as latitude/longitude coordinates. Unfortunately, these geotags are very noisy and can at times be hundreds of meters away from a photo's true location. On the other hand, some geotags are quite accurate. If we knew which were good, we could propagate locations from those photos to their neigh-

Figure 8. Pose network—can you tell where the second photo was taken relative to the first?

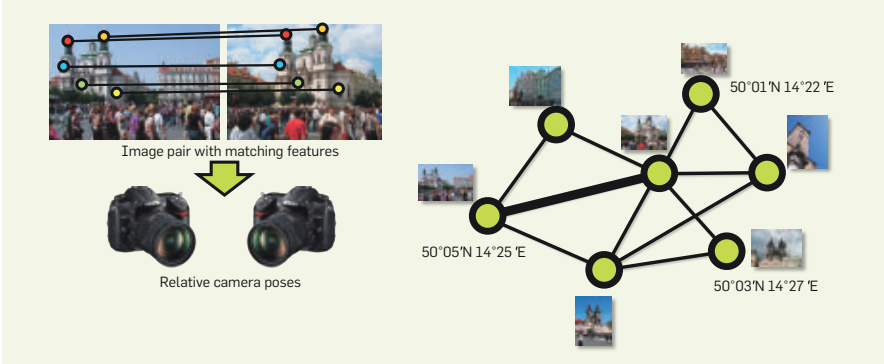


Figure 9. 3D reconstructions.



bors in the network. Given a raw set of geotagged photos, however, we do not know which are accurate.

To overcome this problem we have developed a new technique that uses the image network to combine these position estimates in a more intelligent, robust way, “averaging out” errors in the noisy observations by passing geometric information between nodes in the image network. This algorithm uses a *message-passing* strategy based on an technique known as *loopy belief propagation* commonly used in machine learning, computer vision, and other areas.¹⁸ This algorithm is scalable and can find good solutions to very nonlinear problems. While complex, our algorithm starts with the simple idea that each image should repeatedly average its location with that of its neighbors, hence using the graph to smooth noisy location estimates. Because of the extreme noise, this simple averaging approach doesn’t work well; therefore, we developed a more sophisticated approach.⁹

The message-passing process described here repeats for a number of rounds, so each image repeatedly updates its position based on information from its neighbors. This algorithm results in fairly accurate camera positions, and applying standard optimization techniques (such as gradient descent) using these positions as a starting point can yield further improvements. With this algorithm we have built some very large 3D models, including the reconstructions of the city of Dubrovnik and parts of Rome shown in Figure 9. To process these large problems, we implemented the algorithm using the MapReduce framework and ran these as jobs on a large Hadoop cluster. (For more information, see our project’s Web page¹⁰). In other work on the 3D modeling problem, we reconstructed all of the major sites in Rome from hundreds of thousands of Flickr photos in less than 24 hours (thus reconstructing “Rome in a Day”).^{3,19}

While photo-sharing sites such as Flickr and Facebook continue to grow at a breathtaking pace, they still do not have enough images to reach our eventual goal of reconstructing the entire world in 3D. The main problem is that the geospatial distribution of



Figure 10. PhotoCity.

photographs is highly nonuniform, as noted in the previous section—there are hundreds of thousands of photos of Notre Dame but virtually none of the café across the street.

One solution to this problem is to entice people to take photos of underrepresented places through *gamification*. This is the idea behind PhotoCity, an online game developed in collaboration with the University of Washington. In PhotoCity, teams of players compete against one another by taking photos at specific points in space to capture flags and buildings.²² Through this game, we collected more than 100,000 photos of the Cornell and University of Washington campuses over a period of a few weeks. We used these photos to reconstruct large portions of the two campuses, including areas that otherwise did not have much photographic coverage on sites such as Flickr. A few example building models created from these photos, along with a screenshot of the PhotoCity interface, are shown in Figure 10. On the left is a screenshot of the PhotoCity interface showing an overhead map depicting the state of the game. On the right are a few 3D models created from photos uploaded by players.

Creating a successful game involved two key challenges: building a robust online system for users to upload photos for processing; and designing the game mechanics in such a way that users were excited about playing. To address the first challenge, we built a version of our 3D reconstruction algorithm that could take a new photo of a building and quickly integrate it into our current 3D model of that building, updating that model with any new information contributed by that photo.

For the second challenge of design-

ing effective game mechanics, we developed a mix of incentives. One set of incentives involved competition at different levels (for example, between students at the same school, as well as a race for each school to build the best model). Another set involved giving each player visual feedback about how much he or she contributed to the model, by showing 3D points created by that player’s photos and by updating models so that players could see the progress of the game as a whole over time. A survey of players after the conclusion of the competition revealed that different players were motivated by different incentives; some were driven by competition, while others simply enjoyed seeing the virtual world grow over time.

Future Work

This article has presented some of our initial work into unlocking the information latent in large photo-sharing Web sites using network-analysis algorithms, but the true promise of this type of analysis is yet to be realized. The opportunities for future work in this area lie along two different lines. First, new algorithms are needed to extract visual content more efficiently and accurately: the algorithms presented here produce incorrect results on some specific types of scenes, for example, and they are relatively compute-intensive, requiring many hours on large clusters of computers to process just a few thousand images.

Second, this type of analysis could be applied to other disciplines. Many scientists are interested in studying the world and how it has changed over time, including archaeologists, architects, art historians, ecologists, urban planners, and so on. As a specific example, the 3D reconstruction

technique could simplify mapping remote archaeological sites,⁴ where using traditional laser range scanners is expensive and challenging. A cheaper and simpler alternative would be to use a digital camera to take many photos of a site, and then run our reconstruction algorithms on those photos once the researchers return from the field. As another example, we have recently studied how to automatically mine online photo collections for images of natural phenomena like snowfall and flowering, potentially giving ecologists a new technique for collecting observational data at a continental scale.²³

Imagine all of the world's photos as coming from a “distributed camera,” continually capturing images all around the world. Can this camera be calibrated to estimate the place and time each of these photos was taken? If so, we could start building a new kind of image search and analysis tool—one that would, for example, allow a scientist to find all images of Central Park over time in order to study changes in flowering times from year to year, or that would allow an engineer to find all available photos of a particular bridge online to determine why it collapsed. Gaining true understanding of the world from the sea of photos online could have a truly transformative impact.

An earlier version of this article was presented as a keynote talk at Arts | Humanities | Complex Networks—A Leonardo satellite symposium at NetSci2010 (<http://artshumanities.netsci2010.net>). 



Related articles on queue.acm.org

Photoshop Scalability: Keeping It Simple

Clem Cole, Russell Williams

<http://queue.acm.org/detail.cfm?id=1858330>

Unifying Biological Image Formats with HDF5

Matthew T. Dougherty et al.

<http://queue.acm.org/detail.cfm?id=1628215>

Document & Media Exploitation

Simson L. Garfinkel

<http://queue.acm.org/detail.cfm?id=1331294>

References

1. Agarwal, S., Snavely, N., Simon, I., Seitz, S., and Szeliski, R. *Building Rome in a day*. *International Conference on Computer Vision* (2009).
2. The App Garden. 2010; <http://www.flickr.com/services/api/>.
3. Building Rome in a day; <http://grail.cs.washington.edu/>

This article has presented some of our initial work into unlocking the information latent in large photo-sharing Web sites using network-analysis algorithms, but the true promise of this type of analysis is yet to be realized.

4. Chen, X., Morvan, Y., He, Y., Dorsey, J. and Rushmeier, H. An integrated image and sketching environment for archaeological sites. *Workshop on Applications of Computer Vision in Archaeology*, (2010).
5. Comaniciu, D. and Meer, P. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24, 5 (2002).
6. Crandall, D., Backstrom, L., Cosley, D., Suri, S., Huttenlocher, D. and Kleinberg, J. Inferring social ties from geographic coincidences. *Proceedings of the National Academy of Science* 107, 52 (2010), 22436–22441.
7. Crandall, D., Backstrom, L., Huttenlocher, D. and Kleinberg, J. Mapping the world's photos. *International World Wide Web Conference* (2009).
8. Crandall, D., Backstrom, L., Huttenlocher, D., and Kleinberg, J. Mapping the world's photos; <http://www.cs.cornell.edu/~crandall/photomap/>.
9. Crandall, D., Owens, A., Snavely, N. and Huttenlocher, D. Discrete-continuous optimization for large-scale structure from motion. *Conference on Computer Vision and Pattern Recognition* (2011), 3001–3008.
10. Crandall, D., Owens, A., Snavely, N., Huttenlocher, D. Discrete-continuous optimization for large-scale structure from motion, (2011); <http://vision.soi.cornell.edu/disco/>.
11. Flickr. 6,000,000,000; <http://blog.flickr.net/en/2011/08/04/6000000000/>.
12. Hartley, R. and Zisserman, A. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, MA, 2003.
13. Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabasi, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D. and Van Alstyne, M. Computational social science. *Science* 323, 5915 (2009), 721–723.
14. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *International Journal in Computer Vision* (2004).
15. Melki, S. Photo reproduced under a Creative-Commons-Attribution license. Flickr user sergemelki; <http://www.flickr.com/photos/sergemelki/3391168464/in/photostream/>.
16. Michel, J.-B., Shen, Y.K., Aiden, A.P., Veres, A. and Gray, M.K., The Google Books Team, Pickett, J.P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M., Lieberman-Aiden, E. Quantitative analysis of culture using millions of digitized books. *Science* 331, 6014 (2011), 176–182.
17. Milgram, S. Psychological maps of Paris. In *Environmental Psychology: People and Their Physical Settings*. H.M. Proshansky, W.H. Ittelson, and L.G. Rivlin, eds. Holt, Rinehart and Winston, New York, NY, 1976, 104–124.
18. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.
19. Photo Tourism; <http://phototour.cs.washington.edu/>.
20. Shaffer, J. Bigger, faster photos. The Facebook Blog (2011); <http://blog.facebook.com/blog.php?post=10150262684247131>.
21. Snavely, N., Seitz, S. and Szeliski, R. Modeling the world from Internet photo collections. *International Journal of Computer Vision* 80, 2 (2008).
22. Tuite, K., Snavely, N., Hsiao, D.-Y., Tabing, N., and Popović, Z. PhotoCity: Training experts at large-scale image acquisition through a competitive game. *Conference on Human Factors in Computing* (2011).
23. Zhang, H., Korayem, M., Crandall, D., and LeBuhn, G. Mining Photo-sharing Websites to Study Ecological Phenomena. *International World Wide Web Conference* (2012).

David Crandall is an assistant professor in the School of Informatics and Computing at Indiana University in Bloomington, IN. His research interests are computer vision and data mining, with a focus on visual object recognition, image understanding, machine learning, and mining and modeling of complex networks.

Noah Snavely is an assistant professor of computer science at Cornell University, where he has been on the faculty since 2009. Snavely works in computer graphics and computer vision, with a particular interest in using vast amounts of imagery from the Internet to reconstruct and visualize the world in 3D, and in creating new tools for enabling people to capture and share their environments.

© 2012 ACM 0001-0782/12/06 \$10.00



Mobile computer-vision technology will soon become as ubiquitous as touch interfaces.

**BY KARI PULLI, ANATOLY BAKSHEEV,
KIRILL KORNYAKOV, AND VICTOR ERUHIMOV**

Real-Time Computer Vision with OpenCV

COMPUTER VISION IS a rapidly growing field devoted to analyzing, modifying, and high-level understanding of images. Its objective is to determine what is happening in front of a camera and use that understanding to control a computer or robotic system, or to provide people with new images that are more informative

or aesthetically pleasing than the original camera images. Application areas for computer-vision technology include video surveillance, biometrics, automotive, photography, movie production, Web search, medicine, augmented reality gaming, new user interfaces, and many more.

Modern cameras are able automatically to focus on people's faces and trigger the shutter when they smile. Optical text-recognition systems help transform scanned documents into text that can be analyzed or read aloud by a voice synthesizer. Cars may include automated driver-assistance systems that help

users park or warn them about potentially dangerous situations. Intelligent video surveillance plays an increasingly important role in monitoring the security of public areas.

As mobile devices such as smartphones and tablets come equipped with cameras and more computing power, the demand for computer-vision applications is increasing. These devices have become smart enough to merge several photographs into a high-resolution panorama, or to read a QR code, recognize it, and retrieve information about a product from the Internet. It will not be long before mobile computer-

vision technology becomes as ubiquitous as touch interfaces.

Computer vision is computationally expensive, however. Even an algorithm dedicated to solving a very specific problem, such as panorama stitching or face and smile detection, requires a lot of power. Many computer-vision scenarios must be executed in real time, which implies that the processing of a single frame should complete within 30–40 milliseconds. This is a very challenging requirement, especially for mobile and embedded computing architectures. Often, it is possible to trade off quality for speed. For example, the panorama-stitching algorithm can find more matches in source images and synthesize an image of higher quality, given more computation time. To meet the constraints of time and the computational budget, developers either compromise on quality or invest more time into optimizing the code for specific hardware architectures.

Vision And Heterogeneous Parallel Computing

In the past, an easy way to increase the performance of a computing device was to wait for the semiconductor processes to improve, which resulted in an increase in the device's clock speed. When the speed increased, all applications got faster without having to modify them or the libraries that they relied on. Unfortunately, those days are over.

As transistors get denser, they also leak more current, and hence are less energy efficient. Improving energy efficiency has become an important priority. The process improvements now allow for more transistors per area, and there are two primary ways to put them to good use. The first is via parallelization: creating more identical processing units instead of making the single unit faster and more powerful. The second is via specialization: building domain-specific hardware accelerators that can perform a particular class of functions more efficiently. The concept

of combining these two ideas—that is, running a CPU or CPUs together with various accelerators—is called heterogeneous parallel computing.

High-level computer-vision tasks often contain subtasks that can be run faster on special-purpose hardware architectures than on the CPU, while other subtasks are computed on the CPU. The GPU (graphics processing unit), for example, is an accelerator that is now available on every desktop computer, as well as on mobile devices such as smartphones and tablets.

The first GPUs were fixed-function pipelines specialized for accelerated drawing of shapes on a computer display, as illustrated in Figure 1. As GPUs gained the capability of using color images as input for texture mapping, and their results could be shared back with the CPU rather than just being sent to the display, it became possible to use GPUs for simple image-processing tasks.

Making the fixed-function GPUs partially programmable by adding shaders was a big step forward. This enabled programmers to write special programs that were run by the GPU on every three-dimensional point of the surface and at every pixel rendered onto the output canvas. This vastly expanded the GPU's processing capability, and clever programmers began to try general-purpose computing on a GPU (GPGPU), harnessing the graphics accelerator for tasks for which it was not originally designed. The GPU became a useful tool for image processing and computer-vision tasks.

The graphics shaders, however, did not provide access to many useful hardware capabilities such as synchronization and atomic memory operations. Modern GPU computation languages such as CUDA, OpenCL, and DirectCompute are explicitly designed to support general-purpose computing on graphics hardware. GPUs are still not quite as flexible as CPUs, but they perform parallel stream processing much more efficiently, and an increasing number of nongraphics applications are being rewritten using the GPU compute languages.

Computer vision is one of the tasks that often naturally map to GPUs. This is not a coincidence, as computer vision really solves the inverse to the

Figure 1. Computer vision and GPU.

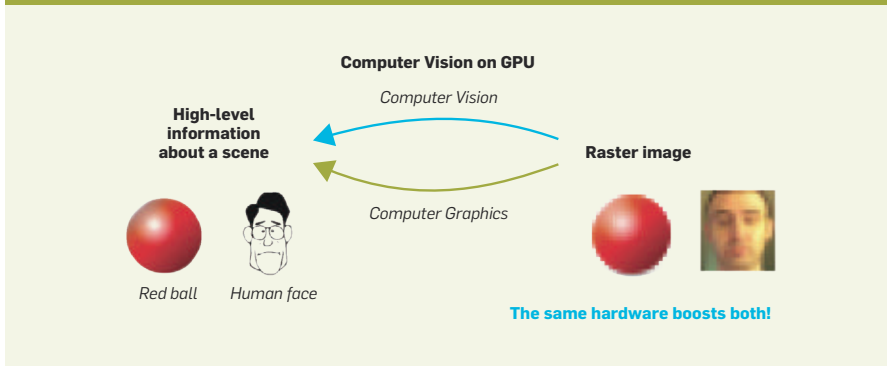
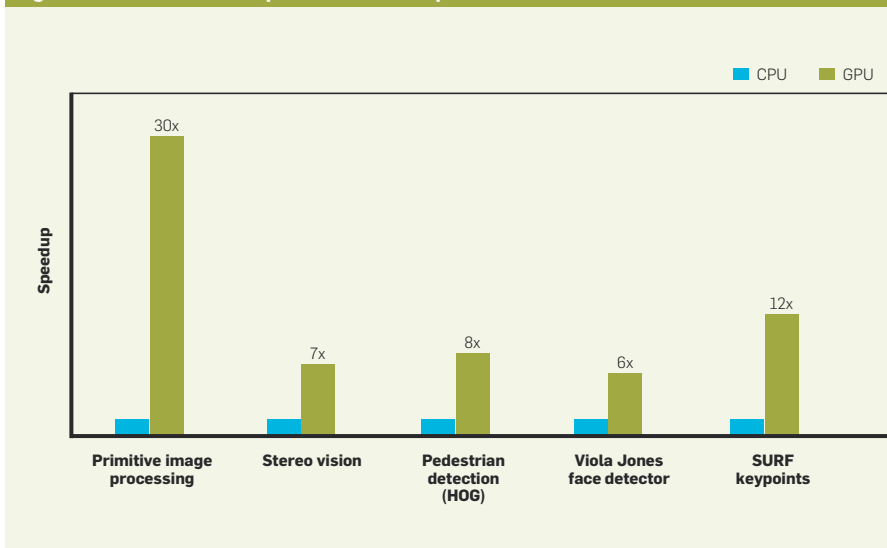


Figure 2. CPU versus GPU performance comparison.



computer graphics problem. While graphics transforms a scene or object description to pixels, vision transforms pixels to higher-level information. GPUs contain lots of similar processing units and are very efficient in executing simple, similar subtasks such as rendering or filtering pixels. Such tasks are often known as “embarrassingly parallel,” because they are so easy to parallelize efficiently on a GPU.

Many tasks, however, do not parallelize easily, as they contain serial segments where the results of the later stages depend on the results of earlier stages. These serial algorithms do not run efficiently on GPUs and are much easier to program and often run faster on CPUs. Many iterative numerical optimization algorithms and stack-based tree-search algorithms belong to that class.

Since many high-level tasks consist of both parallel and serial subtasks, the entire task can be accelerated by running some of its components on the CPU and others on the GPU. Unfortunately, this introduces two sources of inefficiency. One is synchronization: when one subtask depends on the results of another, the later stage needs to wait until the previous stage is done. The other inefficiency is the overhead of moving the data back and forth between the GPU and CPU memories—and since computer-vision tasks need to process lots of pixels, it can mean moving massive data chunks back and forth. These are the key challenges in accelerating computer-vision tasks on a system with both a CPU and GPU.

OpenCV Library

The open source computer vision library, OpenCV, began as a research project at Intel in 1998.⁵ It has been available since 2000 under the BSD open source license. OpenCV is aimed at providing the tools needed to solve computer-vision problems. It contains a mix of low-level image-processing functions and high-level algorithms such as face detection, pedestrian detection, feature matching, and tracking. The library has been downloaded more than three million times.

In 2010 a new module that provides GPU acceleration was added to OpenCV. The GPU module covers a

significant part of the library’s functionality and is still in active development. It is implemented using CUDA and therefore benefits from the CUDA ecosystem, including libraries such as NVIDIA Performance Primitives (NPP).

The GPU module allows users to benefit from GPU acceleration without requiring training in GPU programming. The module is consistent with the CPU version of OpenCV, which makes adoption easy. There are differences, however, the most important of which is the memory model. OpenCV implements a container for images called `cv::Mat` that exposes access to image raw data. In the GPU module the container `cv::gpu::GpuMat` stores the image data in the GPU memory and does not provide direct access to the data. If users want to modify the pixel data in the main program running on the GPU, they first need to copy the data from `GpuMat` to `Mat`.

```
#include <opencv2/opencv.hpp>
#include <opencv2/gpu/gpu.hpp>
using namespace cv;
...
Mat image = imread("file.png");
gpu::GpuMat image_gpu;
image_gpu.upload(image);
gpu::GpuMat result;
gpu::threshold(image_gpu,
    result, 128, CV_THRESH_BINARY);
result.download(image);
imshow("WindowName", image);
waitKey ();
```

In this example, an image is read from a file and then uploaded to GPU memory. The image is thresholded there, and the result is downloaded to CPU memory and displayed. In this simple example only one operation is performed on the image, but several others could be executed on the GPU without transferring images back and forth. The usage of the GPU module is straightforward for someone who is already familiar with OpenCV.

This design provides the user with explicit control over how data is moved between CPU and GPU memory. Although the user must write some additional code to start using the GPU, this approach is flexible and allows more efficient computations. In general, it is a good idea to research, develop, and debug a computer-vision application us-

ing the CPU part of OpenCV, and then accelerate it with the GPU module. Developers should try different combinations of CPU and GPU processing, measure their timing, and then choose the combination that performs the best.

Another piece of advice for developers is to use the asynchronous mechanisms provided by CUDA and the GPU module. This allows simultaneous execution of data transfer, GPU processing, and CPU computations. For example, while one frame from the camera is processed by the GPU, the next frame is uploaded to it, minimizing data-transfer overheads and increasing overall performance.

Performance of OpenCV GPU Module

OpenCV’s GPU module includes a large number of functions, and many of them have been implemented in different versions, such as the image types (char, short, float), number of channels, and border extrapolation modes. This makes it challenging to report exact performance numbers. An added source of difficulty in distilling the performance numbers down is the overhead of synchronizing and transferring data. This means that best performance is obtained for large images where a lot of processing can be done while the data resides on the GPU.

To help the developer figure out the trade-offs, OpenCV includes a performance benchmarking suite that runs GPU functions with different parameters and on different datasets. This provides a detailed benchmark of how much different datasets are accelerated on the user’s hardware.

Figure 2 is a benchmark demonstrating the advantage of the GPU module. The speedup is measured against the baseline of a heavily optimized CPU implementation of OpenCV. OpenCV was compiled with Intel’s Streaming SIMD Extensions (SSE) and Threading Building Blocks (TBB) for multicore support, but not all algorithms use them. The primitive image-processing speedups have been averaged across roughly 30 functions. Speedups are also reported for several high-level algorithms.

It is quite normal for a GPU to show a speedup of 30 times for low-level functions and up to 10 times for high-level

functions, which include more overhead and many steps that are not easy to parallelize with a GPU. For example, the granularity for color conversion is per-pixel, making it easy to parallelize. Pedestrian detection, on the other hand, is performed in parallel for each possible pedestrian location, and parallelizing the processing of each window position is limited by the amount of on-chip GPU memory.

As an example, we accelerated two packages from Robot Operation System (ROS)⁸—stereo visual odometry and textured object detection—that were originally developed for the CPU.

They contain many functional blocks and a class hierarchy.

Wherever it made sense, we offloaded the computations to the GPU. For example, OpenCV GPU implementations performed Speeded-Up Robust Feature (SURF) key point detection, matching, and search of stereo correspondences (block matching) for stereo visual odometry. The accelerated packages were a mix of CPU/GPU implementations. As a result, the visual odometry pipeline was accelerated 2.7 times, and textured object detection was accelerated from 1.5–4 times, as illustrated in Figure 3. Data-transfer

overhead was not a significant part of the total algorithm time. This example shows that replacing only a few lines of code results in a considerable speedup of a high-level vision application.

Stereo Correspondence with GPU Module

Stereo correspondence search in a high-resolution video is a demanding application that demonstrates how CPU and GPU computations can be overlapped. OpenCV's GPU module includes an implementation that can process full HD resolution stereo pair in real time (24 frames per second) on the NVIDIA GTX580.

In a stereo system, two cameras are mounted facing in the same direction. While faraway objects project to the same image locations on each camera, nearby objects project to different locations. This is called disparity. By locating each pixel on the left camera image where the same surface point projects to the right image, you can compute the distance to that surface point from the disparity. Finding these correspondences between pixels in the stereo image pairs is the key challenge in stereo vision.

This task is made easier by rectifying the images. Rectification warps the images to an ideal stereo pair where each scene surface point projects to a matching image row. This way, only points on the same scan line need to be searched. The quality of the match is evaluated by comparing the similarity of a small window of pixels with the candidate-matching pixel. Then the pixel in the right image whose window best matches the window of the pixel on the left image is selected as the corresponding match.

The computational requirements obviously increase as the image size increases, because there are more pixels to process. In a larger image the range of disparities measured in pixels also increases, which requires a larger search radius. For small-resolution images the CPU may be sufficient to calculate the disparities; with full HD resolution images, however, only the GPU can provide enough processing power.

Figure 4 presents a block-matching pipeline that produces a disparity image $d(x,y)$ such that $\text{LeftImage}(x,y)$ corresponds to $\text{RightImage}(x-d(x,y),y)$. The pipeline first rectifies the images

Figure 3. Textured object detection application: CPU and GPU.

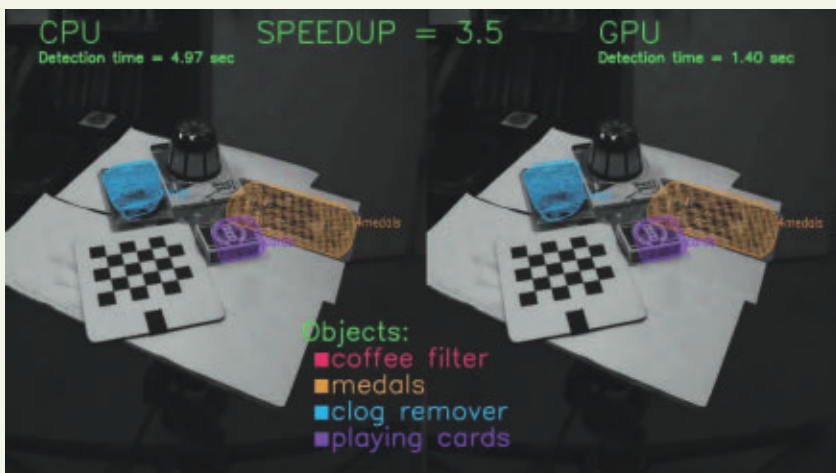


Figure 4. Stereo block matching pipeline.

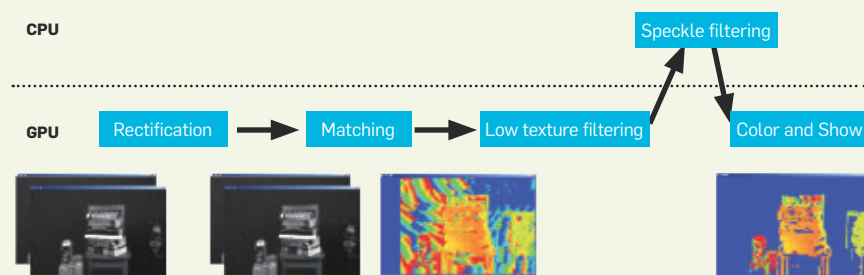
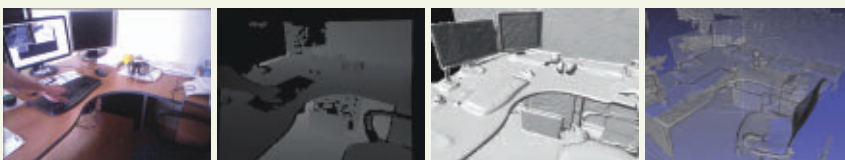


Figure 5. RGB frame, depth frame, ray-casted frame, and point cloud.



and then finds the best matches, as previously described. In areas where there is little texture—for example, a blank wall—the calculated matches are unreliable, so all such areas are marked to be ignored in later processing. As the disparity values are expected to change significantly near object borders, the speckle-filtering stage eliminates speckle noise within large continuous regions of disparity image. Unfortunately, the speckle-filtering algorithm requires a stack-based depth-first search difficult to parallelize, so it is run on the CPU. The results are visualized using a false-color image.

All the steps except speckle filtering are implemented on the GPU. The most compute-intensive step is block matching. NVIDIA GTX580 has accelerated it seven times faster than a CPU implementation on a quad core Intel i5-760 2.8GHz processor with SSE and TBB optimizations. After this speedup the speckle filtering becomes the bottleneck, consuming 50% of the frame-processing time.

An elegant parallel-processing solution is to run speckle filtering on the CPU in parallel with the GPU processing. While the GPU processes the next frame, the CPU performs speckle filtering for the current frame. This can be done using asynchronous OpenCV GPU and CUDA capabilities. The heterogeneous CPU/GPU system now provides a sevenfold speedup for the high-resolution stereo correspondence problem, allowing real-time (24fps) performance at full HD resolution.

KinectFusion

Microsoft's KinectFusion⁴ is an example of an application that previously required slow batch processing but now, when powered by GPUs, can be run at interactive speeds. Kinect is a camera that produces color and depth images. Just by aiming the Kinect device around, one can digitize the 3D geometry of indoor scenes at an amazing fidelity, as illustrated in Figure 5. An open source implementation of such a scanning application is based on the Point Cloud Library,⁶ a companion library to OpenCV that uses 3D points and voxels instead of 2D pixels as basic primitives.

Implementing KinectFusion is not a simple task. Kinect does not return

range measurements for all the pixels, and it works reliably only on continuous smooth matte surfaces. The range measurements that it returns are noisy, and depending on the surface shapes and reflectance properties, the noise can be significant. The noise also increases with the distance to the measured surface. Kinect generates a new depth frame 30 times in a second. If the user moves the Kinect device too fast, the algorithm gets confused and cannot track the motion using the range data. With a clever combination of good algorithms and using the processing power provided by GPUs, however, KinectFusion works robustly.

There are three key concepts that make a robust interactive implementation feasible. First, the tracking algorithm is able to process the new scan data so fast that the camera has time to move very little between the consecutive frames. This makes it feasible to track the camera position and orientation using just the range data.

Second, fusion of depth data is done using a volumetric surface representation. The representation is a large voxel grid that makes it easier to merge the data from different scans in comparison with surface-based representations. To obtain high model quality, the grid resolution is chosen to be as dense as possible ($512 \times 512 \times 512$), so it has to be processed by the GPU for real-time rates.

Finally, the manner in which the new data is merged with the old reduces the noise and uncertainty as more data is gathered, and the accuracy of the model keeps improving. As the model gets better, tracking gets easier as well. Parallel ray casting through the volume is done on the GPU to get depth information, which is used for camera tracking on the next frame. So frame-to-frame movement estimation is performed only between the first and second frames. All other movements are computed on model-to-frame data, which makes camera tracking very robust.

All of these steps are computationally intensive. Volumetric integration requires the high memory bandwidth that only the GPU can deliver at a price low enough to be affordable by normal consumers. Without GPUs this system would simply not be feasible. However, not every step of the computation is easy to do on a GPU. For example,

tracking the camera position is done on a CPU. Though the linear equation matrix required for camera position estimation is fully computed on the GPU, computing the final solution does not parallelize well, so it is done on the CPU, which results in some download and API call overhead. Another problem is that the bottom-level image in the hierarchical image processing approach is only 160×120 , which is not large enough to fully load a GPU. All the other parts are ideal for GPU but limited by the amount of available GPU memory and computing resources.

Further development requires even more GPU power. At the moment, the size of the scene is limited by the volumetric representation. Using the same number of voxels but making them bigger would allow us to capture a larger scene but at a coarser resolution. Retaining the same resolution while scanning larger scenes would require more voxels, but the number of voxels is limited by the amount of memory available on GPU and by its computational power.

Mobile Devices

While PCs are often built with a CPU and a GPU on separate chips, mobile devices such as smartphones and tablets put all the computing elements on a single chip. Such an SoC (system on chip) contains one or more CPUs, a GPU, as well as several signal processors for audio and video processing and data communication. All modern smartphones and some tablets also contain one or more cameras, and OpenCV is available on both Android and iOS operating systems. With all these components, it is possible to create mobile vision applications. The following sections look at the mobile hardware in more detail, using NVIDIA's Tegra 2 and Tegra 3 SoCs as examples, and then introduce several useful multimedia APIs. Finally, two mobile vision applications are presented: panorama creation and video stabilization.

Tools for Mobile Computer Vision

At the core of any general-purpose computer is the CPU. While Intel's x86 instruction set rules on desktop computers, almost all mobile phones and tablets are powered by CPUs from ARM. ARM processors follow the RISC (reduced instruction set computing)

Energy savings with GLSL on Tegra 3.

OpenCV Function (10,000 iterations)	Energy Savings
median blur	3.43
planar warper	6.25
warpPerspective	6.45
cylindrical warper	3.89
blur3x3	3.60
warpAffine	15.38

approach, as can be deduced from ARM's original name, Advanced Risc Machines. While x86 processors were traditionally designed for high computing power, ARM processors were designed primarily for low-power usage, which is a clear benefit for battery-powered devices. As Intel is reducing power usage in its Atom family for mobile devices, and recent ARM designs are getting increasingly powerful, they may in the future reach a similar design point, at least on the high end of mobile computing devices. Both Tegra 2 and Tegra 3 use ARM Cortex-A9 CPUs.

Mobile phones used to have only one CPU, but modern mobile SoCs are beginning to sport several, providing symmetric multiprocessing. The reason is the potential for energy savings. One can reach roughly a similar level of performance using two cores running at 1GHz each than with one core running at 2GHz. Since the power consumption increases super-linearly with the clock speed, however, these two slower cores together consume less power than the single faster core. Tegra 2 provides two ARM cores, while Tegra 3 provides four. Tegra 3 actually contains five (four plus one) cores, out of which one, two, three, or four cores can be active at the same time. One of the cores, known as the shadow or companion core, is designed to use particularly little energy but can run only at relatively slow speeds. That mode is sufficient for standby, listening to music, voice calls, and other applications that rely on dedicated hardware such as the audio codec and require only a few CPU cycles. When more processing power is needed (for example, reading email), the slower core is replaced by one of the faster cores, and for increased performance (browsing, gaming) additional cores kick in.

SIMD (single instruction, multiple data) processing is particularly useful for pixel data, as the same instruction can be used on multiple pixels simultaneously. SSE is Intel's SIMD technology, which exists on all modern x86 chips. ARM has a similar technology called NEON, which is an optional coprocessor in the Cortex A9. The NEON can process up to eight, and sometimes even 16 pixels at the same time, while the CPU can process only one element at a time. This is very attractive for computer-vision developers, as it is often easy to obtain three to four times performance speedup—and with careful optimization even more than six times. Tegra 2 did not include the NEON extension, but each of Tegra 3's ARM cores has a NEON coprocessor.

All modern smart phones include a GPU. The first generation of mobile GPUs implemented the fixed-functionality graphics pipeline of OpenGL ES 1.0 and 1.1. Even though the GPUs were designed for 3D graphics, they could be used for a limited class of image-processing operations such as warping and blending. The current mobile GPUs are much more flexible and support OpenGL shading language (GLSL) programming with the OpenGL ES 2.0 API, allowing programmers to run fairly complicated shaders at each pixel. Thus, many old-school GPGPU tricks developed for desktop GPUs about 10 years ago can now be reused on mobile devices. The more flexible GPU computing languages such as CUDA and OpenCL will replace those tricks in the coming years but are not available yet.

Consumption and creation of audio and video content is an important use case on modern mobile devices. To support them, smartphones contain dedicated hardware encoders and decoders both for audio and video. Additionally, many devices have a special ISP (image signal processor) that processes the pixels streaming out from the camera. These media accelerators are not as easily accessible and useful for computer-vision processing, but the OpenMAX standard helps.¹ OpenMAX defines three different layers: AL (application), IL (integration), and DL (development). The lowest, DL, specifies a set of primitive functions from five domains: audio/video/image coding and image/

signal processing. Some of them are of potential interest for computer-vision developers, especially video coding and image processing, because they provide a number of simple filters, color space conversions, and arithmetic operations. IL is meant for system programmers for implementing the multimedia framework and provides tools such as for camera control. AL is meant for application developers and provides high-level abstractions and objects such as Camera, Media Player, and Media Recorder. The OpenMAX APIs are useful for passing image data efficiently between the various accelerators and other APIs such as OpenGL ES.

Sensors provide another interesting opportunity for computer-vision developers. Many devices contain sensors such as an accelerometers, gyroscopes, compasses, and GPSs. They are not able to perform calculations but can be useful if the application needs to reconstruct the camera orientation or 3D trajectory. The problem of extracting the camera motion from a set of frames is challenging, both in terms of performance and accuracy. Simultaneous localization and mapping (SLAM), structure from motion (SfM), and other approaches can compute both the camera position and even the shapes of the objects the camera sees, but these methods are not easy to implement, calibrate, and optimize, and they require a lot of processing power. The sensors can nonetheless deliver a fairly accurate estimate of the device orientation at a fraction of the cost of relying only on visual processing. For accurate results the sensor input should be used only as a starting point, to be refined using computer-vision techniques.

OpenCV On Tegra

A major design and implementation goal for OpenCV has always been high performance. Porting both OpenCV and applications to mobile devices requires care, however, to retain a sufficient level of performance. OpenCV has been available on Android since the Google Summer of Code 2010 when it was first built and run on Google Nexus One. Several demo applications illustrated almost real-time behavior, but it was obvious that OpenCV needed optimization and fine-tuning for mobile hardware.

That is why NVIDIA and Itseez decided to create a Tegra-optimized version of OpenCV. This work benefited from three major optimization opportunities: code vectorization with NEON, multithreading with the Intel TBB (Threading Building Blocks) library, and GPGPU with GLSL.

Taking advantage of the NEON instruction set was the most attractive of the three choices. Figure 6 compares the performance of original and NEON-optimized versions of OpenCV. In general, NEON requires basic arithmetic operations using simple and regular memory-access patterns. Those requirements are often satisfied by image-processing primitives, which are almost ideal for acceleration by NEON vector operations. As those primitives are often in the critical path of high-level computer vision workflows, NEON instructions can significantly accelerate OpenCV routines.

Multithreading on up to four symmetric CPUs can help at a higher level. TBB and other threading technologies enable application developers to get the parallel-processing advantage of multiple CPU cores. At the application level independent activities can be distributed among different cores, and the operating system will take care of load balancing. This approach is consistent with the general OpenCV strategy for multithreading—to parallelize the whole algorithmic pipeline—while on a mobile platform we often also have to speed up primitive functions.

One approach is to split low-level functions into several smaller sub-tasks, which produces faster results. A popular technique is to split an input image into several horizontal stripes and process them simultaneously. An alternative approach is to create a background thread and get the result later while the main program works on other parts of the problem. For example, in the video stabilization application a special class returns an asynchronously calculated result from the previous iteration. Multithreading limits the speedup factor by the number of cores, which on the most advanced current mobile platforms is four, while NEON supports vector operations on 16 elements. Of course, both of these technologies can be com-

bined. If the algorithm is constrained by the speed of memory access, however, multithreading may not provide the expected performance improvement. For example, the NEON version of `cv::resize` does not gain from adding new threads, because a single thread already fully consumes the memory-bus capacity.

The final method applied during the optimization of the OpenCV library for the Tegra platform is GPGPU with GLSL shaders. Though the mobile GPU has limitations as discussed previously, on certain classes of algorithms the GPU is able to show an impressive performance boost while consuming very little energy. On mobile SoCs it is possible to share memory between CPU and GPU, which allows interleaving C++ and GLSL processing of the same image buffer.

Figure 7 shows example speedups of some filters and geometric transformations from the OpenCV library.

An additional benefit of using the GPU is that at full speed it runs at a lower average power than the CPU. On mobile devices this is especially important, since one of the main usability factors for consumers is how long the battery lasts on a charge. We measured the average power and time elapsed to perform 10,000 iterations of some optimized C++ functions, compared with the same functions written in GLSL. Since these functions are both faster on the GPU, and the GPU runs at lower peak power. We measured the result is significant energy savings (see the accompanying table). We measured energy savings of 3–15 times when porting these functions to GPU.

Figure 6. Performance improvement with NEON on Tegra 3.

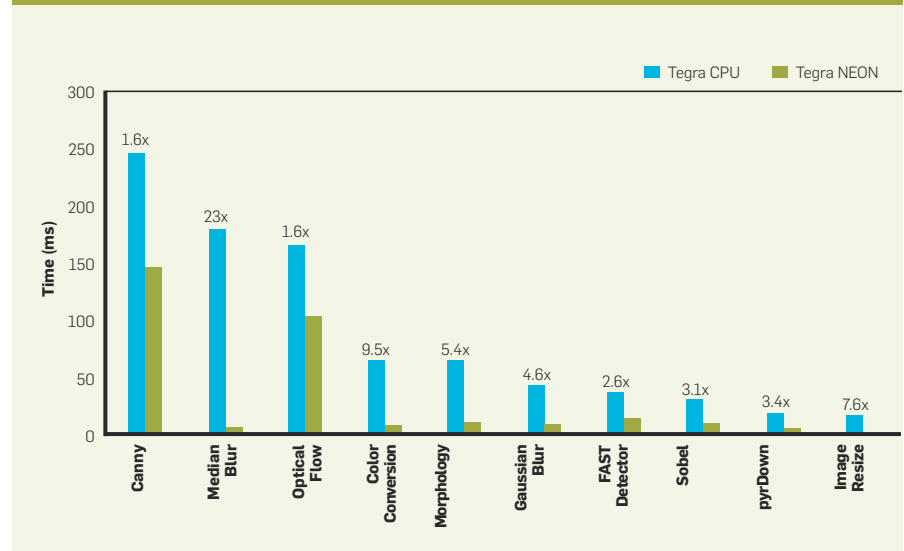
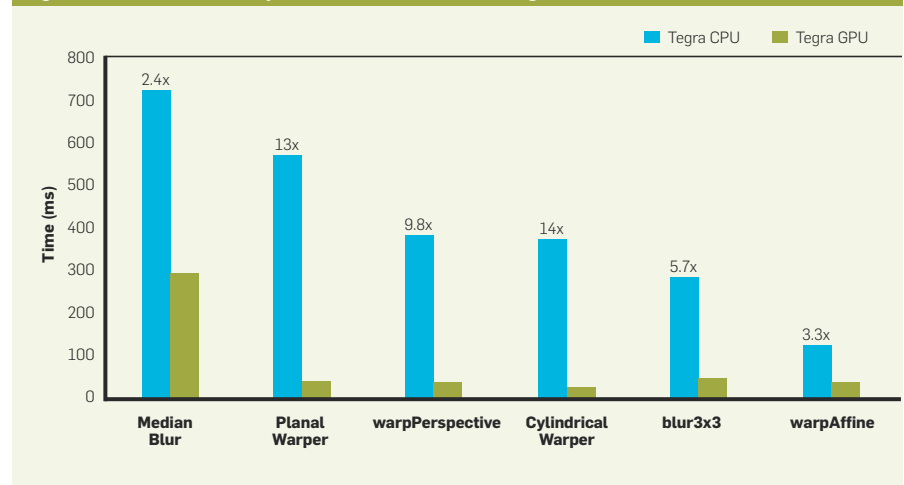


Figure 7. Performance improvement with GLSL on Tegra 3.



Applications

We have developed two mobile vision applications using OpenCV: one that stitches a panoramic image from several normal photographs, and another

that stabilizes streaming video. The performance requirements are challenging. Our goal is real-time performance, where each frame should be processed within about 30 milliseconds, of which

basic operations such as simply copying a 1280×720 -pixel frame may take eight milliseconds. Consequently, to a large extent the final design of an application and its underlying algorithm is determined by this constraint.

In both cases we were able to satisfy the time limits by using the GPU for optimizing the applications' bottlenecks. Several geometric transformation functions such as image resizing and various types of image warping were ported to the GPU, resulting in a doubling of the application performance. The results were not nearly as good when performing the same tasks using NEON and multithreading. One of the reasons was that both applications deal with high-resolution four-channel images. As a result, the memory bus was overloaded and the CPU cores competed for the cache memory. At the same time we needed to program bilinear interpolation manually, which is implemented in GPU hardware. We learned that the CPU does not work as well for full-frame geometric transformations, and the help of the GPU was invaluable. Let's consider both applications in more detail.

Panorama stitching. In the panorama-stitching application our goal was to combine several ordinary images into a single panorama with a much larger field of view (FOV) than the input images.⁷ Figure 8 demonstrates the stitching of several detailed shots into a single high-resolution image of the whole image.

Figure 9 shows the processing pipeline for the OpenCV panorama-stitching application. The process of porting to Tegra started from some algorithmic improvements, followed by NEON and multithreading optimization; yet after all these efforts, the application still was not responsive enough and could not stitch and preview the resulting panorama at interactive speeds. Among the top bottlenecks were image resizing and warping. The former is required because different algorithmic steps are performed at different resolutions, and each input frame is resized about three times, depending on the algorithmic parameters. The type of warping needed depends on the desired projection mode (spherical, cylindrical, among others) and is performed before the final panorama blending.

Figure 8. Input images and the resulting panorama.

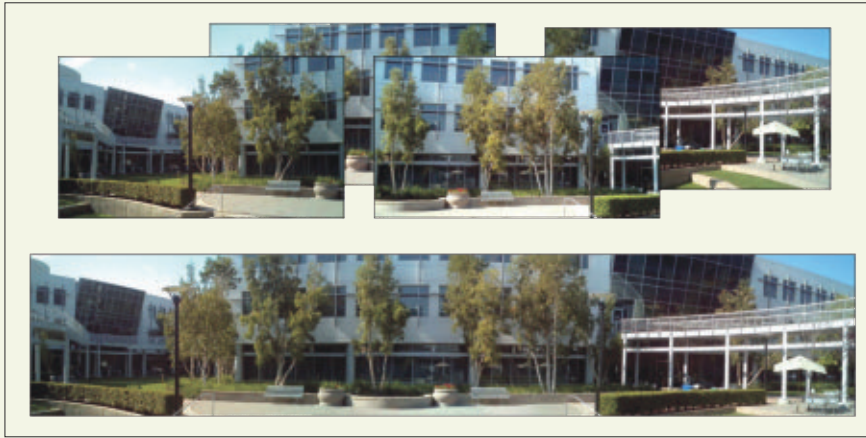


Figure 9. Panorama stitching pipeline.

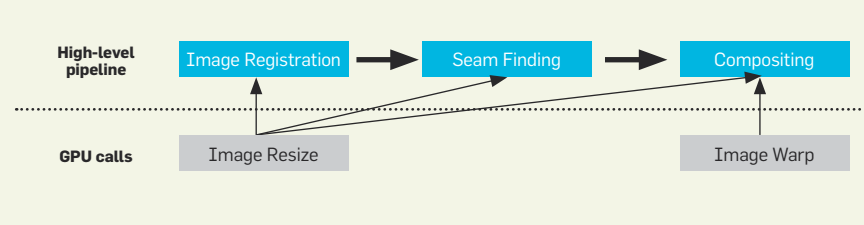


Figure 10. Video stabilization input sequence.

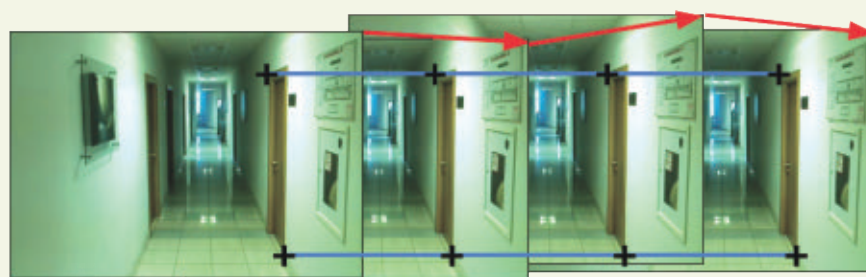
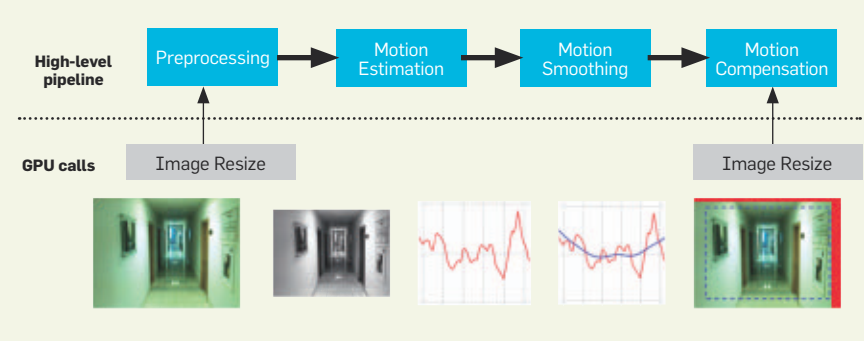


Figure 11. Video stabilization pipeline.



With the GPU version of `cv::resize` we were able to decrease scaling time from 41 milliseconds to 26 milliseconds for each input frame, which is equal to 1.6 times local speedup. Because of the GPU implementation of image warping, we could achieve even better local improvements—a boost of 8–14 times in performance, depending on the projection type. As a result, total application speedup was 1.5–2.0 times, meeting performance requirements.

Video stabilization. One of the negative consequences of recording video without a tripod is camera shake, which significantly degrades the viewing experience. To achieve visually pleasant results, all movements should be smooth, and the high-frequency variations in camera orientation and translation must be filtered. Numerous approaches have been developed, some have become open source or commercially available tools. There exist computationally intensive approaches offline that take a considerable amount of time, while the lightweight online algorithms are more suitable for mobile devices. High-end approaches often reconstruct the 3D movement of the camera and apply sophisticated nonrigid image warping to stabilize the video.² On mobile devices more lightweight approaches using translation, affine warping, or planar perspective transformations may make more sense.³

We experimented with translation and affine models, and in both cases the GPU was able to eliminate the major hotspot, which was the application of the compensating transformation to an input frame. Applying translation to compensate for the motion simply means shifting the input frame along the x and y axes and cutting off some of the boundary areas for which some of the frames now do not contain color information (see Figure 10).

In terms of programming, one should choose a properly located submatrix and then resize it into a new image at the same resolution as the original video stream, as suggested in Figure 11. Surprisingly, this simple step consumed more than 140 milliseconds. Our GPU GLSL implementation was five to six times faster than C++ and took about 25 milliseconds.

Nevertheless, 25 milliseconds is still too long for a real-time algorithm, which is why we next tried to obtain more speed from asynchronous calls. A special class was created for stabilizing frames on the GPU. This class immediately returns a result from the previous iteration stored in its image-buffer field and creates a `TBB::task` for processing the next frame. As a result, GPU processing is performed in the background, and the apparent cost and delay for the caller is equal to just copying a full frame. This trick was also applied to an expensive color-conversion procedure, and with further optimizations of the memory-access patterns, we achieved real-time processing performance.

Future Directions

GPUs were originally developed to accelerate the conversion of 3D scene descriptions into 2D images at interactive rates, but as they have become more programmable and flexible, they have also been used for the inverse task of processing and analyzing 2D images and image streams to create a 3D description, to control some application so it can react to the user or events in the environment, or simply to create higher-quality images or videos. As computer-vision applications become more commonplace, it will be interesting to see whether a different type of computer-vision processor that would be even more suitable for image processing is created to work with a GPU, or whether the GPU remains suitable even for this task. The current mobile GPUs are not yet as flexible as those on larger computers, but this will change soon enough.

OpenCV (and other related APIs such as Point Cloud Library) have made it easier for application developers to use computer vision. They are well-documented and vibrant open source projects that keep growing, and they are being adapted to new computing technologies. Examples of this evolution are the transition from a C to a C++ API in OpenCV and the appearance of the OpenCV GPU module. The basic OpenCV architecture, however, was designed mostly with CPUs in mind. Maybe it is time to design a new API that explicitly takes heterogeneous multiprocessing into account,

where the main program may run on a CPU or several CPUs, while major parts of the vision API run on different types of hardware: a GPU, a DSP (digital signal processor), or even a dedicated vision processor. In fact, Khronos has recently started working on such an API, which could work as an abstraction layer that allows innovation independently on the hardware side and allows for high-level APIs such as OpenCV to be developed on top of this layer while being somewhat insulated from the changes in the underlying hardware architecture.

Acknowledgments

We thank Colin Tracey and Marina Kolpakova for help with power analysis; Andrey Pavlenko and Andrey Kamaev for GLSL and NEON code; and Shalini Gupta, Shervin Emami, and Michael Stewart for additional comments. NVIDIA provided support, including hardware used in the experiments. 

References

1. Khronos OpenMAX standard; <http://www.khronos.org/openmax>.
2. Liu, F., Gleicher, M., Wang, J., Jin, H., Agarwala, A. Subspace video stabilization. *ACM Transactions on Graphics* 30, 1 (2011), 4:1–4:10.
3. Matsushita, Y., Ofek, E., Ge, W., Tang, X., Shum, H.-Y. Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 7 (2006), 1150–1163.
4. Newcombe, R.A., Izadi, S., et al. Kinectfusion: Real-time dense surface mapping and tracking. *IEEE International Symposium on Mixed and Augmented Reality* (2011), 127–136.
5. OpenCV library; <http://code.opencv.org>.
6. Point Cloud Library; <http://pointclouds.org>.
7. Szeliski, R. Image alignment and stitching: a tutorial. *Foundations and Trends in Computer Graphics and Vision* 2, 1 (2006), 1–104.
8. Willow Garage. Robot Operating System; <http://www.ros.org/wiki/>.

Kari Pulli is a senior director at NVIDIA Research, where he heads the Mobile Visual Computing Research team and works on topics related to cameras, imaging, and vision on mobile devices. He has worked on standardizing mobile media APIs at Khronos and JCP and was technical lead of the Digital Michelangelo Project at Stanford University.

Anatoly Baksheev is a project manager at Itseez. He started his career there in 2006 and was the principal developer of multi-projector system Argus Planetarium. Since 2010 he has been the leader of the OpenCV GPU project. Since 2011 he has worked on the GPU acceleration module for Point Cloud Library.

Kirill Korniyakov is a project manager at Itseez, where he leads the development of OpenCV library for mobile devices. He manages activities on mobile operating-system support and computer-vision applications development, including performance optimization for NVIDIA Tegra platform.

Victor Eruhimov is CTO of Itseez. Prior to co-founding the company, he worked as a project manager and senior research scientist at Intel, where he applied computer-vision and machine-learning methods to automate Intel fabs and revolutionize data processing in semiconductor manufacturing.

DOI:10.1145/2184319.2184338

A few hubs with many connections share with many individuals with few connections.

BY BENJAMIN DOER, MAHMOUD FOUZ, AND TOBIAS FRIEDRICH

Why Rumors Spread So Quickly in Social Networks

UNDERSTANDING STRUCTURAL AND algorithmic properties of complex networks is important, due in part to the Internet's global social and commercial importance. Our focus here is to analyze how news spreads in social networks, simulating a simple information-spreading process in various network topologies and demonstrating that news spreads much more quickly in existing social-network topologies than in other network topologies. We support this finding by analyzing information spreading in the mathematically defined preferential attachment (PA) network topology, a common model for real-world networks, proving that sublogarithmic time suffices to spread news to all nodes of a network. All previously



This visualization by Miguel Rios at Twitter shows the volume of @replies traveling into and out of Japan and worldwide retweets in the one-hour period just before and after the Tōhoku earthquake on March 11, 2011. For an animated version visit <http://blog.twitter.com/2011/06/global-pulse.html>

studied network topologies need at least logarithmic time. Surprisingly, nodes with few neighbors are crucial for quick dissemination.

Social networks like Facebook and Twitter are reshaping the way people communicate and take collective action, playing a crucial role in, for example, the 2011 Arab Spring uprisings and London riots. It has been argued that the “instantaneous nature” of these networks influenced the speed

» key insights

- Social networks grow naturally; their graph structure is not designed for any particular use but still allows for the quick spread of news.
- Our mathematical proof shows rumors in social networks spread much more quickly than in most other network topologies, even complete graphs.
- The source of the speedy spread of information is fruitful interaction between the few nodes with many neighbors and the large number of nodes with few neighbors.



actual real-world events unfolded.⁴ Social networks spreading news so quickly is remarkable; the structure of social networks and the process that distributes news were not designed for this purpose. On the contrary, they were not designed at all but evolved in a random and decentralized manner.

Is it correct to assume social networks ease the spread of information (“rumors”)? And, if they do, which of their properties make it possible? To answer, we simulated a simple rumor-spreading process on several graphs reflecting the structure of existing large social networks (see Figure 1). For example, a rumor begun at a random node of the Twitter network reaches on average 45.6 million of the total of 51.2 million members within only eight rounds of communication.

We also analyzed this process on an abstract model of social networks, the so-called PA graphs introduced in 1999 by Barabási and Albert.³ In Doerr et al.,¹⁷ we devised a mathematical

proof that rumors in these networks spread much more quickly than in many other network topologies—even quicker than in networks with a communication link between any two nodes (complete graphs). As an explanation, we observe that nodes with relatively few neighbors build a shortcut between nodes with large degree (hubs) that, due to their large number of possible communication partners, talk less often to each other directly.

Rumor Spreading

Social networks arise in a variety of contexts, formed by people connected by knowing each other, Facebook members agreeing to be friends (in Facebook), scientific authors having a joint publication, and actors appearing in the same production.

Despite this diversity, many different types of networks share characteristic properties. Well known is that any two individuals are connected through just “six degrees of separa-

tion,” as first formulated in 2003 by Frigyes Karinthy (see Barabási²) and became known to a broad audience through Stanley Milgram’s “small world study.”²⁵ Likewise, for the Web, Albert et al.¹ predicted a diameter (maximum distance between two nodes in the graph) of only 19 connections in the network formed by Web pages and the links between them.

Several experimental studies^{1,13,22} have revealed another intrinsic property of social networks: The histogram of node connectivity follows a power law; the number of nodes with k neighbors is inversely proportional to a polynomial in k . To explain this phenomenon, Barabási and Albert³ suggested the PA model for real-world networks that show a power law. The model is widely used, due to its simplicity, and the article was the fifth most cited in *Science*, as of April 2012, according to ISI Web of Knowledge (<http://www.webofknowledge.com/>). Graphs in the PA model are construct-

ed in a random, “rich-get-richer” fashion; a newly entering node connects to m existing nodes chosen randomly but gives preference to nodes that are already popular; that is, they have many neighbors. Note that the parameter m controls the density of the graph, or the ratio of the number of present edges to the number of all possible edges. For these graphs, Barabási and Albert³ empirically discovered a power law of k^{-3} later proved mathematically by Bollobás et al.¹¹ Similar models emerged at the time, all leading to a power-law distribution. Also known is the PA model does

not share all properties observed in real-world networks; for example, it is less clustered.

Still, the PA model has helped deduce many interesting properties of social networks. Famous structural results prove small diameters for such graphs,¹⁰ determine their degree (in terms of number of neighbors) distribution,¹¹ and show high expansion properties²⁴ and robustness against random damage, along with vulnerability against malicious attacks.^{8,9,15,18} Algorithmic work shows that in such networks, viruses spread more easily than in many other network topologies⁵ and that gossip-based decentralized algorithms are better at approximating averages.¹²

Here, we assume a rumor is sufficiently interesting so people learn it when talking to someone who knows it. This is substantially different from the probabilistic virus-spreading model⁵ where the probability of being infected is proportional to the number of neighbors infected. Two types of rumor-spreading mechanisms have been identified in the literature: In the push model, only nodes that know the rumor contact neighbors to inform them, and is used to transmit information in computer networks.^{16,21} In contrast, to capture the effect of gossiping in social networks, it is more

appropriate for uninformed nodes to actively ask for new information. We use the push-pull model of Demers et al.¹⁶ (see also Karp et al.²⁰) in which all nodes regularly contact a neighbor and exchange all the information they have.

We assume nodes choose their communication partners uniformly at random from their neighbors, excluding the person they contacted immediately before. In this model, we regard the spread of a single piece of information initially present at a single node. For simplicity, as in most previous work, the rumor-spreading process is synchronized; that is, it takes place at discrete points over time and each time step, with each node contacting a neighbor to exchange information. This simplifies a real-world scenario where users act at different speeds, but in sufficiently large networks these differences balance out and thus assume an average speed used by all nodes.

Note that the communications process is different in each social network. The push-pull model we analyze is naturally best at capturing personal communication between two individuals by, say, phone, text message, email message, or other directed communication. Many on-line social networks also allow other ways to communicate (such as posts on personal Web pages), possibly resulting in friends being notified of a post when they next log in and then forwarding the news, given that it is of sufficient interest. Such forms of communication can be modeled only by more complicated means than the push-pull protocol.

Experimental Results

Supporting the notion that news spreads quickly in social networks, we have simulated the rumor-spreading process on samples from the Twitter and Orkut social networks (from Bhattacharjee et al.^{6,23}), as well as on PA graphs. As most social networks have roughly similar structure, we chose these large networks (data readily available) as instances of social networks. For comparison, we also included complete graphs and random-attachment graphs (also called the m -out model, as in, for example,

Figure 1. How a rumor progresses from a large-degree node A to another node B; due to their having large number of neighbors, A and B need more time to push or pull the rumor.

With fewer neighbors, C pulls the rumor from A and quickly pushes it to B.

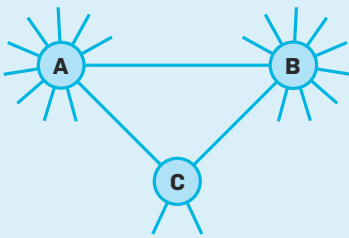
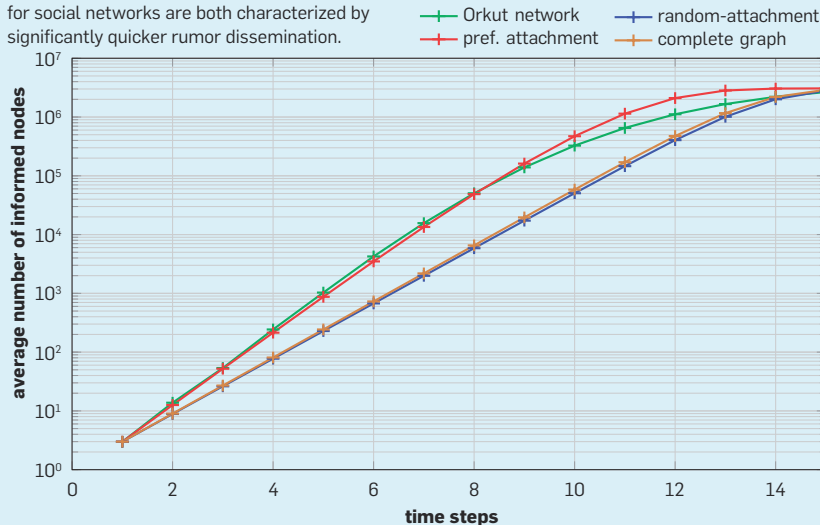


Figure 2. Average number of informed nodes as a function of time for the Orkut network and preferential attachment, random attachment, and complete graphs of the same size $n = 3,072,441$ nodes and density parameter $m = 38$.

The real network and the mathematical model for social networks are both characterized by significantly quicker rumor dissemination.



Bohman and Frieze⁷) in which each node chooses m neighbors uniformly at random from all nodes.

Note that in both the random attachment (RA) and the PA graph model, we are able to control the density of the graph through the parameter m , allowing us to run experiments on graphs with the same number of nodes and density as equivalent real-world graphs.

Figures 2 and 3 reflect how a rumor begun in a random node spreads in graphs corresponding to the Orkut and Twitter social networks. News spreads much more quickly in real-world networks and in PA graphs than in complete and RA graphs. In the Twitter experiment, a considerable difference is apparent between the PA model and the real-world graph, indicating the Twitter graph is captured less well by the theoretical model.

To determine how graph size might influence the speed a rumor spreads, we ran the process on PA graphs, RA graphs, and complete graphs of varying size. The results (see Figure 4) indicate logarithmic time is needed for RA and complete graphs, whereas for PA graphs time is needed of a slightly smaller order of magnitude.

Mathematical Analysis

We supported this empirical finding through mathematical analysis, proving the rumor-spreading process disseminates news in sublogarithmic time, or the time needed to inform all nodes, as well as any constant fraction (such as 1%, 10%, or 50%).

We denote by G_m^n the PA graph on n nodes with density parameter m . Since the graph G_m^n is a random graph, none of the statements mentioned earlier holds with certainty. However, the probability that the random graph G_m^n does not satisfy our assumptions and observations tends to zero for n growing to infinity. In the following paragraphs, when we make a statement concerning a random object, that statement is meant to hold with high probability. For the PA graph G_m^n , with m being any constant larger than one, we were able to prove that after a surprisingly short time any given news item spreads to all nodes.

Theorem 1. There is a constant K such that the rumor-spreading pro-

Surprisingly, nodes with few neighbors are crucial for quick dissemination.

cess we described spreads a rumor from any starting node to all other nodes in at most $K \cdot \log(n)/\log(\log(n))$ time steps. This result improves the previous best bound for rumor spreading in PA graphs¹⁴ of order $\log(n)^2$. Theorem 1 showed for the first time (in 2011) that rumor spreading in PA graphs is much quicker than for the other network topologies covered here so far. For RA graphs, it is easy to see that the diameter is of order $\log(n)$, which is also clearly a lower bound for rumor-spreading time. Similar reasoning also shows that, independent of starting node, a constant fraction of all nodes has distance $\Theta(\log(n))$ from the starting node. Hence a logarithmic number of rounds is needed to inform any constant fraction of the nodes. Similar bounds follow for hypercube networks common in computer science applications.

However, the diameter of the network does not always reveal the time needed to spread a rumor. The complete graph has a diameter of exactly one, but time to all other networks is of logarithmic order. This result was proved in 2000 by Karp et al.²⁰ for a rumor-spreading process in which nodes were allowed to choose their random communication partners from among all neighbors, including the one they might have just talked to. It is not difficult to see the Karp et al. proof is valid in our setting. That proof also shows that a logarithmic number of rounds is still necessary to inform any constant fraction of the nodes. Similar results hold for the classical Erdős-Rényi random graphs as might be deduced from Fountoulakis et al.¹⁹ and Karp et al.²⁰ All these results were proved through mathematical means; that is, they did not rely on experiments conducted for certain graph sizes n but are valid for all graph sizes. For the proof of Theorem 1 see Doerr et al.,¹⁷ though here we outline a main argument that also explains why rumor spreading in social networks is so speedy.

Toward this goal, let A and B be neighboring nodes in G_m^n . We denote their degrees by d_A and d_B . Assume that A is informed and B is not. How does the rumor progress from A to B ? Since A contacts its neighbors randomly, it will take approximately d_A rounds

until A contacts B ; thus A pushes the news to B . Likewise, it takes an expected number of approximately d_B time steps until B calls A , pulling the rumor from there. If d_A and d_B are large (such as $n^{1/3}$), then it would take an expected number of almost $1/2n^{1/3}$ rounds to

propagate the rumor from A to B along the direct link.

Theorem 1 established a much smaller bound. Hence there must be a better way to get the news from A to B . It is the small-degree nodes that make the difference. Now assume there is

a third node C that is a neighbor of both A and B and has a small degree d_C of, say, only four neighbors. After an expected number of roughly $d_C = 4$ rounds, C will have contacted A and thus learned the news from A . Likewise, after another expected number of $d_C = 4$ rounds, C will have contacted B and told it the news. That is, in $2 \cdot d_C = 8$ time steps, the rumor went from A to B through C . Fortunately, such a node C exists with high probability; the PA rule ensures newly entering nodes put enough preference on connecting with A and B .

One mechanism enabling the spread of rumors in social networks is that small-degree nodes learn the rumor from an informed neighbor, then quickly forward it to all other neighbors. In a sense, they act as an automatic link between their neighbors; once one neighbor is informed, then all other neighbors are informed—without doing anything. Such a mechanism is missing (such as in complete graphs) because all nodes have a high degree of $n - 1$. Consequently, all neighbors of the starting-node A have a small probability of calling A and asking for the news; A is just one of their $n - 1$ neighbors.

Also note that such high-speed links are abundant in PA graphs. To clarify, call a node popular if it has a degree of $\Theta(\log(n)^2)$ or higher. We can now show that between any two popular nodes, there is a path of length $O(\log(n)/\log(\log(n)))$ such that every second node on the path has the minimal possible degree of m . As per our assumptions, equations, and observations these nodes function as quick links, propagating rumors in an expected number of roughly $2 \cdot m$ rounds. Consequently, the expected time a rumor must traverse the whole path is about m times its length. With extra care, namely by showing there is a huge number of such paths between any two popular nodes, we can even show that once a popular node is informed, after $O(\log(n)/\log(\log(n)))$ rounds with high probability, all popular nodes are informed.

Since nodes tend to attach to popular nodes, a rumor started in a small-degree node is propagated to some popular node even quicker, namely in $O(\log(n)^{3/4} \log(\log(n)))$ rounds. Once all

Figure 3. Average number of informed nodes over time for the Twitter network and comparable preferential attachment, random attachment, and complete graphs (size $n = 51,217,936$ nodes, density parameter $m = 32$).

The two plots modeling social networks again outperform the random attachment and complete graph.

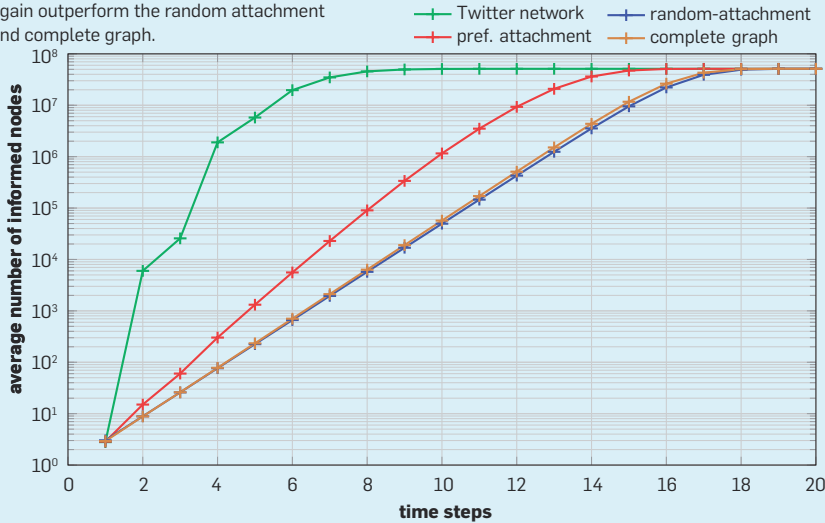
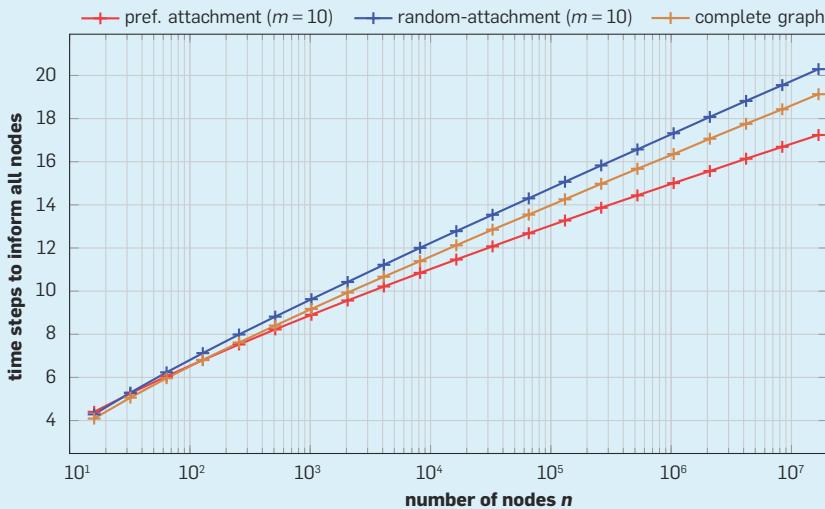


Figure 4. Average time needed to inform all nodes of different networks of varying size; the data shows logarithmic dependence for random-attachment and complete graphs.

For preferential attachment graph (a mathematical model for social networks), the times appear to be of lower order. Note that the generally speedy propagation partially obscures the advantage of the preferential attachment network structure. Alternatively, if one is willing to include 16 rounds of rumor-spreading communications between people (on average), then one can share a rumor with more than 300,000 others, as in the preferential attachment model, but with only around 30,000 people organized in random-attachment fashion.



popular nodes are informed, a symmetric argument can show that after another $O(\log(n)^{3/4} \log(\log(n)))$ rounds, the remaining small-degree nodes, mostly by calling more popular nodes, would all be informed; for more, see Doerr et al.¹⁷

Conclusion

We simulated a natural rumor-spreading process on various graphs representing real-world social networks and several classical network topologies. We also performed a mathematical analysis of the process in PA graphs. Simulation and analysis both demonstrate the speediness of rumor spreading in social networks.

A key observation in the mathematical proof, as well as being a good explanation for this phenomenon, is that small-degree nodes learn a rumor once one of their neighbors knows it, then quickly forward it to their neighbors. This propagation scheme facilitates sending rumors from one large-degree node to another.

How does this play out in everyday life? It partially explains why social networks are observed to spread information quickly, even though the process is not organized centrally, and the network is not designed in an intelligent way. Crucial is fruitful interaction between hubs with many connections and average users with few friends. Hubs make the news available to a big audience, whereas average users quickly convey the information from one neighbor to the next. 

References

1. Albert, R., Jeong, H., and Barabási, A.-L. Diameter of the World-Wide Web. *Nature* 401 (Sept. 1999), 130–131.
2. Barabási, A.-L. *Linked: How Everything Is Connected to Everything Else and What It Means*. Plume, 2003.
3. Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *Science* 286 (Oct. 1999), 509–512.
4. Beaumont, P. The truth about Twitter, Facebook, and the uprisings in the Arab world. *The Guardian* (Feb. 25, 2011); <http://www.guardian.co.uk/world/2011/feb/25/twitter-facebook-uprisings-arab-libya>
5. Berger, N., Borgs, C., Chayes, J.T., and Saberi, A. On the spread of viruses on the Internet. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms* (Vancouver, B.C., 2005), 301–310.
6. Bhattacharjee, B., Druschel, P., Gummadi, K. et al. Online Social Networks Research @ The Max Planck Institute for Software Systems; <http://socialnetworks.mpi-sws.org>
7. Bohman, T. and Frieze, A.M. Hamilton cycles in 3-out. *Random Structures & Algorithms* 35, 4 (2009), 393–417.
8. Bollobás, B. and Riordan, O. Robustness and vulnerability of scale-free random graphs. *Internet Mathematics* 1, 1 (2003), 1–35.
9. Bollobás, B. and Riordan, O. Coupling scale-free and



A rumor begun at a random node of the Twitter network reaches on average 45.6 million of the total of 51.2 million members within only eight rounds of communication.



- classical random graphs. *Internet Mathematics* 1, 2 (2003), 215–225.
10. Bollobás, B. and Riordan, O. The diameter of a scale-free random graph. *Combinatorica* 24, 1 (2004), 5–34.
11. Bollobás, B., Riordan, O., Spencer, J., and Tusnády, G. The degree sequence of a scale-free random graph process. *Random Structures & Algorithms* 18, 3 (2001), 279–290.
12. Boyd, S.P., Ghosh, A., Prabhakar, B., and Shah, D. Randomized gossip algorithms. *IEEE Transactions on Information Theory* 52, 6 (2006), 2508–2530.
13. Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. Graph structure in the Web. *Computer Networks* 33, 1–6 (2000), 309–320.
14. Chierichetti, F., Lattanzi, S., and Panconesi, A. Rumor spreading in social networks. *Theoretical Computer Science* 412, 24 (2011), 2602–2610.
15. Cooper, C., Frieze, A.M., and Vera, J. Random deletion in a scale-free random graph process. *Internet Mathematics* 1, 4 (2004), 463–483.
16. Demers, A.J., Greene, D.H., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H.E., Swinehart, D.C., and Terry, D.B. Epidemic algorithms for replicated database maintenance. *Operating Systems Review* 22, 1 (1988), 8–32.
17. Doerr, B., Fouz, M., and Friedrich, T. Social networks spread rumors in sublogarithmic time. In *Proceedings of the 43rd ACM Symposium on Theory of Computing* (San Jose, CA). ACM Press, New York, 2011, 21–30.
18. Flaxman, A.D., Frieze, A.M., and Vera, J. Adversarial deletion in a scale-free random graph process. *Combinatorics, Probability & Computing* 16, 2 (2007), 261–270.
19. Fountoulakis, N., Huber, A., and Panagiotou, K. Reliable broadcasting in random networks and the effect of density. In *Proceedings of the IEEE Conference on Computer Communications* (San Diego, Mar. 15–19). IEEE Press, 2010, 2552–2560.
20. Karp, R., Schindelhauer, C., Shenker, S., and Vöcking, B. Randomized rumor spreading. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science* (Redondo Beach, CA, Nov. 12–14). IEEE Computer Society Press, 2000, 565–574.
21. Kempe, D., Dobra, A., and Gehrke, J. Gossip-based computation of aggregate information. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science* (Cambridge, MA, Oct. 11–14). IEEE Computer Society Press, 2003, 482–491.
22. Kumar, S.R., Raghavan, P., Rajagopalan, S., and Tomkins, A. Extracting large-scale knowledge bases from the Web. In *Proceedings of the 25th International Conference on Very Large Data Bases* (Edinburgh, Scotland, Sept. 7–10). Morgan Kaufmann, 1999, 639–650.
23. Leskovec, J. Stanford Large Network Dataset Collection; <http://snap.stanford.edu/data/>
24. Mihail, M., Papadimitriou, C., and Saberi, A. On certain connectivity properties of the Internet topology. *Journal of Computer and System Sciences* 72, 2 (2006), 239–251.
25. Milgram, S. The small-world program. *Psychology Today* 2 (1967), 60–67.

Benjamin Doerr (doerr@mpi-inf.mpg.de) is a senior researcher in the Department of Algorithms and Complexity at Max-Planck-Institut für Informatik, Saarbrücken, Germany, and an adjunct professor at Saarland University, Saarbrücken, Germany.

Mahmoud Fouz (mahmoudfouz@rocket-internet.com.tr) is a managing director of Rocket Internet, Dubai, U.A.E.; this work was done while he was a Ph.D. student in the Computational Complexity group of the Department of Computer Science at Saarland University, Saarbrücken, Germany.

Tobias Friedrich (tfriedrich@mpi-inf.mpg.de) is an independent research group leader in the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University, Saarbrücken, Germany, and a senior researcher in the Department of Algorithms and Complexity at Max-Planck-Institut für Informatik, Saarbrücken, Germany.

DOI:10.1145/2184319.2184339

A user's trust in a single device can be extended to many other devices.

BY BRYAN PARNO

Trust Extension for Commodity Computers

AS SOCIETY RUSHES to digitize sensitive information and services, users and developers must adopt adequate security protections. However, such protections often conflict with the benefits expected from commodity computers. Consumers and businesses value commodity computers because they provide good performance and an abundance of features at relatively low cost, but attempts to construct secure systems from the ground up are expensive, time-consuming, and unable to keep up with the changing marketplace.^{2,8,11,12} For example, the VAX VMM security kernel was developed over nine years (1981–1990), but the kernel was never deployed. This failure was due, in part, to the absence of support for Ethernet, a feature considered crucial by the time the kernel was completed but not anticipated when originally designed.¹¹

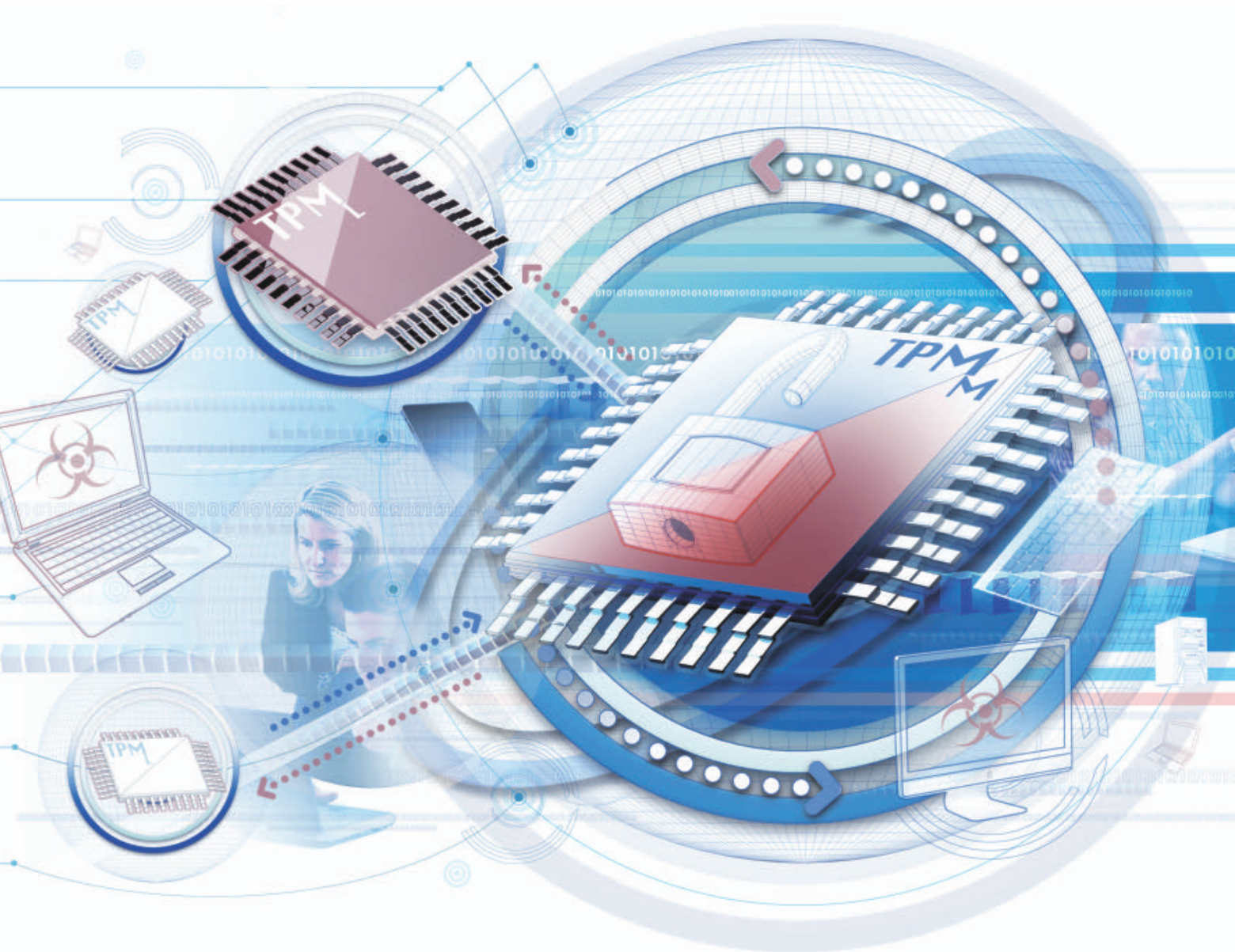
Rather than build secure systems from scratch, the tension between security and features can be resolved by extending the trust users have in one device to enable them to use another commodity device or service securely without sacrificing the performance, features, or cost expected of commodity systems.²¹ Note this article focuses on average users and commodity systems rather than advanced users, special-purpose computers, or highly constrained environments (such as those in the military).

At a high level, this premise is supported by developing software, hardware, and cryptographic techniques to extend user trust in a small special-purpose hardware device to provide strong security protection for both local and remote computation on sensitive data while preserving the performance and features of commodity computers.

Included is an overview of hardware security technologies (see the sidebar “Security Features in Commodity Computers”), how to extend trust in a special-purpose mobile device to verify the security hardware in a user’s local machine, how to extend that trust in a meaningful way to software on the local machine, how to extend trust in that software to network elements, and finally how to extend that trust to remote computers where neither software nor hardware is trusted; for a detailed comparison with related work see other publications.^{21,23}

» key insights

- Improving software security is insufficient; also needed is the ability to securely verify whether a computer employs the new software.
- Providing security on demand (such as via the Flickr architecture) helps balance security, performance, and features.
- Verifiable computation allows a client to outsource the computation of a function and efficiently verify the results returned while keeping inputs and outputs private; constraining the way the worker/server computes the function enables such efficient verification.



Trust Extension

Creating trustworthy systems requires advances on many fronts, including programming languages, operating systems, network protocols, and definitions of security. More fundamentally, both computers and users must be able to make accurate, informed trust decisions. After all, even if software improves, users must be able to determine which systems employ the new, improved software. In particular, users must be able to judge whether a system (local or remote) should be trusted before handing over sensitive data.

This article describes techniques that provide firm evidence on which to base trust decisions. Once a user decides to trust a particular device (such as a cellphone), it shows how to use that device to verify, on the

user's behalf, the trustworthiness of other devices and services (such as laptop computers and cloud-based computation). The user's trust in the original device inspires trust in its assessment of the new device or service. If the assessment is positive, the user has essentially extended trust in the first device to include the new device or service.

Informally, this is our definition of trust: To trust an entity X with private data (or a security-sensitive task), users must believe that at no point in the future will they have cause to regret having given their data (or entrusted the task) to X .

Bootstrapping Trust

The problem of allowing users to bootstrap trust in their own personal computers is fundamental and

should be easier than other potential scenarios; if users cannot establish trust in their computers, they are unlikely to be able to trust a remote computer. When working with their own computers, users can be reasonably certain those computers are physically secure; that is, an attacker has not tampered with their hardware configurations. Such an assumption aligns naturally with standard human intuition about security; a resource (such as a physical key) physically controlled by an individual is typically more secure than a resource a user might give someone else. Fortunately, the physical protection of valuable items has been a major focus of human ingenuity for millennia.

If a user's computer is physically secure, then special-purpose secure hardware can be used to monitor and

Security Features in Commodity Computers

While other instantiations are possible, the following techniques are described in terms of features of trusted platform modules, or TPMs, and recent CPUs from AMD and Intel:

Measurement. When TPM-equipped platforms first boot, platform hardware takes a measurement (SHA-1 hash) of the BIOS, recording it in one of the TPM's Platform Configuration Registers (PCRs).^{25,28} The BIOS is then responsible for measuring the next software component (such as the bootloader) and associated data files. The BIOS records the measurement in a PCR before executing the software. If each subsequent software component performs these steps—measure, record, execute—the TPM holds a set of measurements of all code executed on the platform.²⁵

Attestation. To securely convey measurements to an external verifier, the TPM creates attestations; with a verifier-supplied nonce, the TPM uses a private key never accessible outside the TPM to generate a TPM quote, or a digital signature over the nonce and the content of the PCRs. The nonce assures the verifier that the attestation is fresh and not copied from a previous boot cycle.

To ensure the attestation comes from a real hardware TPM, not a software emulation, the TPM includes an endorsement keypair and endorsement certificate for the public key from the platform's manufacturer declaring it belongs to a real TPM. To preserve user privacy, the TPM uses attestation-identity keys (bound anonymously to the endorsement key) to sign attestations.

Secure storage. The TPM also includes a limited amount of nonvolatile RAM (NVRAM). Reading and writing to NVRAM can be restricted based on PCR content, so an NVRAM location can be made accessible only to a particular set of software. That software can use the protected NVRAM to store a symmetric key that can then be used to encrypt and integrity-protect bulk data.

Dynamic root of trust for measurement. To protect the launch of a virtual machine monitor (VMM), recent commodity CPUs from AMD and Intel extend the x86 instruction set to support a dynamic root of trust for measurement (DRTM) operation.^{1,10} DRTM provides many of the security benefits of rebooting the computer (such as starting from a clean slate) while bypassing the overhead of a full reboot; that is, devices remain enabled, the BIOS and bootloader are not invoked, and memory contents remain intact. DRTM is implemented through a new SKINIT instruction on AMD CPUs or GETSEC[SENDER] on Intel CPUs, which can launch a VMM at an arbitrary time (hence the colloquialism late launch) with built-in protection against software-based attacks. When a DRTM is invoked, the CPU's state is reset, and direct memory access (DMA) protections for a region of memory are enabled. The CPU hashes the content (such as data and executable code) of the memory region, extends the measurement into the TPM's PCR 17 (and 18 on Intel), and begins executing the code.

Trust assumptions. Relying on a TPM's guarantees involves trusting the TPM manufacturer, the platform vendor, and the PKI linking them to the TPM's keys. The CPU, RAM, and chipset must also be trusted. The TPM is designed to withstand software-based attacks and limited hardware attacks (such as rebooting the machine) but is not expected to resist sophisticated physical attacks.

report the software state of the platform. Given the software state, users (or their agents) can decide whether the platform should be trusted. While a full-blown secure coprocessor (such as the IBM 4758)²⁷ might appeal, cost limits deployment. However, as of 2010, more than 350 million³⁰ commodity computers had been sold worldwide with a special-purpose security chip called the TPM, as described in the sidebar.²⁸ With appropriate software support, it can be used to report on the software executing on the computer. The resulting attestation can be verified by a user's trusted device (such as a cellphone or special-purpose USB fob).²⁹ Thus, the user's trust in the de-

vice can be extended, via the TPM, to establish trust in the software on the machine. For example, Garriss et al. used a cellphone to verify a virtual machine monitor on a kiosk computer,⁵ though a kiosk might not satisfy the hardware security assumptions discussed earlier.

However, given even a TPM-equipped machine, the question remains: How does one bootstrap trust in the TPM itself? Surprisingly, neither the TPM specifications nor the academic literature addresses this problem. They instead tend to assume the user magically possesses the TPM's public key. Unfortunately, any straightforward approach to trusting the TPM

falls victim to a cuckoo attack^{a,20,21} in which the adversary convinces the user that a TPM the adversary physically controls resides in the user's own local computer; Figure 1a outlines a possible implementation of the cuckoo attack. Malware on the user's local machine proxies the user's TPM-related messages to a remote, TPM-enabled machine controlled by an attacker. Any credential the user requests from the local TPM (such as an endorsement certificate) can also be provided by the attacker's TPM and relayed to the malware on the user's machine.

To analyze a cuckoo attack, we developed a model^{20,21} using predicate logic for bootstrapping trust in a local computer with secure hardware. The model defines a set of predicates describing the trustworthiness of people, computers, and secure hardware (such as TPMs), as well as a set of trust axioms. For example, if a trusted person P says computer C is secure, C can be seen as physically secure, thus encoding our previous assumption that people are reasonably effective at assessing and maintaining physical security.

Trying to use this model to reason about the security of the user's local machine brings us to a logical contradiction,^{20,21} namely that the local machine is both trusted and untrusted, thus capturing the essence of the cuckoo attack; the user cannot decide whether or not to trust the local machine.

From Figure 1a, it may seem a cuckoo attack can be prevented by severing the connection between the local malware and an adversary's remote PC. The assumption is that, without a remote TPM to provide the correct responses, the infected machine must either refuse to respond or allow the true TPM to communicate with the user's device, revealing the presence of the malware.

However, regardless of implementation, cutting off network access fails to prevent a cuckoo attack. As outlined in Figure 1b, a cuckoo attack is possible because the malware on the local machine has access to a "TPM oracle"

a Cuckoos replace other birds' eggs with their own, tricking victim birds into feeding the cuckoo chick as if it were their own; likewise, the attacker "replaces" the user's trusted TPM with his own TPM, leading the victim to treat the attacker's TPM as the user's own.

that provides TPM-like answers without providing TPM security guarantees. Since the adversary has physical possession of his own TPM_M , he can extract its private keys and credentials and include them with the malware. Thus provisioned, the malware on the local machine can emulate TPM_M , even without network access.

The right way to prevent a cuckoo attack is to introduce a secure channel to the local TPM, instantiated through either a physically hardwired channel allowing users to connect directly to the TPM on the local machine, or a mechanism to allow the user to learn authentic cryptographic information about the local TPM, and hence establish a cryptographic secure channel.

Both the formal model and a usability assessment were used to analyze a dozen instantiations of these approaches,^{20,21} including adding a special-purpose port or button to the computer, reusing existing interfaces (such as USB and FireWire), using software-only attestation, and encoding cryptographic material as a 2D barcode or serial number printed on the computer. Each involves advantages and disadvantages. In the short term, placing an alphanumeric hash of the TPM's public key on the exterior of the computer seems to offer the best trade-off among security, usability, and deployability. In the long term, the strongest security would come from a special-purpose hardware interface wired directly to the machine's secure hardware (such as the TPM), designed so the user is unable to inadvertently connect the verifying device to another interface. This solution removes almost any opportunity for user error and does not require preservation of secrets.

Executing Code Securely

Establishing a secure connection between a user and the security hardware on the user's computer does not, unfortunately, provide a full-featured, trustworthy execution environment. Security hardware tends to be either resource-impooverished or special-purpose (or both). Hence, the user's trust in security hardware must be extended to the user's entire computer.

However, establishing truly secure functionality on a general-purpose computer raises this crucial question:

How can secure code execution coexist with the untrustworthy mountain of buggy yet feature-rich software common on modern computers? For example, how can a user's keystrokes be kept private if the operating system, the most privileged software on the computer, cannot be trusted to be free of vulnerabilities? This coexistence is even more challenging due to the need to preserve the system's existing functionality and performance.

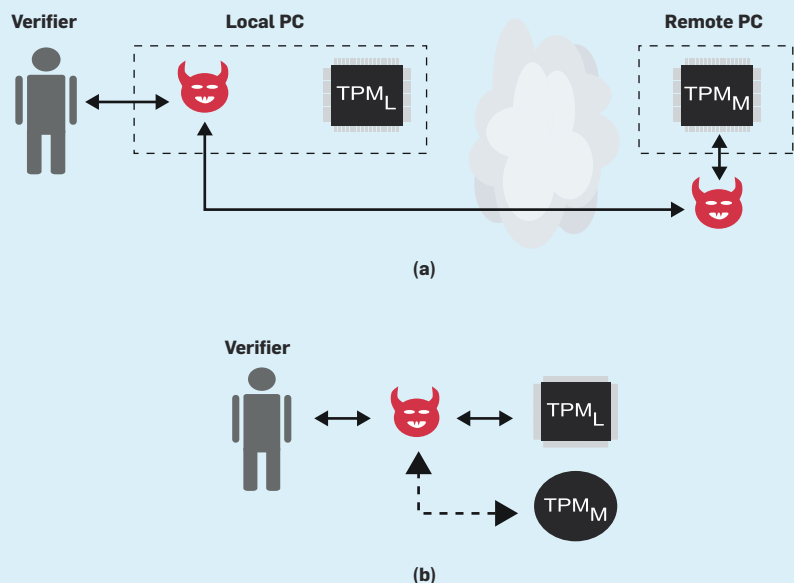
Previous work sought to deal with the problem by running a persistent security layer, dubbed a security kernel, VMM, or hypervisor, in the computer's most privileged mode.^{8,11,12,26} The security layer was responsible for creating isolation domains for ordinary, untrusted code and for security-sensitive code. Unfortunately, the approach also involves several drawbacks; for example, the security layer's need to interpose on hardware accesses leads to performance degradation for ordinary code and often requires eliminating access to devices too complicated to emulate (such as 3D graphics cards).⁴ Moreover, the need to run both untrusted

and trusted code simultaneously can lead to security vulnerabilities (such as side-channel attacks), as well as code bloat in the security layer; for example, the initial implementation of the Xen VMM in 2003 required 42,000 lines of code⁴ but almost doubled to 83,000 lines within a few years.¹³

To avoid these drawbacks, the Flicker architecture^{14,16–18,21} was developed to satisfy the need for features and security. Indeed, Flicker shows these conflicting needs can be satisfied by constructing a secure execution environment on demand. When invoked for secure code execution (such as signing a certificate or authenticating to a Web site), Flicker creates an isolated environment such that none of the software executing before Flicker begins can monitor or interfere with Flicker code execution, while all traces of Flicker code execution can be eliminated before regular software execution resumes. For example, a certificate authority could sign certificates with its private key and keep the key secret, even from an adversary with control of the BIOS, OS, and DMA-enabled devices.

Figure 1. Cuckoo attack.

In one implementation (a), malware on a user's local machine sends messages intended for the local TPM (TPM_L) to a remote attacker who feeds the messages to a TPM (TPM_M) inside a machine controlled (physically) by the attacker. Given physical control of TPM_M , the attacker is able to violate its security guarantees via hardware attacks. At a logical level (b), the attacker controls all communication between verifier and local TPM while also having access to an oracle that provides all the answers of a normal TPM without providing the security properties expected of a TPM.



Secure code execution. This section describes Flicker as implemented on an AMD system (see Figure 2); the implementation is similar on an Intel system. To take advantage of Flicker, application developers must provide the security-sensitive code (called the piece of application logic, or PAL) selected for Flicker protection as an x86 binary and define its interface with the rest of their application; tools have been developed to aid in automating this process. To create a secure loader block, or SLB (supplied as an argument to the DRTM instruction, *SKINIT*), application developers link PAL against a code module developed to perform the steps necessary to set up and tear down the Flicker session.

Since *SKINIT* is a privileged instruction, when executing the resulting SLB, the application passes it to a kernel module that allocates memory, initializes various values in the SLB, and handles untrusted setup and tear-down operations. The kernel module is not included in the trusted computing base (TCB) of the application, since its actions are verified.

Flicker achieves its properties through the DRTM capabilities outlined earlier. Rather than launch a VMM, it pauses the current execution environment (such as the untrusted OS), saving information about the kernel's page tables to memory, along with the content of key registers (such as CR0, CR3, GDTR, IDTR, and TR). On multicore machines, execution must be halted on all but the bootstrap processor, and the other cores must be sent an INIT inter-processor interrupt so they respond correctly to a handshaking synchronization step performed during execution of *SKINIT*.

The Flicker session begins with execution of the *SKINIT* instruction, which receives the SLB selected for Flicker protection as an argument. As described earlier, *SKINIT* resets the CPU state, adds entries to the device-exclusion vector to disable the DMA to the memory region containing the SLB, disables interrupts (including system-management interrupts) to prevent the previously executing code from regaining control, disables debugging support, even for hardware debuggers, and extends a hash of the SLB into a PCR in the TPM. Finally, it hands control



A cuckoo attack is possible because the malware on the local machine has access to a “TPM oracle” that provides TPM-like answers without providing TPM security guarantees.



to Flicker's initialization routine. To simplify execution of PAL code, Flicker loads the global descriptor table; loads the CS, DS, and SS registers; transitions to ring 3 (user space) execution; and calls the PAL, providing the address of PAL inputs as a parameter. When PAL execution terminates, Flicker cleans up any trace of the security-sensitive code's execution by clearing out memory and register content. Finally, it resumes the previous execution environment. This resumption entails returning to ring 0 (kernel) execution, restoring the saved OS state, re-enabling paging interrupts and jumping back to kernel code.

Since Flicker's protections are deployed only on demand, Flicker induces no performance overhead or feature reduction during regular computer use. Limiting Flicker's persistence also strengthens its security guarantees, since it avoids the complexity (and potential vulnerability) of solutions based on a VMM or a security kernel. Hence, the core Flicker system was implemented with a TCB of only 250 lines of code.

However, non-persistence poses challenges of its own. For instance, to enable more complex applications, TPM-based secure storage must be leveraged to maintain state across Flicker sessions. Preventing a variety of subtle attacks on this saved state requires developing additional protocols.²² A platform using Flicker can convince remote parties a Flicker session executed with a particular PAL.¹⁸ To provide result integrity, when PAL execution terminates, Flicker extends PCR 17 with hashes of the PAL's input and output parameters. It then extends PCR 17 with a fixed public constant, providing two powerful security properties. First, it prevents other software from extending values into PCR 17 and attributing them to the PAL; the fact that *SKINIT* resets PCR 17 to 0 prevents malicious software from extending values before the Flicker session. Second, it revokes access to any secrets kept in the TPM's secure storage that might have been available during PAL execution. Combining these techniques enables a PAL to communicate securely, with both secrecy and integrity protections, with a remote party through a secure cryptographic channel.

Flicker was implemented for both 32-bit Windows and Linux running on AMD and Intel CPUs (<http://flickertcb.sourceforge.net>) and includes useful code modules PALs can choose to include (such as for memory management, TPM operations, cryptographic operations, and secure channels). Using them, Flicker has since been applied to several broad classes of applications. To illustrate stateless applications, Flicker provides verifiable isolated execution of a kernel rootkit detector on a remote machine. To illustrate applications for which integrity protection of application state suffices, Flicker helps verify execution of a distributed computing application based on the Berkeley Open Infrastructure for Network Computing framework.³ Finally, to illustrate applications with state requiring secrecy and integrity, Flicker helps protect computations performed with the private key for a certificate authority, the passwords for an SSH server and for a full-disk encryption utility, and the content of a differentially private database.

To summarize this evaluation, invoking the `SKINIT` instruction requires $\sim 48\text{ms}$, while secure storage operations take $\sim 22\text{ms}$, meaning a typical Flicker session requires $\sim 70\text{ms}$, plus the time needed for the application-specific PAL to execute. Generating a TPM quote for the Flicker session is by far the slowest operation; depending on the TPM, that quote can take $\sim 362\text{ms}$ – 756ms . Fortunately, it can be computed in parallel with untrusted code since it is performed outside the Flicker environment. In general, Flicker has little effect on the performance of untrusted code, and the OS remains stable. Frequent brief Flicker sessions have little effect on system performance. Longer-running Flicker sessions may produce input lag or dropped network packets, but our experiments indicate Flicker does not corrupt data transfers (such as between USB and disk).

Suggested architectural improvements. Overall, experiments identified two significant performance bottlenecks for minimal TCB execution on current CPU architectures: inability to execute PALs and untrusted code simultaneously on different cores of a multicore machine and slow TPM op-

erations to protect PAL state during a context switch between secure and insecure execution.

Alleviating them involves a number of recommendations to hardware manufacturers^{14,17,21} to support an execution model in which multiple PALs can execute in parallel with legacy software. The key ones are a hardware mechanism that isolates the memory pages belonging to a PAL from all other code and for a hardware context switch mechanism that efficiently suspends and resumes PALs without exposing a PAL's execution state to other code.

A secure execution control block (SECB) is defined as a structure that holds PAL state and resource allocations for launching a PAL and for storing its state when not executing. Figure 3 outlines the proposed life cycle of a PAL. To begin execution of a PAL described by a newly allocated SECB, a new CPU instruction, secure launch, or `SLAUNCH`, can be added that takes as its argument the starting physical address of an SECB. It combines the hardware virtual machine management data structures on AMD and Intel CPUs with the security functionality of

a DRTM operation.

To isolate memory and facilitate rapid context switches, the memory controller can maintain an access-control table with one entry per physical page, where each entry specifies which CPUs (if any) have access to the physical page. Memory pages are by default marked `ALL` to indicate they are accessible by all CPUs and DMA-capable devices.

When PAL execution is started using `SLAUNCH`, the memory controller updates its access-control table so each page allocated to the PAL (specified by the list of memory pages in the SECB) is accessible to only the CPU executing the PAL. When the PAL is subsequently suspended, the state of its memory pages transitions to `NONE`, indicating nothing currently executing on the platform is allowed to read or write to those pages. Note that the memory allocated to a PAL includes space for data and is a superset of the pages containing the PAL binary. Since this data is ephemeral, the PAL must take steps (such as using the TPM's secure storage facilities) to ensure its data persists across reboots.²²

When PAL execution terminates, a

Figure 2. Timeline of steps needed to execute a PAL.

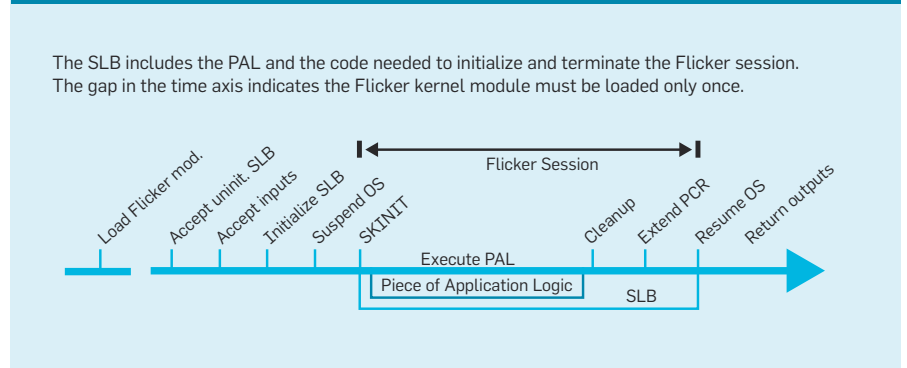
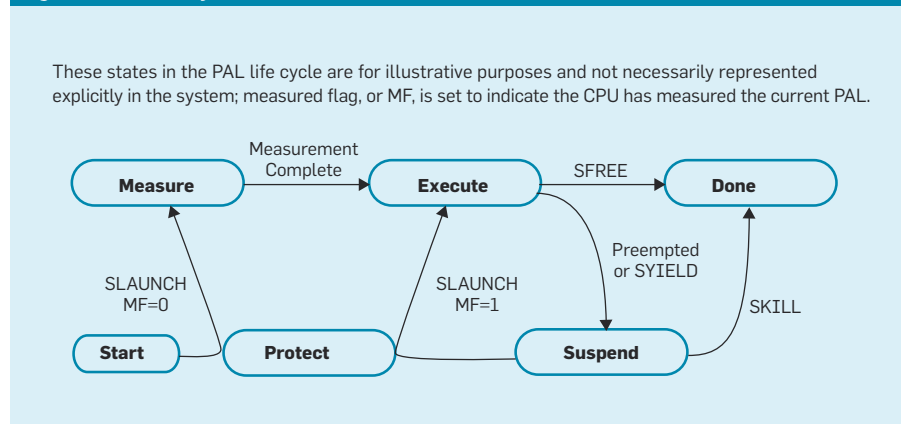


Figure 3. PAL life cycle.



well-behaved PAL calls `SFREE`, which clears the memory allocated to the PAL, as well as any microarchitectural state that might contain sensitive data, and marks the memory pages with `ALL`. Likewise, the untrusted OS may kill a PAL via `SKILL`, clearing the PAL's state before freeing the associated memory pages.

Also recommended is a PAL pre-emption timer in the CPU that can be configured by the untrusted OS. When the timer expires or a PAL voluntarily yields (via a new CPU instruction, `secure yield`, or `SYIELD`), the PAL's CPU state should automatically and securely be written to its SECB by hardware, all sensitive microarchitectural state should be cleared, and control should be transferred to an appropriate handler in the untrusted OS.

The untrusted OS can resume a PAL by executing `SLAUNCH` on the desired CPU, parameterized with the physical address of the PAL's SECB. The PAL's measured flag indicates to the CPU that the PAL has been measured and is only being resumed, not launched for the first time. During PAL resumption, the `SLAUNCH` instruction signals the memory controller that the PAL's state should be accessible to the CPU on which the PAL is executing. Note that the PAL may execute on a different CPU each time it resumes. To enable multi-core PALs, if a CPU attempts to resume an already executing PAL, that CPU will be added to the pool of CPUs available to the PAL.

The recommendations outlined here eliminate use of TPM operations during context switches and require the TPM to measure the PAL only once (instead of on every context switch). An implementation of these recommendations would be expected to achieve PAL context switch times on the order of those possible using hardware virtualization support, or $0.6\mu\text{s}$ on current hardware. This performance reduces the overhead of context switches by orders of magnitude, making switching in and out of a PAL much more practical.

Flicker thus provides a solid foundation for constructing secure systems that operate in conjunction with standard software; developers of security-sensitive code modules need to trust only their own code, along with as few as 250 lines of Flicker code, to protect the secrecy and integrity of the code's execution. Flicker guarantees these properties, even if the BIOS, OS, and DMA-enabled devices are all malicious. The current implementation (<http://flickertcb.sourceforge.net/>) offers reasonable performance for many applications; the hardware recommendations here would enable many more.

Leveraging Secure Code Execution

If secure code execution can be provided on endhosts, the next frontier would be to examine how such trust can be extended into the network to improve the performance and efficiency of network applications. That is, if endhosts (or at least portions of each endhost) can be

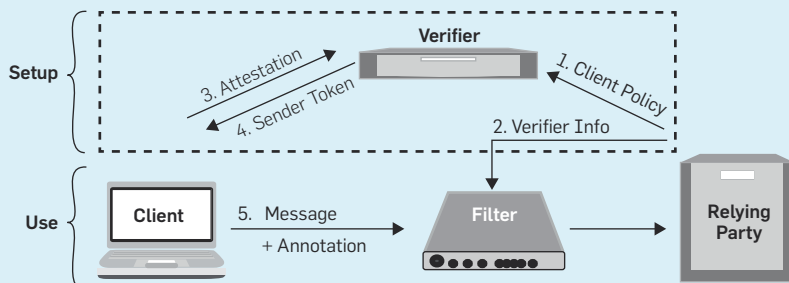
trusted, then network infrastructure no longer must arduously and imprecisely reconstruct data already known by the endhosts. For instance, suppose a mail server wants to improve the accuracy of its spam identification using host-based information. A 2009 study found the average and standard deviation of the size of email messages sent in the past 24 hours are two of the best indicators of whether a given email message is spam or legitimate.⁹ This data is easy for an endhost to collect but difficult for a single mail recipient to obtain.

As an initial exploration of how endhost hardware security features can be used to improve the network, a general architecture called Assayer was designed for securely conveying endhost information to network elements.^{21,24} While Assayer may not represent the optimal way to convey this information, it is a valuable first step for highlighting the various issues. For example, is it possible to provide useful host-based information while also protecting user privacy? Which cryptographic primitives are needed to verify this information in a secure and efficient manner? Initial findings suggest improved endhost security can improve the security and efficiency of the network while reducing the complexity of in-network elements.^{21,24}

Naively, hardware-based attestation, as discussed earlier, might be used for each piece of information sent to the network. Also returning to an earlier example, a mail server might ask each client to include an attestation in every email message. The mail server's spam filter could verify the attestation, then incorporate the host-provided information into its classification algorithm. Any legacy traffic arriving without an attestation could simply be processed by the existing algorithms. Unfortunately, checking attestations is time-consuming and requires interaction with the client machine. Even if such checks are feasible for an email filter, they would be unacceptable for other applications (such as distributed denial-of-service, or DDoS, mitigation) requiring per-packet checks at line rates. Likewise, labeling outbound traffic with mandatory access control labels, as proposed by McCune et al.¹⁵ works well in tightly

Figure 4. Assayer components.

The relying party (such as a mail server or an ISP) delegates inspection of clients to one or more verifiers and configures one or more filters with information about the verifiers. The filters check the client's annotation, then act on the information in the annotation; for example, a filter might drop the message or forward it at a higher priority to the relying party.



controlled enterprise networks but is unsuited to the heterogeneity of the Internet.

So how can the average case be made fast and noninteractive within the diversity of the Internet? The natural approach is to cryptographically extend the trust established by a single hardware-based attestation over multiple outbound messages. The cost of the initial verification is thus amortized over subsequent messages. To achieve this amortization, the Assayer architecture employs two distinct phases: infrequent setup in which the relying party (such as the mail server) establishes trust in the client and more frequent usage in which the client generates authenticated annotations on outbound messages (see Figure 4).

The relying party delegates the task of inspecting clients to one or more off-path verifier machines. Every T days, the client convinces a verifier it has securely installed a trustworthy code module that keeps track of network-relevant information (such as number of email messages recently sent and amount of bandwidth recently used).

The various code modules execute atop a minimal hypervisor that implements Flicker-like functionality with performance closer to what Flicker would provide with the hardware changes recommended earlier. To protect user privacy, code modules lack visibility into the client machine's software state; for example, a client module for Web browsing cannot determine which Web browser the client is using. Assayer instead employs application-specific incentives (such as giving priority to traffic carrying annotations) to convince the commodity software to submit outbound traffic to the Assayer modules on the client machine.

Having established the trustworthiness of the client, the verifier issues a limited-duration `sender token` that is bound to the client's code module. During the usage phase, client code submits outbound messages to an Assayer code module that uses the `sender token` to authenticate the message annotations it generates. These annotations are then checked by one or more fast-path filter middleboxes that verify the annotations and react accordingly. For instance, a relying party trying to identify spam might feed



How can a user's keystrokes be kept private if the operating system, the most privileged software on the computer, cannot be trusted to be free of vulnerabilities?



the authenticated information from the filter into its existing spam classification algorithms. Alternatively, a Web service might contract with its ISP to deploy filters on its ingress links to mitigate DDoS attacks by prioritizing legitimate traffic. If the traffic does not contain annotations, then the filter treats it as legacy traffic; for example, a DDoS filter would give annotated traffic priority over legacy traffic. Two protocols have been designed to instantiate the `sender tokens` and message annotations: one efficient symmetric-key-based and one less-efficient asymmetric-key-based offering additional security properties.^{21,24}

To evaluate the usefulness of trustworthy host-based information, the application of Assayer is considered in three case studies: spam identification, DDoS mitigation, and super-spreader worm detection. Assayer is well suited to combating spam and can mitigate many network-level DDoS attacks. Surprisingly, while it is technically feasible to use Assayer to combat super-spreader worms, which contact many hosts quickly, such use would face challenges when it comes to deployment incentives.

To better understand the performance implications of conveying host-based information to the network, a full Assayer prototype has been implemented, including multiple protocol implementations. The size of the protection layer on the client that protects code modules from the endhost (and vice versa) is minuscule (841 lines of code), and the individual modules are even smaller. The verifier prototype can sustain 3,300 client verifications per second and bursts of up to 5,700 clients per second. Generating and verifying annotations for outbound traffic requires only a few microseconds for Assayer's most efficient scheme. Even on a gigabit link, Assayer can check the annotations with a reasonable throughput cost of 3.7%–18.3%, depending on packet size.

Verifiable Remote Computation

With Flicker, a user's computer is assumed to be physically secure. To generalize Flicker's results, techniques are needed to establish trust in code execution when even the hardware is untrustworthy. This scenario is par-

ticularly compelling, as the growth of cloud computing and mobile devices contributes to the desire to outsource computing from a client device to an online service. In these applications, how can users be assured the secrecy of their data is protected? Equally important, how can users verify the results returned are correct, without redoing the computations?

While various forms of homomorphic encryption provide only data secrecy,⁷ the results of arbitrary tasks (abstracted as function evaluations) on a computational service (such as in the cloud) can be efficiently verified without trusting any hardware or software on the system.^{6,21} This result contrasts with previous approaches that were inefficient or that could verify only the results of restricted function families.

To formalize secure computational outsourcing we introduce the notion of verifiable computing.^{6,21} Abstractly, a client wishes to evaluate a function F (such as sign a document or manipulate a photograph) over various dynamically selected inputs $x_1, \dots, x_k$ on one or more untrusted computers, then verify that the values returned are indeed the result of applying F to the given inputs. The critical requirement is that the client's effort to generate and verify work instances must be substantially less than the work required to perform the computation.

The definition of verifiable computation here is non-interactive: The client sends a single message to the worker and vice versa. The definition also uses an amortized notion of complexity for the client, which can perform expensive preprocessing but following this stage is required to run efficiently.

Drawing on techniques from secure multiparty computation, the first protocol for verifiable computing, presented here, provably provides computational integrity for work done by an untrusted party; it also provides provable secrecy for the computation's inputs and outputs.^{6,21} This feature is bundled into the protocol at no additional computational cost. Such privacy is critical for many real-world outsourcing scenarios in which a function is computed over highly sensitive data (such as medical records and trade secrets).

Moreover, the protocol provides



Developers of security-sensitive code modules need to trust only their own code, along with as few as 250 lines of Flickr code, to protect the secrecy and integrity of the code's execution.



asymptotically optimal performance (amortized over multiple inputs). Specifically, it requires a one-time preprocessing stage that takes $O(|C|)$ time, where C is the smallest known Boolean circuit computing F . For each work instance, the client performs $O(|m|)$ work to prepare an m -bit input, the worker performs $O(|C|)$ work to compute the results, and the client performs $O(|n|)$ work to verify the n -bit result.

This result shows arbitrary computations can be outsourced to untrusted workers, preserve the secrecy of the data, and efficiently verify the computations were done correctly. Verifiable computing could thus be used to, say, outsource work to a cloud-based service or within a distributed computing project like Folding@home,¹⁹ which outsources protein-folding simulations to millions of Internet users worldwide. To prevent workers from cheating, these projects often assign the same work unit to multiple clients, then compare results; verifiable computing would eliminate these redundant computations and provide strong cryptographic protections against colluding workers, though some subtleties are involved when workers are caught cheating.^{6,21} The protocol is based on the crucial (and somewhat surprising) observation that Yao's Garbled Circuit protocol,³¹ in addition to providing secure two-party computation, provides a "one-time" verifiable computation. That is, Yao's Construction can be adapted to allow a client to outsource the computation of a function on a single input.

In the preprocessing stage of the outsourcing protocol, the client garbles the circuit C according to Yao's Construction. Then, in the "input preparation" stage, the client reveals the random labels associated with the input bits of x in the garbling. These labels allow the worker to compute the random labels associated with the output bits, and from them the client reconstructs $F(x)$. If the bit labels are sufficiently long and random, it is improbable that the worker can guess those for an incorrect output; therefore the client is assured the response, $F(x)$, is correct.

Unfortunately, reusing the circuit for a second input x' is insecure, since once the output labels of $F(x)$ are re-

vealed, nothing stops the worker from presenting the same labels as correct for $F(x')$. Creating a new garbled circuit requires as much work as if the client computed the function itself, so on its own, Yao's circuits do not provide an efficient method for outsourcing computation.

The second crucial idea is to combine Yao's Garbled Circuit with a fully homomorphic encryption system (such as Gentry's⁷) to be able to reuse the garbled circuit safely for multiple inputs. More specifically, instead of revealing the labels associated with the bits of input x , the client encrypts those labels under the public key of a fully homomorphic scheme. A new public key is generated for every input in order to prevent information from one execution from being useful for later executions. The worker then uses the homomorphic property to compute an encryption of the output labels and provide them to the client, which decrypts them and reconstructs $F(x)$.

Even without secure hardware, these results demonstrate a user's trust in one device can be leveraged to verify (and hence trust) the results of computations performed by any number of remote, untrusted commodity computers.

Conclusion

Motivated by the trend toward entrusting sensitive data and services to insecure computers, this article has described techniques to enable users to extend their trust in one device to other devices or services. Starting from trust in a simple USB device, users can verify the reports from secure hardware on their computer, then securely execute code on that computer, despite the presence of a mountain of untrustworthy code, while the untrusted code continues to enjoy the performance and features it does already. Efficiently extending the trust in this secure environment into the network improves the security of a range of network protocols. Finally, the user's trust is extended to the verification of computations performed by remote computers, even though they may be arbitrarily malicious. Many people using computers and online services today only imagine their data and privacy are protected. In the long run, the ability to

bootstrap trust in these computers and services will help replace this illusion with the actual foundation of security users expect and deserve.

Acknowledgments

This article is based on my Ph.D. dissertation,²¹ though much of the research it describes was conducted in collaboration with my advisor Adrian Perrig at Carnegie Mellon University, as well as with Hiroshi Isozaki, Jonathan McCune, Mike Reiter, and Arvind Seshadri;^{16–18} Zongwei Zhou;²⁴ and Craig Gentry and Rosario Gennaro.⁶ I am extremely grateful for their contributions. This article also benefited from feedback from Jay Lorch, Jonathan McCune, Diana Parno, and Adrian Perrig. C

References

- Advanced Micro Devices. *AMD64 Architecture Programmer's Manual*. AMD Publication No. 24593, Rev. 3.14, 2007; http://support.amd.com/us/Processor_TechDocs/24593_APM_v2.pdf
- Ames, Jr., S.R. Security kernels: A solution or a problem? In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, CA, Apr. 27–29). IEEE Computer Society, Los Alamitos, CA, 1981, 141.
- Anderson, D.P. BOINC: A system for public-resource computing and storage. In *Proceedings of the IEEE/ACM Workshop on Grid Computing* (Pittsburgh, PA, Nov. 8). IEEE Computer Society, Los Alamitos, CA, 2004, 4–10.
- Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Kotsovinos, E., Madhavapeddy, A., Neugebauer, R., Pratt, I., and Warfield, A. *Xen 2002*. Technical Report UCAM-CL-TR-553, University of Cambridge, Cambridge, U.K., Jan. 2003.
- Garriss, S., Cáceres, R., Berger, S., Sailer, R., van Doorn, L., and Zhang, X. Trustworthy and personalized computing on public kiosks. In *Proceedings of the Conference on Mobile Systems, Applications, and Services* (Breckenridge, CO, June 17–20). ACM Press, New York, 2008, 199–210.
- Gennaro, R., Gentry, C., and Parno, B. Non-interactive verifiable computation: Outsourcing computation to untrusted workers. In *Advances in Cryptology: Proceedings of the 30th International Cryptology Conference* (Santa Barbara, CA, Aug. 15–19, 2010), 465–482.
- Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st ACM Symposium on Theory of Computing* (Bethesda, MD, May 31–June 2). ACM Press, New York, 2009, 169–178.
- Gold, B.D., Linde, R.R., and Cudney, P.F. KVM/370 in retrospect. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, CA, Apr. 29–May 2). IEEE Computer Society, Los Alamitos, CA, 1984, 13–23.
- Hao, S., Syed, N.A., Feamster, N., Gray, A.G., and Krasser, S. Detecting spammers with SNARE: Spatio-temporal network-level automatic reputation engine. In *Proceedings of the USENIX Security Symposium* (Montréal, Aug. 10–14). USENIX Association, Berkeley, CA 2009, 101–118.
- Intel Corp. *Intel Trusted Execution Technology: Measured Launched Environment Developer's Guide*. Document no. 315168-005, Santa Clara, CA, June 2008.
- Karger, P.A., Zurko, M.E., Bonin, D.W., Mason, A.H., and Kahn, C.E. A retrospective on the VAX VMM security kernel. *IEEE Transactions on Software Engineering* 17, 11 (Nov. 1991), 1147–1165.
- Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Norrish, M., Kolanski, R., Sewell, T., Tsch, H., and Winwood, S. seL4: Formal verification of an OS kernel. In *Proceedings of the ACM Symposium on Operating Systems Principles* (Big Sky, MT, Oct. 11–14). ACM Press, New York, 2009, 207–220.
- Magenheimer, D. *Xen/IA64 Code Size Stats*. Xen developer's mailing list 2005; <http://lists.xensource.com/>
- McCune, J.M. *Reducing the Trusted Computing Base for Applications on Commodity Systems*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, Jan. 2009.
- McCune, J.M., Berger, S., Cáceres, R., Jaeger, T., and Sailer, R. Shamon: A system for distributed mandatory access control. In *Proceedings of the Annual Computer Security Applications Conference* (Miami Beach, Dec. 11–15). IEEE Computer Society, Los Alamitos, CA, 2006, 23–32.
- McCune, J.M., Parno, B., Perrig, A., Reiter, M.K., and Isozaki, H. Flicker: An execution infrastructure for TCB minimization. In *Proceedings of ACM EuroSys* (Glasgow, Scotland, Mar. 31–Apr. 4). ACM Press, New York, 2008, 315–328.
- McCune, J.M., Parno, B., Perrig, A., Reiter, M.K., and Seshadri, A. How low can you go? Recommendations for hardware-supported minimal TCB code execution. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Seattle, Mar. 1–5). ACM Press, New York, 2008, 14–25.
- McCune, J.M., Parno, B., Perrig, A., Reiter, M.K., and Seshadri, A. Minimal TCB code execution (extended abstract). In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, CA, May 20–23). IEEE Computer Society, Los Alamitos, CA, 2007, 267–272.
- Pande Lab. The folding@home project. Stanford University; <http://folding.stanford.edu/>
- Parno, B. Bootstrapping trust in a 'trusted' platform. In *Proceedings of the USENIX Workshop on Hot Topics in Security* (San Jose, CA, July 29). USENIX Association, Berkeley, CA, 2008, 9:1–9:6.
- Parno, B. *Trust Extension as a Mechanism for Secure Code Execution on Commodity Computers*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, May 2010.
- Parno, B., Lorch, J.R., Douceur, J.R., Mickens, J., and McCune, J.M. Memoir: Practical state continuity for protected modules. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, CA, May 22–25). IEEE Computer Society, Los Alamitos, CA, 2011, 379–394.
- Parno, B., McCune, J.M., and Perrig, A. *Bootstrapping Trust in Modern Computers*. Springer, New York, 2011.
- Parno, B., Zhou, Z., and Perrig, A. *Help Me Help You: Using Trustworthy Host-Based Information in the Network*. Technical Report CMU-CyLab-09-016, Carnegie Mellon University, CyLab, Pittsburgh, PA, Nov. 2009.
- Sailer, R., Zhang, X., Jaeger, T., and van Doorn, L. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the USENIX Security Symposium* (San Diego, CA, Aug. 9–13). USENIX Association, Berkeley, CA, 2004, 16–32.
- Singaravelu, L., Pu, C., Haertig, H., and Helmuth, C. Reducing TCB complexity for security-sensitive applications: Three case studies. In *Proceedings of ACM EuroSys* (Leuven, Belgium, Apr. 18–21). ACM Press, New York, 2006, 161–174.
- Smith, S.W. and Weingart, S. Building a high-performance, programmable secure coprocessor. *Computer Networks* 31, 8 (Apr. 1999), 831–860.
- Trusted Computing Group. *Trusted Platform Module Main Specification*. Version 1.2, Revision 103, 2007; http://www.trustedcomputinggroup.org/resources/tpm_main_specification
- Vasudevan, A., Parno, B., Qu, N., Gligor, V.D., and Perrig, A. *Lockdown: A Safe and Practical Environment for Security Applications*. Technical Report CMU-CyLab-09-011, Carnegie Mellon University, CyLab, Pittsburgh, PA, July 2009.
- Wave Systems Corp. *Trusted Computing: An Already Deployed, Cost-Effective, ISO Standard, Highly Secure Solution for Improving Cybersecurity*, 2010; http://www.nist.gov/itl/upload/Wave-Systems_Cybersecurity-NOI-Comments_9-13-10.pdf
- Yao, A. Protocols for secure computations. In *Proceedings of the IEEE Symposium on Foundations of Computer Science* (Chicago, Nov. 3–5). IEEE Computer Society, Los Alamitos, CA, 1982, 160–164.

Bryan Parno (parno@microsoft.com) is a researcher in Microsoft Research, Redmond, WA.

© 2012 ACM 0001-0782/12/06 \$10.00

Exploring the technology trends in basic, integrated, and cloud data services.

BY MICHAEL J. CAREY, NICOLA ONOSE,
AND MICHALIS PETROPOULOS

Data Services

TODAY'S BUSINESS PRACTICES require access to enterprise data by both external as well as internal applications. Suppliers expose part of their inventory to retailers, and health providers allow patients to view their health records. But how do enterprise data owners ensure access to their data is appropriately restricted and has predictable impact on their infrastructure? In turn, application developers sitting on the “wrong side” of the Web from their data need a mechanism to find out which data they can access, what their semantics are, and how they can integrate data from multiple enterprises. Data services are software components that address these issues by providing rich metadata, expressive languages, and APIs for service consumers to use to send queries and receive data from service providers.

Data services are a specialization of Web services that can be deployed on top of data stores, other services, and/or applications to encapsulate a wide

range of data-centric operations. In contrast to traditional Web services, services that provide access to data need to be *model-driven*, offering a semantically richer view of their underlying data and enabling advanced querying functionality. Data service consumers need access to enhanced metadata, which is both machine-readable and human-readable, such as schema information. This metadata is needed to use or integrate entities returned by different data services. For example, a `getCustomers` data service might retrieve customer entities, while `getOrdersByCID` retrieves orders given a customer id. In the absence of any schema information, the consumer is not sure if he or she can compose these two services to retrieve the orders of customers. Moreover, consumers of data services can utilize a query language and pass a query expression as a parameter to a data service; they can use metadata-guided knowledge of data services and their relationships to formulate queries and navigate between sets of entities.

Modern data services are descendants of the stored procedure facilities provided by relational database systems, which allow a set of SQL statements and control logic to be parameterized, named, access-controlled, and then called from applications that

» key insights

- Data services differ from traditional Web services in that they serve as “fronts” for data and are based on a richer model of that data.
- The growing importance of data services in the movement toward a hosted-services world is evidenced by the number of contexts within which they have been utilized in recent years: data publishing, data exchange and integration, service-oriented architectures (SOA), data as a service (DaaS), and most recently, cloud computing.
- While data services were initially conceived to solve problems in the enterprise world, the cloud is now making data services accessible to a much broader range of consumers.



ILLUSTRATION BY CELIA JOHNSON

wish to have the procedure perform its task(s) without their having to know the internal details of the procedure.⁴² The use of data services has many of the same access control and encapsulation motivations. Data owners (database administrators) publish data services because they are able to address security concerns and protect sensitive data, predict the impact of the published data services on their resources, and safeguard and optimize the performance of their servers. In traditional IT settings, it is not uncommon for stored procedures to be the only permitted data access method for these reasons,

as letting applications submit arbitrary queries has long been dismissed as too permissive and unpredictable.¹² This is even more of an issue in the world of data services, where providers of data services typically have less knowledge of (and much less control over) their client applications.

We are now moving toward a hosted services world and new ways of managing and exchanging data. The growing importance of data services in this movement is evidenced by the number of contexts within which they have been utilized in recent years: data publishing, data exchange and inte-

gration, service-oriented architectures (SOA), data as a service (DaaS), and most recently, cloud computing. Lack of standards, though, has allowed different middleware vendors to develop different models for representing and querying their data services.¹⁴ Consequently, federating databases and integrating data in general may become an even bigger headache. This picture becomes still more complicated if we consider databases living in the cloud.³ Integrating data across the cloud will be all the more daunting because the models and interfaces there are even more diverse, and also because com-

mon assumptions about data locality, consistency, and availability may no longer hold.

The purpose of this article is three-fold: to explain basic data services architecture(s) and terminology; to present three popular contexts in which data services are being deployed today; along with one exemplary commercial framework targeting each context; and to discuss certain advanced issues, such as transactions and con-

sistency, pertaining to data services. We use a very simple “enterprise-y” e-commerce scenario (involving customer and order data) to illustrate the different data service styles, which will include basic, integrated, and cloud data services. We chose a simple e-commerce scenario because it is intuitive and some variant of it makes sense for each of the styles. We conclude by briefly examining emerging trends and exploring possible directions for future

data services research and development. Although we do include product-based examples, it is not our intention to recommend products or to survey the product space; the purpose of our examples is concreteness (and reality).

Data Services Architecture

A data service can be employed on top of data stores providing different interfaces, such as database servers, CRM applications, or cloud-based storage systems, and using diverse underlying data models, such as relational, XML, or simple key/value pairs. In each case, however, the data and metadata of data services are exposed to their consumers in terms of a common *external model*. Figure 1 presents the architecture of a data service, where the mapping between the model of the data store and the external model of the data service is shown. Users building data services can define the external model and express the mapping, however declarative or procedural it may be, either manually or by using utilities provided by the data service framework to automatically generate the external model and mapping for certain classes of data stores. In the declarative case, these mappings are often similar in many respects to view definitions in relational databases.⁴² Note that data services can encapsulate read access and/or update access to the underlying data; we focus largely on read access here for ease of exposition.

Figure 1 also highlights the two prevailing methods for consuming data services: either through *functions* or through *queries*. Functions encapsulate the data and make it accessible only through a set of carefully defined, and possibly application-specific, function signatures. In contrast, queries can be formulated based on the external model using a (possibly restricted) query language. Functions capture the current practice of exporting parameterized stored procedures as data services. When clients utilize queries as consuming methods, various query languages can be used, among them SQL, XPath/XQuery, or proprietary languages such as SOQL from Salesforce.com⁴³ or Microsoft’s OData queries.³⁷ The mapping between the query language of the data store (if any) and the query language utilized by service cli-

Figure 1. Data service architecture.

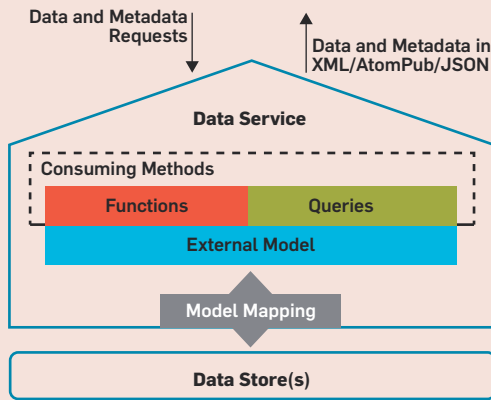
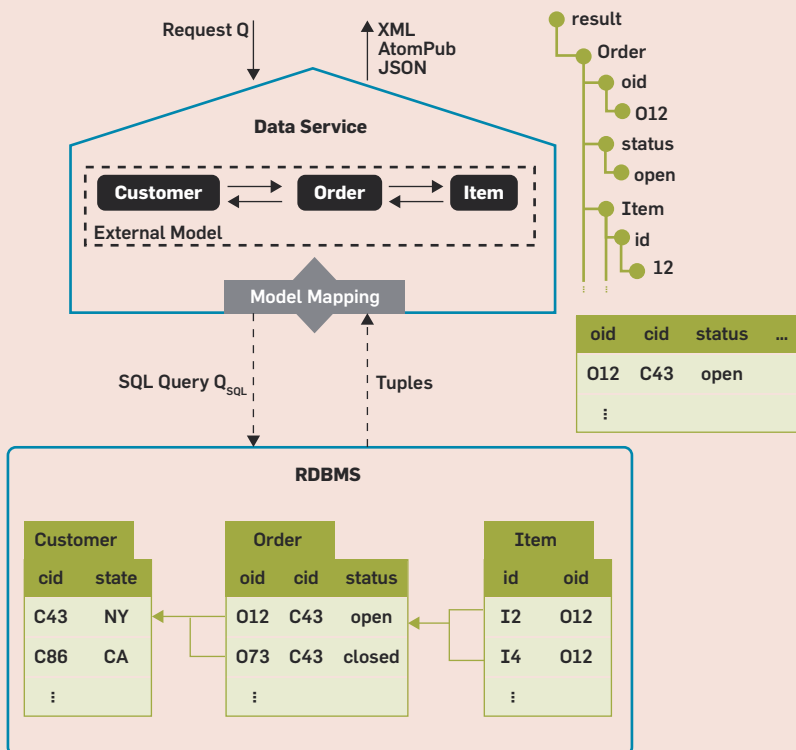


Figure 2. Service-enabling a relational data store.



ents is performed during each service request based on the model mapping of the data service. The information used to guide this mapping is provided at design time by the data service implementer. The syntax in which data service output is returned to clients varies from JSON³¹ to XML to Atom-Pub,²⁸ as shown in Figure 1.

A last point that needs clarification is how the external model is made available to the clients of data services. Apart from their regular data-returning functions and queries, data service frameworks generally provide a set of special functions and queries that return a description of the external model in order to guide clients in consuming a service's interconnected functions and/or queries in a meaningful way.

In contrast, traditional Web services have a much simpler architecture: they are purely functional and their implementations are opaque. Some traditional Web services rely on WSDL¹⁹ to describe the provided operations, where the input messages accepted and the output messages expected are defined, but these operations are semantically disconnected from the client's point of view, with each one providing independent functionality. Therefore, the presence of a data model is distinctive to data services.

Service-Enabling Data Stores

In the basic data service usage scenario, the owners of a data store enable Web clients and other applications to access their otherwise externally inaccessible data by publishing a set of data services, thus *service-enabling* their data store. Microsoft's WCF Data Services framework³⁵ and Oracle's ODSI³⁸ are two of a number of commercial frameworks that can be used to achieve this goal. (A survey of such frameworks is beyond the scope of this article.)

To illustrate the key concepts involved in service-enabling a data store, we will use the WCF Data Services framework as an example. We show how the framework can be used to service-enable a relational data store (although in principle any data store can be used.) Figure 2 shows a relational database that stores information about customers, the orders they have placed, and the items comprising these orders. The owners map this internal relation-



Data service frameworks generally provide a set of special functions and queries that return a description of the external model in order to guide clients in consuming a service's interconnected functions and/or queries in a meaningful way.



al schema to an external model that includes entity types Customer, Order and Item. Foreign key constraints are mapped to navigation properties of these entity types, as shown by the arrows in the external model of Figure 2. Hence, given an Order entity, its Items and its Customer can be accessed.

WCF Data Services implements the Open Data Protocol (OData),³⁷ which is an effort to standardize the creation of data services. OData uses the Entity Data Model (EDM) as the external model, which is an object-extended E/R model with strong typing, identity, inheritance and support for scalar and complex properties. EDM is part of Microsoft's Entity Framework, an object-relational mapping (ORM) framework,² which provides a declarative mapping language and utilities to generate EDM models and mappings given a relational database schema.

Clients consuming OData services, such as those produced by the WCF Data Services framework, can retrieve a Service Metadata Document³⁷ describing the underlying EDM model by issuing the following request that will return the entity types and their relationships:

```
http://<service_uri>/SalesDS.svc/
$metadata
```

OData services provide a uniform, URI-based querying interface and map CRUD (Create, Retrieve, Update, and Delete) operations to standard HTTP verbs.²² Specifically, to create, read, update or delete an entity, the client needs to submit an HTTP POST, GET, PUT or DELETE request, respectively. For example, the following HTTP GET request will retrieve the *open* Orders and Items for the Customer with cid equal to C43:

```
http://<service_uri>/SalesDS.svc/
Customers('C43')/Orders?
$filter=status eq 'open'&
$expand=Items
```

The second line of the above request selects the Customer entity with key C43 and navigates to its Order entities. The \$filter construct on the third line selects only the "open" Order entities, and the \$expand on the

Service-Enabling Data Sources

Here, we provide additional information related to the WCF Data Services example in order to make the example more concrete. Figure A shows a graphical representation of the EDM model for the relational database in Figure 2, as displayed to a WCF Data Service developer, along with the declarative mapping details for the *Order* entity at the bottom. The navigation properties are displayed at the bottom of each entity type, and the connecting lines denote the property associations between the entity types that facilitate navigation.

Clients consuming WCF Data Services can retrieve metadata describing the underlying EDM model by issuing a request to return a service's entity types and their relationships, for example,

```
http://<service_uri>/SalesDS.svc/$metadata
```

Metadata is returned in the Conceptual Schema Definition Language (CSDL), an example of which is shown in Figure B.

In terms of data service results in WCF, Figure C shows the results returned for the example query in the section entitled “Service-Enabling Data Stores,” which requested a list of the open orders for Customer C43. The tuples retrieved by the underlying SQL query are subsequently transformed to the AtomPub result format used by OData services, as shown in the figure. Note that the query result contains link elements that allow the client to form navigation requests to related entities.

Figure A. An entity data model.

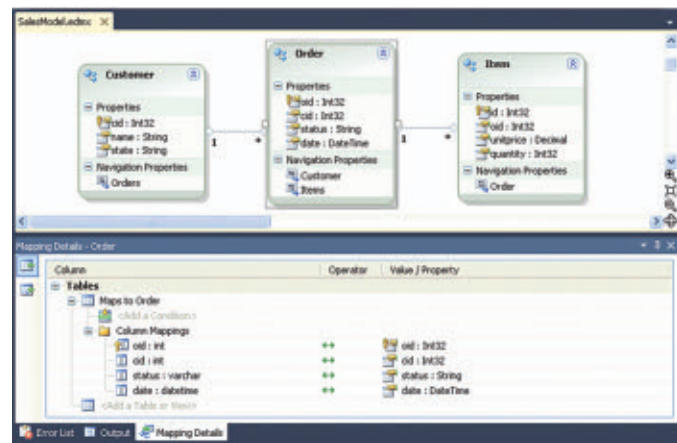


Figure B. An OData service metadata document.



Figure C. An OData service response document.



last line navigates and inlines the related Item entities within each Order. Other query constructs, such as \$select, \$orderby, \$top and \$skip, are also supported as part of a request.

To provide interested readers with a bit more detail, a graphical representation of the external model (EDM) together with examples of the OData data service metadata and an AtomPub response from an OData service can be found in the sidebar “Service-Enabling Data Sources.”

Driven by the mapping between the EDM model and the underlying data store’s relational model, the WCF Data Services framework automatically translates the above request into the query language of the store. In our example then, under the covers, the above request would be translated into the following SQL query involving a left outer join:

```
SELECT Order.*, Item.*
FROM Customer JOIN Order LEFT OUTER JOIN Item
WHERE Customer.cid='C43'
AND Order.status='open'
```

The WCF Data Services framework can also publish stored procedures as part of an OData service, called *Function Imports*, which can be used for any of the CRUD operations and are integrated in the same URI-based query interface.

Integrated Data Services

In the second data service usage scenario, data services are created to integrate as well as to service-enable a *collection of data sources*. The underlying sources are often heterogeneous in nature. The result of integration is that consumers of a data service see what appears to be one coherent, service-enabled data source rather than being faced with a hodgepodge of disparate schemas and data access APIs.

Figure 3 shows an example usage scenario. We will illustrate this use case using a commercial data services middleware platform, the AquaLogic Data Services Platform, ALDSP,¹³ developed at BEA Systems and later rebranded as ODSI, the Oracle Data Services Integrator.³⁸ ODSI is based on a functional model⁴⁴ and the functional data service consuming method of Figure 1. (Again, we use ODSI as just one example of the

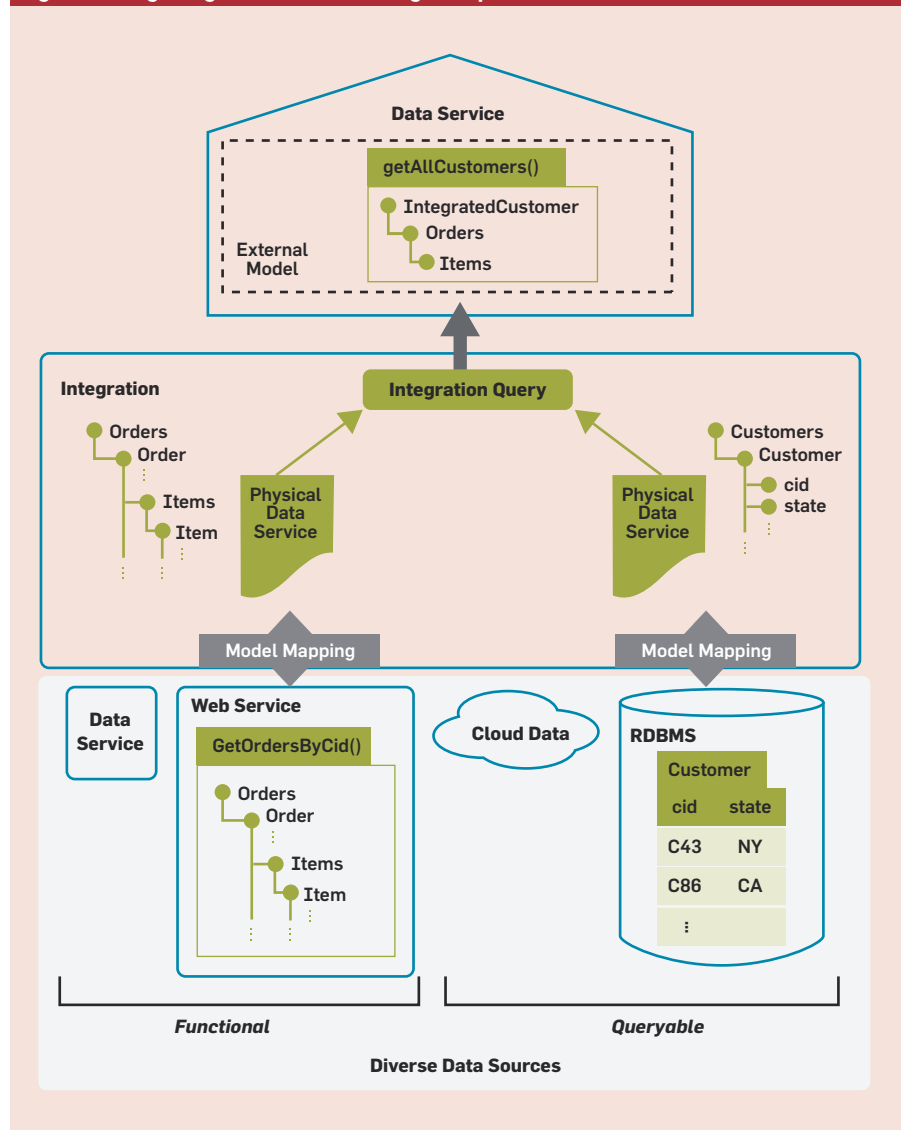
products in this space.)

At the bottom of Figure 3 we see a variety of data sources. These can include queryable data sources, such as relational databases and perhaps data stored and managed in the cloud, as well as functional data sources like traditional Web services and other preexisting data services. For each type of data source, the data services platform provides a default mapping that gives the data service architect a view of that data source in terms of a common data and programming model. In ODSI, what the data service architect sees initially is a set of *physical data services* that are modeled as groups of functions that each take zero or more XML arguments and return XML results. For instance, a relational table is modeled as a data ser-

vice that has read, create, update, and delete functions; the read function returns a stream of XML elements, one per row from the underlying table, if used in a query without additional filter predicates. A Web service-based data source is modeled as a function whose inputs and outputs correspond to the schema information found in the service’s WSDL description.

For a given set of physical data services, the task of the data service architect is then to construct an integrated model for consumption by client application developers, SOA applications, and/or end users. In the example shown in Figure 3, data from two different sources, a Web service and a relational database, are being combined to yield a *single view of customer* data service that hides its integration

Figure 3. Integrating and service-enabling enterprise data.



details from its consumers. In ODSI, the required merging and mapping of lower-level data services to create higher-level services is achieved by function composition using the XQuery language.⁹ (This is very much like defining and layering views in a relational DBMS setting.) Because data service architects often prefer graphical tools to handwriting queries, ODSI provides a graphical query editor to support this activity; the sidebar “Integrated Data Services” describes its usage for this use case in a bit more detail for the interested reader.

Stepping back, Figure 3 depicts a *declarative* approach to building data services that integrate and service-enable disparate data sources. The external model is a functional model based on XQuery functions. The approach is declarative because the integration logic is specified in a high-level language—the integration query is written in XQuery in the case of ODSI. Because of this approach, suppose the resulting function is subsequently called from a query such as the following, which could either come from an application or from another data service defined on top of this one:

```
for $cust in ics:getAllCustomers( )
where $cust/State = 'Rhode Island'
return $cust/Name
```

In this case, the data services platform can *see through* the function definition and optimize the query’s execu-

tion by fetching only Rhode Island customers from the relational data source and retrieving only the orders for those customers from the order management service to compute the answer. This would not be possible if the integration logic was instead encoded in a procedural language like Java or a business process language like BPEL.³² Moreover, notice that the query does not request all data for customers; instead, it only asks for their names. Because of this, another optimization is possible: The engine can answer the query by fetching only the names of the Rhode Island customers from the relational source and altogether avoid any order management system calls. Again, this optimization is possible because the data integration logic has been declaratively specified. Finally, it is important to note that such function definitions and query optimizations can be composed and later decomposed (respectively) through arbitrary layers of specifications. This is attractive because it makes an incremental (piecewise) data integration methodology possible, as well as allowing for the creation of specialized data services for different consumers, without implying runtime penalties due to such layering when actually answering queries.

In addition to exemplifying the integration side of data services, ODSI includes some interesting advanced data service modeling features. ODSI supports the creation and publishing

of collections of interrelated data services (a.k.a. dataspace).¹³ Metadata about collections of published data services is made available in ODSI through catalog data services. Also, in addition to supporting method calls from clients, ODSI provides an optional generic query interface that authorized clients can use to submit ad hoc queries that range over the results of calls to one or more data service methods. Thus, ODSI offers a hybrid of the two data service consuming methods discussed earlier.

To aid application developers, methods in ODSI are classified by purpose. ODSI supports read, create, update, and delete methods for operating on data instances. It also supports relationship methods to navigate from one object instance (for example, a customer) to related object instances (for example, complaints). ODSI methods are characterized as being either functions (side-effect free) or procedures (potentially side-effecting). Finally, a given method’s visibility can be designated as being one of: accessible by outside applications, usable only from within other data services, or private to one particular data service.

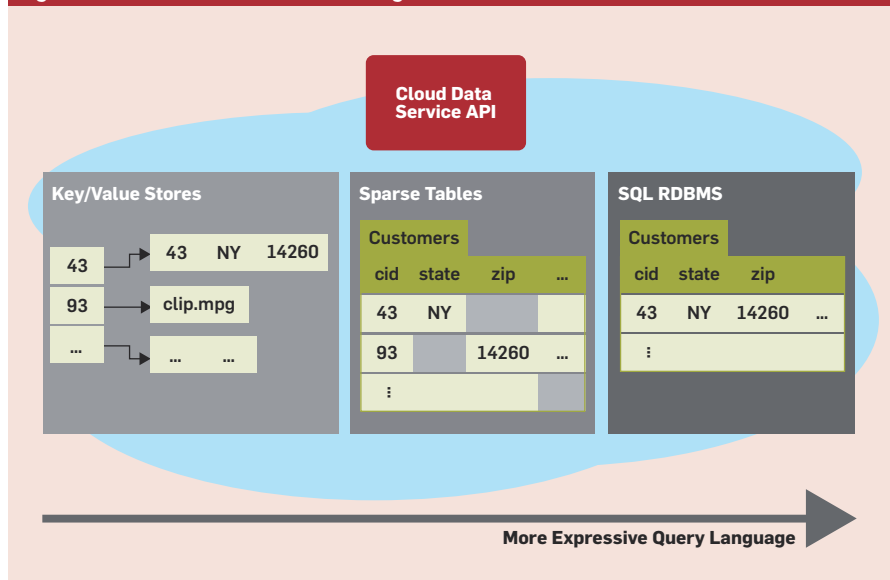
Cloud Data Services

We have described how an enterprise data source or an integrated set of data sources can be made available as services. Here, we focus on a new class of data services designed for providing data management in the cloud.^a

The cloud is quickly becoming a new universal platform for data storage and management. By storing data in the cloud, application developers can enjoy a pay-as-you-go charging model and delegate most administrative tasks to the cloud infrastructure, which in turn guarantees availability and near-infinite scalability. As depicted in Figure 4, cloud data services today offer various external data models and consuming methods, ranging from *key-value stores* to *sparse tables* all the way to *RDBMSs in the cloud*. In terms of consuming methods (see Figure 1), key-value stores offer a simple function-based interface, while sparse tables are accessed via queries.

^a These service descriptions were current as of mid-2011.

Figure 4. Data services and cloud storage.



RDBMSs allow both a functional interface, if data access is provided via stored procedures only, and a query-based interface, if the database may be queried directly. A recent and detailed survey on cloud storage technologies¹⁶ proposes a similar classification of cloud data services, but further differentiates sparse tables into *document stores* and *extensible record stores*. With respect to the architecture in Figure 1, some of these services forgo the model mapping layer, choosing instead to directly expose their underlying model to consuming applications.

To illustrate the various types of cloud data services, we will briefly examine the Amazon Web Services (AWS)¹ platform, as it has arguably pioneered the cloud data management effort. Other IT companies are also switching to building cloud data management frameworks, either for internal applications (for example, Yahoo!'s PNUTS²⁰) or to offer as publicly available data services (for example, Microsoft's WCF data services, as made available in Windows Azure³⁴).

Key-value stores: The simplest kind of data storage services is key-value stores that offer atomic CRUD operations for manipulating serialized data structures (objects, files, among others) that are identifiable by a key.

An example of a key-value store is Amazon S3¹ that provides storage support for variable size data blocks, called objects, uniquely identified by (developer assigned) keys. Data blocks reside in *buckets*, which can list their content and are also the unit of access control. Buckets are treated as subdomains of `s3.amazonaws.com`. (For instance, the object `customer01.dat` in the bucket `custorder` can be accessed as `http://custorder.s3.amazonaws.com/customer01.dat`.)

The most common operations in S3 are:

- ▶ create (and name) a bucket,
- ▶ write an object, by specifying its key, and optionally an access control list for that object,
- ▶ read an object,
- ▶ delete an object, and,
- ▶ list the keys contained in one of the buckets.

Data blocks in S3 were designed to hold large objects (multimedia files), but they can potentially be used as da-

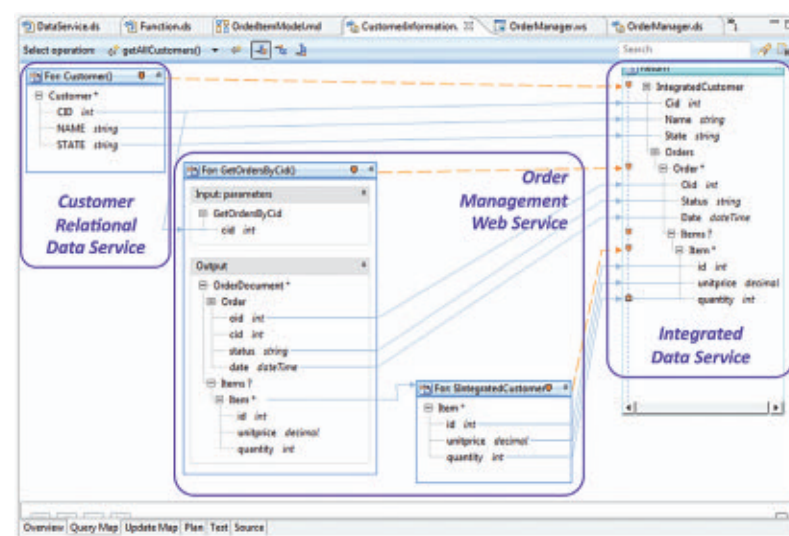
Integrated Data Services

Here, we dive a bit deeper into the ODSI example to more fully convey ODSI's approach to data service integration. In particular, the accompanying figure presents a screen shot that shows how the ODSI graphical query editor would be used in the example scenario when the customer data source is a relational customer table and the orders and line items are coming from an order management Web service that takes a customer id as input and returns the orders for that customer as output.

Let us take a closer look at the figure. The upper left box in the query editor screen shot represents the Customer physical data service, which returns elements containing the CID, NAME, and STATE information for customers. The middle box represents the Web service call `GetOrdersByCid()`, which takes an XML input containing the cid of a customer and returns an XML result containing the orders (including the nested line items) for this customer. The artifact being constructed is itself an XQuery query that will be the body of a callable data service function—a function called `getAllCustomers()`—and the box on the far right of the screen shot represents the XML result structure for the function. The rest of the screen shot is a graphical specification of how the data from the two physical data services is to be mapped and combined. The resulting `getAllCustomers()` data service function can then be made available as a callable Web service by simply instructing ODSI to do so through a few additional mouse clicks.

More information about ODSI's approach to data integration and its data service authoring tools may be found in Carey et al.¹³ and Oracle.³⁸

Integrating RDBMS and Web service data sources.



tabase pages storing (several) records. Brantner et al.¹¹ started from this observation and analyzed the trade-offs in building a database system over S3. However, recent industrial trends are favoring the incorporation of more DBMS functionality directly into the cloud infrastructure, thus offering a higher-level interface to the application.

Dynamo²¹ is another well-known Amazon example of a key-value store. It differs in granularity from S3 since it stores only objects of a relatively small size (< 1MB), whereas data blocks in S3 may go up to 5GB.

Sparse tables are a new paradigm of storage management for structured and semi-structured data that has emerged in recent years, especially after the interest generated by Google's Bigtable.¹⁸ (Bigtable is the storage system behind many of Google's applications and is exposed, via APIs, to Google App Engine²⁵ developers.) A sparse table is a collection of data records, each one having a row and a set of column identifiers, so that at the logical level records behave like the rows of a table. There may be little or no overlap between the columns used

in different rows, hence the “sparsity.” The ability to address data based on (potentially many) columns differentiates sparse tables from key/value stores and makes it possible to index and query data more meaningfully. Compared to traditional RDBMSs that require static (fixed) schemas, sparse tables have a more flexible data model, since the set of columns identifiers may change based on data updates. Bigtable has inspired the creation of similar open source systems such as HBase²⁹ and Cassandra.¹⁵

SimpleDB¹ is Amazon’s version of a sparse table and exposes a Web service interface for basic indexing and querying in the cloud. A column value in a SimpleDB table may be atomic, as in the relational model, or a list of atomic values (limited in size to 1KB). SimpleDB’s tables are called domains. SimpleDB queries have a SQL-like syntax and can perform selections, projections and sorting over domains. There is no support for joins or nested subqueries.

A SimpleDB application stores its customer information in a domain called Customers and its order information in an Orders domain. Using SimpleDB’s REST^b interface, the application can insert records (`id='C043', state='NY'`) into Customers and (`id='O012', cid='043', status='open'`) into Orders. Further inserts do not necessarily need to conform to these schemas, but for the sake of our example we will assume they do.

Since SimpleDB does not implement joins, joins must be coded at the client application level. For example, to retrieve the orders for all NY clients, an application would first fetch the client info via the query:

```
select id from Customers where
state='NY'
```

the result of which would include C043 and would then retrieve the corresponding orders as follows:

```
select * from Orders where cid= 'C043'
```

A major limitation for SimpleDB is that the size of a table instance is bounded. An application that manipulates a large

volume of data needs to manually partition (“shard”) it and issue separate queries against each of the partitions.

RDBMSs: In cloud computing systems that provide a virtual machine interface, such as EC2,¹ users can install an entire database system in the cloud. However, there is also a push toward providing a database management system itself as a service. In that case, administrative tasks such as installing and updating DBMS software and performing backups are delegated to the cloud service provider.

Amazon RDS¹ is a cloud data service that provides access to the full capabilities of a MySQL³⁹ database installed on a machine in the cloud, with the possibility of setting several “read replicas” for read-intensive workloads. Users can create new databases from scratch or migrate their existent MySQL data into the Amazon cloud. Microsoft has a similar offering with SQL Azure,⁷ but chooses a different strategy that supports scaling by physically partitioning and replicating logical database instances on several machines. A SQL Azure source can be service-enabled by publishing an OData service on top of it, as in the section “Service-Enabling Data Stores.” Google’s Megastore⁵ is also designed to provide scalable and reliable storage for cloud applications, while allowing users to model their data in a SQL-like schema language. Data types can be string, numeric types, or Protocol Buffers²⁶ and they can be required, optional or repeated.

Amazon RDS users manage and interact with their databases either via shell scripts or a SOAP^c-based Web services API. In both cases, in order to connect to a MySQL instance, users need to know its DNS name, which is a sub-domain of `rds.amazonaws.com`. They can then either open a MySQL console using an Amazon-provided shell script, or they can access the database like any MySQL instance identified by a DNS name and port.

Advanced Technical Issues

So far we have mostly covered the basics of data services, touching on a range of use cases (single source, integrated source, and cloud sources) along with their associated data ser-

vice technologies. Here, we will briefly highlight a few more advanced topics and issues, including updates and transactions, data consistency for scalable services, and issues related to security for data services.

Data service updates and transactions. As with other applications, applications built over data services require transactional properties in order to operate correctly in the presence of concurrent operations, exceptions, and service failures. Data services based on single sources, for the most part, can inherit their answer to this requirement from the source that they serve their data from. Data services that integrate data from multiple sources, however, face additional challenges—especially since many interesting data sources, such as enterprise Web services and cloud data services, are either unable or “unwilling” to participate in traditional (two-phase commit-based) distributed transactions due to issues related to high latencies and/or temporary loss of autonomy. Data service update operations that involve non-transactional sources can potentially be supported using a compensation-based transaction model⁸ based on Sagas.²³ The classic compensating transaction example is travel-related, where a booking transaction might need to perform updates against multiple autonomous ticketing services (to obtain airline, hotel, rental car, and concert reservations) and roll them all back via compensation in the event that reservations cannot be obtained from all of them. Unfortunately, such support is underdeveloped in current data service offerings, so this is an area where all current systems fall short and further refinement is required. The current state of the art leaves too much to the application developer in terms of hand-coding compensation logic as well as picking up the pieces after non-atomic failures.

Another challenge, faced both by single-source and multisource data services, is the mapping of updates made to the external model to correspondingly required updates to the underlying data source(s). This challenge arises because data services that involve non-trivial mappings—as might be built using the tools provided by WCF or ODSI—present the service

b Representational State Transfer.²²

c Simple Object Access Protocol.³⁶

consumer with an interface that differs from those of the underlying sources. The data service may restructure the format of the data, it may restrict what parts of the data are visible to users, and/or it may integrate data coming from several back-end data sources. Propagating data service updates to the appropriate source(s) can be handled for some of the common cases by analyzing the lineage of the published data, that is, computing the inverse mapping from the service view back to the underlying data sources based on the service view definition.^{2,8} In some cases this is not possible, either due to issues similar to non-updatability of relational views^{6,33} or due to the presence of opaque functional data sources such as Web service calls, in which case hints or manual coding would be required for a data services platform to know how to back-map any relevant data changes.

Data consistency in the cloud. To provide scalability and availability guarantees when running over large clusters, cloud data services have generally adopted lower consistency models. This choice is defended in Helland et al.,³⁰ which states that in large-scale distributed applications the scope of an atomic update needs to be—and generally is—within an abstraction called *entity*. An entity is a collection of data with a unique key that lives on one machine, such as a data record in Dynamo. According to Helland et al., developers of truly scalable applications have no real choice but to cope with the lack of transactional guarantees across machines and with repeated messages sent between entities. In practice, there are several consistency models that share this philosophy.

The simplest model is *eventual consistency*,⁴⁶ first defined in Terry et al.⁴⁵ and used in Amazon Dynamo,²¹ which only guarantees that all updates will reach all replicas eventually. There are some challenges with this approach, mainly because Dynamo uses replication to increase availability. In the case of network or server failures, concurrent updates may lead to update conflicts. To allow for more Dynamo pushes conflict resolution to the application. Other systems try to simplify developers' lives and resolve conflicts inside the data store, based on simple

Applications built over data services require transactional properties in order to operate correctly in the presence of concurrent operations, exceptions, and service failures.

policies: for example, in S3 the write with the latest timestamp wins.

PNUTS, Yahoo's sparse table store, goes one step beyond eventual consistency by also guaranteeing that all replicas of a given record apply all updates in the same order. Such stronger guarantees, called *timeline consistency*, may be necessary for applications that manipulate user data, for example, if a user changes access control rights for a published data collection and then adds sensitive information. Amazon SimpleDB has recently added support for similar guarantees, which it calls *consistent reads*, as well as for *conditional updates*, which are executed only if the current value of an attribute of a record has a specified expected value. Conditional update primitives make it easier to implement solutions for common use cases such as global counters or optimistic concurrency control (by maintaining a timestamp attribute).

Finally, RDBMSs in the cloud (Megastore, SQL Azure) provide ACID semantics under the restriction that a transaction may touch only one entity. This is ensured by requiring all tables involved in a transaction to share the same partitioning key. In addition, Megastore provides support for transactional messaging between entities via queues and for explicit two-phase commit.

Data services security. A key aspect of data services that is underdeveloped in current product and service offerings, yet extremely important, is data security. Web service security alone is not sufficient, as control over who can invoke which service calls is just one aspect of the problem for data services. Given a collection of data services, and the data over which they are built, a data service architect needs to be able to define access control policies that govern which users can do and/or see what and from which data services. As an example, an interesting aspect of ODSI is support for fine-grained control over the information disclosed by data service calls—where the same service call, depending on “who's asking,” can return more or less information to the caller. Portions of the information returned by a data service call can be encrypted, substituted, or altogether elided (schema permitting) from the call's results.¹⁰ More broadly, much work has been done in the areas of access control, security, and

privacy for databases, and much of it applies to data services. These topics are simply too large⁴ to cover in the scope of this article.

Emerging Trends

In this article we have taken a broad look at work in the area of data services. We looked first at the enterprise, where we saw how data services can provide a data-oriented encapsulation of data as services in enterprise IT settings. We examined concepts, issues, and example products related to service-enabling single data sources as well as related to the creation of services that provide an integrated, service-oriented view of data drawn from multiple enterprise data sources. Given that clouds are rapidly forming on the IT horizon, both for Web companies and for traditional enterprises, we also looked at the emerging classes of data services that are being offered for data management in the cloud. As the latter mature, we expect to see a convergence of everything that we have looked at, as it seems likely that rich data services of the future will often be fronting data residing in one or more data sources in the cloud.

To wrap up, we briefly list a handful of emerging trends that can possibly direct future data services research and development. Some of the trends listed stem from existing problems, while others are more predictive in nature. We chose this list, which is necessarily incomplete, based on the evolution of data services we have witnessed while slowly authoring this report over the two last years. Again, while data services were initially conceived to solve problems in the enterprise world, the cloud is now making data services accessible to a much broader range of consumers; new issues will surely arise as a result.

Query formulation tools. Service-enabled data sources sometimes support (or permit) only a restricted set of queries against their schemas. Users trying to formulate a query over multiple such sources can have difficulty determining how to compose such data services. For a schema-based external model, recent work proposed tools to help users author only answerable queries, for example, CLIDE.⁴¹ These tools utilize the schemas and restrictions of the

A key aspect of data services that is underdeveloped in current product and service offerings, yet extremely important, is data security.

service-enabled data sources as a basis for query formulation, rather than just the externally visible data service metadata, to guide the users toward formulating answerable queries. More work is needed here to handle broader classes of queries.

Data service query optimization. In the case of integrated data services with a functional external model, one could imagine defining a set of semantic equivalence rules that would allow a query processor to substitute a data service call used in a query for another service call in order to optimize the query execution time, thus enabling semantic data service optimization. For example, the following equivalence rule captures the semantic equivalence of the data services `getOrderHistory` and `getOpenOrders` when a `[status = 'open']` condition is applied to the former:

```
getOrderHistory(cid) [status =
'open'] ≡ getOpenOrders(cid)
```

Work is needed here to help data service architects to specify such rules and their associated trade-offs “easily” and to teach query optimizers to exploit them.

Very large functional models. For data services using a functional external model, if the number of functions is very large, it is difficult or even impossible for the data owner to explicitly enumerate all functions and for the query developer to have a global picture of them. Consider the example of a data owner, who, for performance reasons, only wants to allow queries that use some non-empty subset of a set of n filter predicates. Enumerating all the $2^n - 1$ combinations as functions would be tedious and impractical for large n . Recent work has studied how models consisting of such large collections of functions, where the function bodies are defined by XPath queries, can be compactly specified using a grammar-like formalism⁴⁰ and how queries over the output schema of such a service can be answered using the model.¹⁷ More work is needed here to extend the formalism and the query answering algorithms to larger classes of queries and to support functions that perform updates.

Cloud data service integration. Since consumers and small businesses are

starting to store their data in the cloud, it makes sense to think about data service integration in the cloud when there are many, many small data services available. For example, how can Clark Cloudlover integrate his Google calendar with his wife's Apple iCal calendar? At this point, there is no homogeneity in data representation and querying, and the number of cloud service providers is rapidly increasing. Google Fusion Tables²⁷ is one example of a system that follows this trend and allows its users to upload tabular data sets (spreadsheets), to store them in the cloud, and subsequently to integrate and query them. Users are able to integrate their data with other publicly available datasets by performing left outer joins on primary keys, called table merges. Fusion Tables also visualize users' data using maps, graphs, and other techniques. Work is needed on many aspects of cloud data sharing and integration.

Data summaries. As the number of data services increases to a “consumer scale,” it will be difficult even to find the data services of interest and to differentiate among data services whose output schemas are similar. One approach to easing this problem is to offer *data summaries* that can be searched and that can give data service consumers an idea of what lies behind a given data service. Data sampling and summarization techniques that have been traditionally employed for query optimization can serve as a basis for work on large-scale data service characterization and discovery.

Cloud data service security. Storing proprietary or confidential data in the cloud obviously creates new security problems. Currently, there are two broad choices. Data owners can either encrypt their data, but this means that all but exact-match queries have to be processed on the client, moving large volumes of data across the cloud, or they must trust cloud providers with their data, hoping there are enough security mechanisms in the cloud to guard against malicious applications and services that might try to access data that does not belong to them. There is early ongoing work²⁴ that may help to bridge this gap by enabling queries and updates over encrypted data, but much more work is needed to see

if practical (for example, efficient) approaches and techniques can indeed be developed.

Acknowledgments

We would to thank Divyakant Agrawal (UC Santa Barbara), Pablo Castro (Microsoft), Alon Halevy (Google), James Hamilton (Amazon) and Joshua Spiegel (Oracle) for their detailed comments on an earlier version of this article. We also thank the associate editor and anonymous reviewers for feedback that improved the quality of this article. This work was supported in part by NSF IIS awards 0910989, 0713672, and 1018961. **C**

References

1. Amazon Web Services, 2010. <http://aws.amazon.com/>.
2. Adya, A., Blakeley, J.A., Melnik, S., and Muralidhar, S. Anatomy of the ADD.NET entity framework. In *Proceedings of SIGMOD Conference* (2007), 877–888.
3. Agrawal, D., Abbadi, A.E., Antony, S. and Das, S. Data management challenges in cloud computing infrastructures. In *DNIS* (2010), 1–10.
4. Allen A. Friedman and Darrell M. West. Privacy and security in cloud computing. *Issues in Technology Innovation* 3 (Oct. 2010), Center for Technology Innovation at Brookings.
5. Baker, J., Bond, C., Corbett, J.C., Furman, J., Khorlin, A., Larson, J., Léon, J.M., Li, Y., Lloyd, A. and Yushprakh, V. Megastore: Providing scalable, highly available storage for interactive services. In *Proceedings of CIDR Conference*, 2011.
6. Bancilhon, F. and Spyrtos, N. Update semantics of relational views. *ACM Trans. Database Syst.* 6 (Dec. 1981), 557–575.
7. Bernstein, P.A., Cseri, I., Dani, N., Ellis, N., Kalhan, A., Kakivaya, G., Lomet, D.B., Manne, R., Novik, L. and Talius, T. Adapting Microsoft SQL Server for cloud computing. In *Proceedings of IEEE International Conference on Data Engineering* (2011), 1255–1263.
8. Blow, M., Borkar, V., Carey, M., Hillery, C., Kotopoulos, A., Lychagin, D., Preotiuc-Pietro, R., Reveliotis, P., Spiegel, J. and Westmann, T. Updates in the AquaLogic Data Services Platform. In *Proceedings of IEEE International Conference on Data Engineering* (2009), 1431–1442.
9. Boag, S., Chamberlin, D., Fernandez, M.F., Florescu, D., Robie, J. and Siméon, J. XQuery 1.0: An XML Query Language. W3C Recommendation (Jan. 23, 2007); <http://www.w3.org/TR/xquery/>.
10. Borkar, V.R., Carey, M.J., Engovatov, D., Lychagin, D., Reveliotis, P., Spiegel, J., Thattai, S. and Westmann, T. Access Control in the AquaLogic Data Services Platform. In *Proceedings of SIGMOD Conference* (2009), 939–946.
11. Brantner, M., Florescu, D., Graf, D.A., Kossmann, D. and Kraska, T. Building a database on S3. In *Proceedings of SIGMOD Conference* (2008), 251–264.
12. Britton-Lee Inc. IDM 500 Software Reference Manual Version 1.3, 1981.
13. Carey, M.J. Data delivery in a service-oriented world: The BEA AquaLogic data services platform. In *Proceedings of SIGMOD Conference* (2006), 695–705.
14. Carey, M.J. SOA what? *IEEE Computer* 41, 3 (2008), 92–94.
15. The Cassandra Apache Project, 2009; <http://cassandra.apache.org/>.
16. Cattell, R. Scalable SQL and NoSQL data stores. *SIGMOD Record* 39, 4 (2010), 12–27.
17. Cautis, B., Deutsch, A., Onose, N. and Vassalos, V. Efficient rewriting of XPath queries using query set specifications. *PVLDB* 2, 1 (2009), 301–312.
18. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A. and Gruber, R.E. Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, (2006).
19. Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S. Web Services Description Language (WSDL) 1.1. W3C Note (Mar. 15, 2001); <http://www.w3.org/TR/wsdl>.
20. Cooper, B.F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.A., Puz, N., Weaver, D. and Yerneni, R. PNUTS: Yahoo!'s hosted data serving platform. In *Proc. VLDB Endow* 1, 2 (2008), 1277–1288.
21. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. and Vogels, W. Dynamo: Amazon's highly available key-value store. In *SOSP* (2007), 205–220.
22. Fielding, R.T. Architectural Styles and the Design of Network-based Software Architectures. Ph.D. thesis, University of California, Irvine, 2000.
23. Garcia-Molina, H. and Salem, K. Sagas. *SIGMOD Rec.* 16, 3 (1987), 249–259.
24. Gentry, C. Computing arbitrary functions of encrypted data. *Commun. ACM* 53, 3 (2010), 97–105.
25. Google. Google App Engine. <http://code.google.com/appengine>.
26. Google. Protocol Buffers: Google's data interchange format, 2008; <http://code.google.com/p/protobuf>.
27. Google. Fusion Tables, 2009; <http://tables.googlelabs.com>.
28. Gregorio, J. and de Hóra, B. The Atom Publishing Protocol, 2005; <http://datatracker.ietf.org/doc/rfc5023/>.
29. HBase, 2010; <http://hbase.apache.org/>.
30. Helland, P. Life beyond distributed transactions, an Apostate's opinion. In *Proceeding of Conference on Innovative Data System Research* (2007).
31. JavaScript Object Notation; <http://www.json.org/>.
32. Jordan, D. and Evdemon, J. Web Services Business Process Execution Language Version 2.0. OASIS Standard (Apr. 11, 2007); <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html>.
33. Y. Kotidis, D. Srivastava, and Y. Velegrakis. Updates through views: A new hope. In *Proceedings of IEEE International Conference on Data Engineering* (2006), 2.
34. Microsoft, Inc. Windows Azure, 2009; <http://www.microsoft.com/windowsazure/windowsazure/>.
35. Microsoft, Inc. WCF Data Services, 2011; <http://msdn.microsoft.com/en-us/data/bb931106>.
36. Mitra, N. and Lafon, Y. SOAP Version 1.2 Part 0: Primer (Second Edition). W3C Recommendation. Apr. 27, 2007; <http://www.w3.org/TR/soap/>.
37. Open Data Protocol (OData), 2010; <http://www.odata.org/>.
38. Oracle and/or its affiliates. Oracle Data Service Integrator 10gR3 (10.3), 2011; http://download.oracle.com/docs/cd/E13162_01/odsi/docs10gr3.
39. Oracle Corporation and/or its affiliates. MySQL; <http://www.mysql.com/>.
40. Petropoulos, M., Deutsch, A. and Papakonstantinou, Y. The query set specification language (QSSL). *WebDB* (2003), 99–104.
41. Petropoulos, M., Deutsch, A., Papakonstantinou, Y., and Katsis, Y. Exporting and interactively querying Web service-accessed sources: The CLIDE system. *ACM Trans. Database Syst.* 32, 4 (2007).
42. Ramakrishnan, R. and Gehrke, J. *Database Management Systems* (3rd Ed.), McGraw-Hill, 2003.
43. Salesforce.com, Inc. Salesforce.com Object Query Language (SOQL), 2010; http://www.salesforce.com/us/developer/docs/api/Content/sforce_api_calls_soql.htm.
44. Shipman, D.W. The Functional Data Model and the Data Language DAPLEX. *ACM Trans. Database Syst.* 6, 1 (1981), 140–173.
45. Terry, D.B., Theimer, M.M., Petersen, K., Demers, A.J., Spreitzer, M.J. and Hauser, C.H. Managing update conflicts in Bayou, a weakly connected replicated storage system. In *SOSP* (1995), 172–182.
46. Vogels, W. Eventually consistent. *ACM Queue* 6, 6 (2008), 14–19.

Michael J. Carey (mjc Carey@ics.uci.edu) is a professor in the Bren School of Information and Computer Sciences at the University of California, Irvine.

Nicola Onose (onose@google.com) is a software engineer at Google. He conducted this work as a postdoctoral research fellow in the Bren School of Information and Computer Sciences at the University of California, Irvine.

Michalis Petropoulos (mpetropo@gmail.com) is an architect at Greenplum/EMC. He conducted this work while a research scientist in the Computer Science and Engineering department at the University of California, San Diego.

© 2012 ACM 0001-0782/12/06 \$10.00

Exploring the power and potential of geometric complexity theory.

BY KETAN D. MULMULEY

The GCT Program Toward the P vs. NP Problem

GEOMETRIC COMPLEXITY THEORY (GCT) is an approach via algebraic geometry and representation theory toward the P vs. NP and related problems.^{9,13,15,29} It was proposed in a series of papers^{4,18–22,24–26} and was developed further in Bürgisser and Ikenmeyer,⁷ Bürgisser et al.,⁸ and Landsberg et al.¹⁴ This article gives an informal overview of GCT. It is meant to be an update on the status of the P vs. NP problem reported in Fortnow.¹¹ See Mulmuley²³ for a more detailed, formal overview of GCT.

Let us begin by recalling an algebraic variant of the P vs. NP problem introduced in a seminal paper.²⁹ It can be formulated in a very concrete form as the *permanent vs. determinant* problem. Here the permanent of an $n \times n$ variable matrix X is defined just like the determinant but without signs.

Specifically,

$$\det(X) = \sum_{\sigma} \text{sign}(\sigma) \prod_{1 \leq i \leq n} x_{i\sigma(i)} \quad \text{and} \\ \text{perm}(X) = \sum_{\sigma} \prod_{1 \leq i \leq n} x_{i\sigma(i)},$$

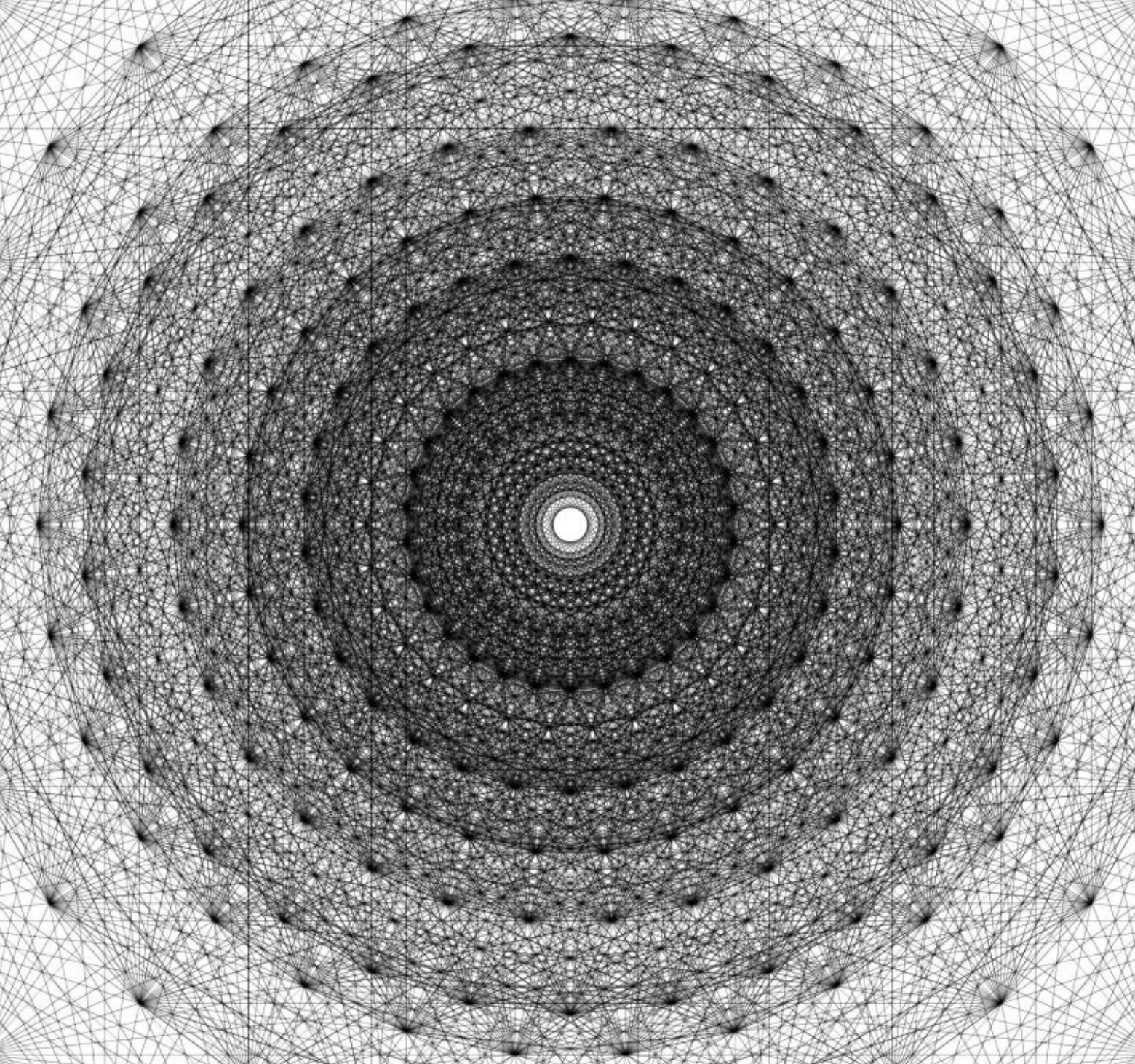
where x_{ij} 's denote the entries of X and σ ranges over all permutations of the integers from 1 to n . Let K , the base field or ring of computation, be $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{C}$, or a finite field F_p of p elements, p being an odd prime. We say that $\text{perm}(X)$ can be *linearly represented* as the determinant of an $m \times m$ matrix if $\text{perm}(X) = \det(Y)$ for some $m \times m$ matrix Y whose entries are linear combinations (possibly nonhomogeneous) over K of the variable entries of X . The permanent vs. determinant conjecture in Valiant²⁹ is that $\text{perm}(X)$ cannot be linearly represented as the determinant of an $m \times m$ matrix when m is small, that is, when m is polynomial in n , or more generally, when it is $O(2^{\log^a n})$ for some constant $a > 0$.

It is known^{3,29} that this conjecture, when K is $\mathbb{Z}$ or F_p , and m is polynomial in n , is implied by a stronger (nonuniform) version^a of the $P \neq NP$ conjecture or even the weaker $\#P \not\subseteq NC$ conjecture. Here, $\#P$ denotes the

^a This version says that NP contains functions that cannot be computed by polynomial size circuits.

» key insights

- GCT provides an approach to the foundational questions of complexity theory via algebraic geometry and representation theory.
- It reveals formidable explicit construction problems at the crossroads of algebraic geometry, representation theory, and complexity theory, and provides evidence that any approach to the foundational questions of complexity theory would have to resolve explicit construction problems of comparable difficulty. This law of conservation of difficulty may explain why P vs. NP and related problems, which look elementary at the surface, have turned out to be so difficult.
- It shows how to break the circle of self-reference around the arithmetic P vs. NP and related problems.



class of functions,^b like the number of satisfying assignments of a Boolean formula, that count the number of solutions of the problems in NP , and NC denotes the class of functions,^b like the determinant, that can be computed efficiently in parallel in polylogarithmic time using polynomially many processors. The implication of the permanent vs. determinant conjecture from the (nonuniform) $\#P$ vs. NC conjecture is based on the fact that the permanent is $\#P$ -complete²⁹ (in the spirit

of the well-known NP -completeness) and that the determinant is (almost) NC -complete. It is also known that the permanent vs. determinant conjecture, when K is a large enough finite field F_p and $m = O(2^{c \log^2 n})$ for some large enough constant $c > 0$, implies the $\#P \not\subseteq NC$ conjecture. As such, the permanent vs. determinant conjecture is, strictly speaking, an algebraic analogue of the $\#P$ vs. NC conjecture, not the P vs. NP conjecture. There is also an analogous algebraic analogue of the P vs. NP conjecture^{21,25} which, when K is a large enough finite field, implies the usual $P \neq NP$ conjecture. But its story

is similar to that of the permanent vs. determinant conjecture. Hence, for simplicity, we only focus on the permanent vs. determinant conjecture here.

By the arithmetic case of this conjecture, we mean the case when $K = \mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{C}$. This case for $K = \mathbb{Z}$ is implied by the case when $K = F_p$ and also, as already mentioned, by the (nonuniform) $\#P$ vs. NC conjecture. The arithmetic case is easier than the case when $K = F_p$, because it avoids complications in algebra that arise in the case of finite fields.

Hence, let us first discuss the arithmetic case when $K = \mathbb{C}$, which implies

^b This definition of NC is broader than the usual definition that allows only 0–1 functions.

the cases when $K = \mathbb{Z}$ or $\mathbb{Q}$. The advantage of dealing with the arithmetic conjecture over $\mathbb{C}$, in contrast to the original Boolean conjectures, is that this arithmetic conjecture is a statement about multivariate polynomials over $\mathbb{C}$. Hence, we can use techniques from algebraic geometry, which is the study of the common zeroes of sets of multivariate polynomials. These techniques work best when the base field is algebraically closed of characteristic zero, such as $\mathbb{C}$. Since the permanent and the determinant are characterized by their symmetries, we can also use techniques from representation theory, which is the study of groups of symmetries. As such, the GCT approach that goes via algebraic geometry and representation theory is very natural in the arithmetic setting.

The articles by Mulmuley and Sohoni^{25,26} reduce the arithmetic permanent vs. determinant conjecture to proving existence of *geometric obstructions* that are proof certificates of hardness of the permanent. The very existence of these obstructions for given n and m implies that the permanent of an $n \times n$ variable matrix cannot be linearly represented as the determinant of an $m \times m$ matrix. The geometric obstructions are objects that live in the world of algebraic geometry and representation theory. Their dimensions can be large, exponential in n and m . But they have short classifying labels. The basic strategy of GCT, called the *flip*,^{21,22} is to construct the classifying label of some geometric obstruction *explicitly* in time polynomial in n and m when m is small. It is called the flip because it reduces the lower bound problem under consideration to the upper bound problem of constructing a geometric obstruction label efficiently. The flip basically means proving lower bounds using upper bounds. Its basic idea in a nutshell is (1) to understand the theory of upper bounds (algorithms) first and (2) to use this theory to prove lower bounds later. But one may wonder why we are going for explicit construction of obstructions, when proving existence of an obstruction even nonconstructively suffices in principle. This is because of the flip theorem in Mulmuley,^{21,23} which says that in the problem under consideration we are essentially *forced*

to construct some proof certificate of hardness explicitly.

The upper bound problems that arise in the context of the flip turn out to be formidable problems at the frontier of algebraic geometry. The flip theorem mentioned above also says that stronger versions of the permanent vs. determinant conjecture and a standard derandomization conjecture¹² in complexity theory imply together solutions to the upper bound problems in algebraic geometry that are akin to the ones that arise in the flip. Furthermore the article by Mulmuley²² gives evidence that even the upper bound problems that arise in the flip may be essentially implications of these conjectures in complexity theory. This suggests a *law of conservation of difficulty*, namely, that problems comparable in difficulty to the ones encountered in GCT would be encountered in *any* approach to the (nonuniform) P vs. NP problem (of which the arithmetic permanent vs. determinant conjecture over $\mathbb{Z}$ is an implication). This does not say that any approach to the P vs. NP problem has to necessarily go via algebraic geometry. But it does suggest that avoiding algebraic geometry may not be pragmatic since it would essentially amount to reinventing in some guise the wheels of this difficult field that have been developed over centuries.

There is also another reason why the explicit construction of geometric obstruction labels turns out to be hard. At the surface it seems that for such efficient construction one may need to compute the permanent itself efficiently, thereby contradicting the very hardness of the permanent that we are trying to prove. By the flip theorem in Mulmuley,^{21,23} this *self-referential difficulty* akin to that in Gödel's Incompleteness Theorem is also not specific to GCT. Any approach would have to cope with it. The article by Mulmuley²² shows how it can be tackled in GCT by decomposing the lower bound problem under consideration into subproblems without this difficulty. Conceptually, this is the main result of GCT in the arithmetic setting.

Finally, let us discuss the permanent vs. determinant conjecture over finite fields that implies the $\#P \not\subseteq NC$ conjecture, the story for the algebraic variant of the P vs. NP problem in

Mulmuley and Sohoni^{21,25} that implies the usual (Boolean) P vs. NP conjecture being similar. Here, the GCT plan is to prove the arithmetic case via algebraic geometry over $\mathbb{C}$ as outlined above first, and then extend this proof to finite fields by proving additional results in algebraic geometry over $\mathbb{C}$, or rather, algebraically closed fields of characteristic zero such as $\mathbb{C}$. At the surface, this plan may seem counter-intuitive. After all, how can one hope to prove statements about finite fields using algebraic geometry over $\mathbb{C}$? A basic prototype for this plan is the analogue of the usual Riemann hypothesis for finite fields proved in Deligne¹⁰ using algebraic geometry over algebraically closed fields of characteristic zero such as $\mathbb{C}$. The proof of this result, a crowning achievement in mathematics, shows that difficult statements about finite fields can be proved using algebraic geometry over algebraically closed fields of characteristic zero. In the same spirit, the GCT approach in the arithmetic setting can be extended so that it applies to the usual (Boolean) $\#P$ vs. NC and P vs. NP conjectures. But this story is beyond the scope of this article. It will be described in a later paper.¹⁷ In this article, we confine ourselves to the arithmetic permanent vs. determinant problem, which captures the crux of the P vs. NP problem.

The rest of this article is organized as follows. We describe the notion of geometric obstructions for the arithmetic permanent vs. determinant problem. This is followed by a description of the *flip* strategy that goes toward *explicit* construction of geometric obstruction labels in polynomial time. We state the upper bound problems in algebraic geometry that arise in this context. We also describe the self-referential difficulty in the problem under consideration and how GCT tackles it by decomposing the problem into subproblems without this difficulty.

Geometric Obstructions

We now describe the GCT approach to the arithmetic permanent vs. determinant problem²⁹ over $\mathbb{C}$ based on the notion of geometric obstructions (proof certificates of hardness).

The starting point of the approach is the classical result that the permanent and determinant are completely

characterized by their symmetries in the following sense.²⁵

(D): Let Y be a variable $m \times m$ matrix. Then, $\det(Y)$ is the unique polynomial (up to a constant multiple) of degree m in the variable entries of Y such that, for any $m \times m$ invertible complex matrices A and B with $\det(A) \det(B) = 1$, $\det(Y) = \det(AY^*B)$, where Y^* is Y or its transpose.

(P): Let X be a variable $n \times n$ matrix. Then, $\text{perm}(X)$ is the unique polynomial (up to a constant multiple) of degree n in the variable entries of X such that, for any diagonal or permutation matrices A and B , $\text{perm}(X) = \text{perm}(AX^*B)$, where X^* is X or its transpose, and the product of the entries of A is one, when A is diagonal, and similarly for B .

The goal is to solve the problem under consideration by exploiting these properties. Toward this end, the article²⁵ constructs algebraic varieties $\Delta[\text{perm}, n, m]$ and $\Delta[\det, m]$ such that if $\text{perm}(X)$, where X is an $n \times n$ variable matrix, can be linearly represented as the determinant of an $m \times m$ matrix, then

$$\Delta[\text{perm}, n, m] \subseteq \Delta[\det, m]. \quad (1)$$

Here, by an algebraic variety we mean the set of common solutions of a system of multivariate polynomial equations over $\mathbb{C}$. These are generalizations of the usual curves and surfaces. For example, the set of common solutions in $\mathbb{C}^4$ of two polynomial equations

- (1): $x_1^2/a^2 + x_2^2/b^2 + x_3^2/c^2 + x_4^2/d^2 = 0$, $a, b, c, d > 0$, and,
- (2): $x_1^2/a'^2 + x_2^2/b'^2 + x_3^2/c'^2 = x_4$, $a', b', c' > 0$,

is a two-dimensional variety Z formed by intersecting the three-dimensional ellipsoid corresponding to the first equation with the three-dimensional paraboloid corresponding to the second equation. By the coordinate ring of a variety we mean the space of polynomial functions on it. This is obtained by restricting the polynomial functions on the ambient vector space containing the variety to the variety. For example, the coordinate ring of Z here is the space of polynomial functions on $\mathbb{C}^4$ restricted to Z .

The varieties $\Delta[\det, m]$ and $\Delta[\text{perm}, n, m]$ are formally defined later. Intuitively, the points in the

variety $\Delta[\det, m]$ correspond to the functions in the arithmetic analogue of NC called VP ²⁹ or the “limits” of such functions, and the points in $\Delta[\text{perm}, n, m]$ correspond to the functions in the arithmetic analogue of $\#P$ called VNP ²⁹ or the “limits” of such functions. Since the permanent vs. determinant conjecture is the arithmetic analogue of the $\#P$ vs. NC conjecture, it thus suffices to show that the inclusion (1) does not hold when m is small.

The goal is to show using algebraic geometry and representation theory that the inclusion (1) is impossible, as conjectured in Mulmuley and Sohoni,²⁵ when m is polynomial in n . We call this *the strong permanent vs. determinant conjecture*. It implies the original conjecture and is almost equivalent to it in the sense that if (1) holds then $\text{perm}(X)$ can be approximated infinitesimally closely by a linear representation of the form $\det(Y')$, with $\dim(Y') = m$. The following is a partial result toward the above stronger conjecture.

Theorem¹⁴ The inclusion (1) is impossible if $m \leq n^2/2$.

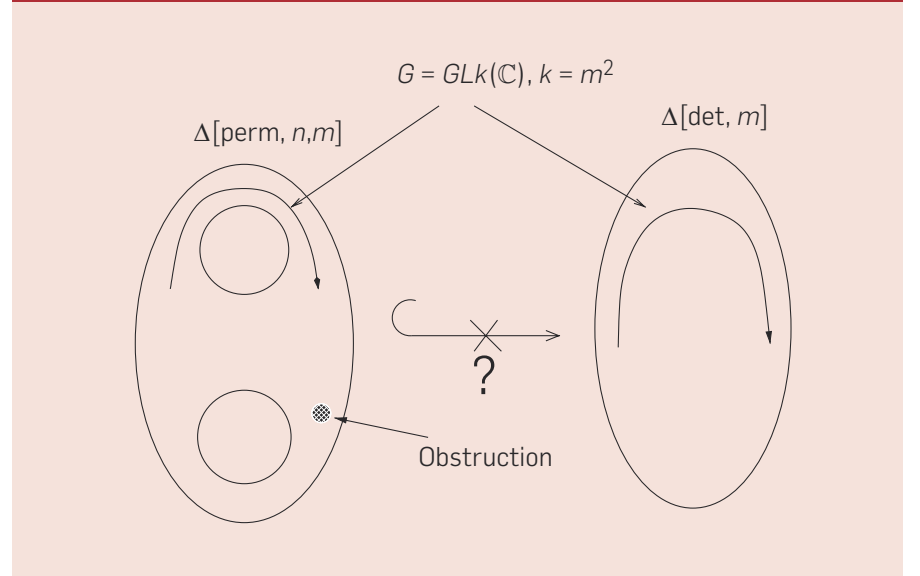
This implies the earlier quadratic lower bound¹⁶ for the permanent but is a bit stronger.

As an aid to prove the strong permanent vs. determinant conjecture in general, Mulmuley and Sohoni²⁶ define the notion of a *geometric obstruction* to the inclusion (1). Informally, a geometric obstruction is a representation-theoretic object that lives on $\Delta[\text{perm}, n, m]$

but not on $\Delta[\det, m]$; see Figure 1. The very existence of such an obstruction serves as a guarantee that the inclusion as in (1) is not possible, because otherwise the obstruction would be living on $\Delta[\det, m]$ as well.

To define geometric obstructions precisely, we need to recall some basic facts from representation theory. Let $G = GL_k(\mathbb{C})$ be the general linear group of $k \times k$ complex invertible matrices. We call a vector space W a representation of G if there is a homomorphism from G to the group of invertible linear transformations of W . For example, $\mathbb{C}^k$ with the usual action of G is its standard representation. There are, of course, far more complex representations of G . Their building blocks were classified by Hermann Weyl.³⁰ He showed that irreducible (polynomial) representations of G are in one-to-one correspondence with nonnegative integer sequences (called partitions) $\lambda = (\lambda_1, \dots, \lambda_l)$, where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_l \geq 0$, and $l \leq k$. An irreducible representation of G in correspondence with λ is denoted by $V_\lambda(G)$. It is called a *Weyl module* of G . For example, the standard representation $\mathbb{C}^k$ of G mentioned above is the Weyl module corresponding to the partition (1) consisting of just one integer 1. The Weyl module $V_\lambda(G)$, when $\lambda = r$, is simply the space $\text{Sym}^r(z_1, \dots, z_k)$ of all homogeneous polynomials of degree r in the variables $z_1, \dots, z_k$ with the following action of G . Given a polynomial $f(\bar{z}) = f(z_1, \dots, z_k) \in \text{Sym}^r(z_1, \dots, z_k)$ and $\sigma \in G$, map $f(\bar{z})$ to

Figure 1. A geometric obstruction.



Formal Definition of the Varieties

For the interested readers, we now formally define the varieties $\Delta[\det, m]$ and $\Delta[\text{perm}, n, m]$ and the action of G on them.

Let Y be an $m \times m$ variable matrix. Let X be an $n \times n$ submatrix of Y , say its lower-right $n \times n$ subminor. Let z be any entry of Y outside X . Let V be the vector space of homogeneous polynomials of degree m in the variable entries of Y . Thus, $\det(Y)$ is an element of V . Let $\tilde{\Delta}[\det, m]$ be the set of elements in V of the form $\det(Y')$, where Y' is an $m \times m$ matrix whose entries are complex homogeneous linear combinations of the variable entries of Y . Then, $\Delta[\det, m] \subseteq V$ is the closure of $\tilde{\Delta}[\det, m]$ in V in the usual complex topology of V . It can be shown to be an algebraic variety. The variety $\Delta[\text{perm}, n, m] \subseteq V$ is constructed similarly using the homogeneous polynomial $z^{m-n} \text{perm}(X) \in V$ in place of the determinant. It can be shown²⁵ that these varieties have the required property mentioned in (1). Actually the varieties here are the affine cones of the projective varieties defined in Mulmuley and Sohoni.²⁵

The action of $G = GL_k(\mathbb{C})$, $k = m^2$, on these varieties is defined as follows. First, observe that the space V is a representation of G by a natural action which, for any matrix $\sigma \in G$, maps any point (homogeneous polynomial) $p(Y) \in V$ to the point $p(\sigma^{-1}Y)$. Here, we think of Y as an m^2 -vector by straightening it, say, rowwise. It can be shown that $\Delta[\det, m]$ and $\Delta[\text{perm}, n, m]$ are invariant under this action of G on V . This induces a natural action of G on these varieties as well. Under this action, each matrix in G acts on a variety by moving its points around and thereby inducing an automorphism. With this action, the coordinate ring of $\Delta[\det, m]$, by which we mean the space of polynomial functions on V restricted to $\Delta[\det, m]$, becomes a representation of G : just map a polynomial function $f(v)$ to $f(\sigma^{-1}v)$ for any $\sigma \in G$ and $v \in \Delta[\det, m]$. Here $\sigma^{-1}v$ denotes the point in $\Delta[\det, m]$ obtained by letting $\sigma^{-1} \in G$ act on v . The coordinate ring of $\Delta[\text{perm}, n, m]$ is similarly a representation of G .

$$f^\sigma(\bar{z}) = f(\bar{z}\sigma). \quad (2)$$

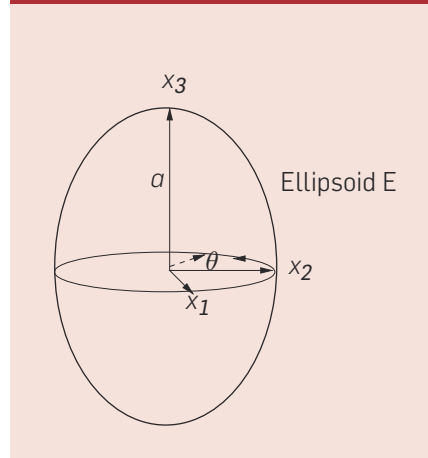
Each finite-dimensional representation of G is like a complex building that can be decomposed into the building blocks—the Weyl modules. Fundamental significance of Weyl's classification results from the complexity theoretic perspective is the following. The dimension of each Weyl module $V_\lambda(K)$ is in general exponential in the bit length of λ . But it has a compact (polynomial size) specification, namely, the labeling partition λ . Existence of such compact specifications of irreducible representations of G plays a crucial role in what follows.

If W is a representation of G , then the elements of G act on W , moving its points around via invertible linear transformations. More generally, a group can similarly act on a variety too. As a simple example, consider the ellipsoid $E \subseteq \mathbb{R}^3$ with the equation $x_1^2 + x_2^2 + x_3^2/a = 0$, $a > 0$. Let U be the unit circle. It becomes an additive group if we identify each point in U with its polar coordinate θ and let the usual addition of angles play the role of the group composition. The group U has a natural action on E : let $\theta \in U$ act on E by rotating E around the x_3 axis by the angle θ ; see Figure 2. Let $\mathbb{R}[E]$

be the coordinate ring of E . This is the space of polynomial functions on $\mathbb{R}^3$ restricted to E . Then, this action of U on E also makes $\mathbb{R}[E]$ a representation of U : given $\theta \in U$ just map any polynomial function $f(\bar{x}) = f(x_1, x_2, x_3)$ on E to $f(\theta \cdot \bar{x})$, where $\theta \cdot \bar{x} \in E$ denotes the point obtained by rotating $\bar{x} \in E$ around the x_3 axis by the angle θ .

Similarly, the group $G = GL_k\mathbb{C}$, with $k = m^2$, acts on the varieties $\Delta[\det, m]$ and $\Delta[\text{perm}, n, m]$ moving their points around (Figure 1), and this action of G on the varieties makes their coordinate rings (the spaces of polynomial functions on them) representations of

Figure 2. An ellipsoid.



G . A formal definition of the action of G and the representation structures of the coordinate rings of $\Delta[\det, m]$ and $\Delta[\text{perm}, n, m]$ are discussed later.

These representation structures turn out²⁶ to depend critically on the properties (D) and (P), respectively. Specifically, the properties (D) and (P) put strong restrictions as to which irreducible representations of G can occur as G -subrepresentations of these coordinate rings.

Formally, a geometric obstruction to the inclusion (1) for given n and m is an irreducible representation $V_\lambda(G)$ of G (a Weyl module) that occurs as a G -subrepresentation in the coordinate ring of $\Delta[\text{perm}, n, m]$ but not in the coordinate ring of $\Delta[\det, m]$; see Figure 1. The partition λ here is called a *geometric obstruction label*. The existence of such an obstruction guarantees that the inclusion as in (1) is impossible, because otherwise the obstruction would occur as a G -subrepresentation in the coordinate ring of $\Delta[\det, m]$ as well.

Thus, to solve the (strong) permanent vs. determinant conjecture, it suffices to show the following:

Geometric obstruction hypothesis (GOH).²⁶ A geometric obstruction exists when m is polynomial in n .

It is conjectured in Mulmuley²² that GOH, or rather its slightly relaxed form, is *equivalent* to the strong permanent vs. determinant conjecture.

The Flip

With the help of GOH, we have reduced the nonexistence problem under consideration to an existence problem. For general varieties, such an existence problem is hopeless. But we can hope to prove existence of a geometric obstruction using the characterization by symmetries provided by the properties (P) and (D). We turn to this story here.

The strategy is to construct, for any n and m polynomial in n , a geometric obstruction label λ *explicitly* in time polynomial in n and m by exploiting the properties (P) and (D). We call this strategy the *flip*, because it reduces the nonexistence problem under consideration to the problem of proving existence of a geometric obstruction, and furthermore, the

c Strictly speaking, we have to use the duals of the coordinate rings here.

lower bound problem is reduced to the upper bound problem of constructing a geometric obstruction label in polynomial time.

The following is a stronger and precise explicit form of GOH that says geometric obstructions can indeed be constructed *explicitly*.

Flip Hypothesis (FH).^{22,23} The geometric obstruction family is *explicit* in the sense that it satisfies the following properties:

FH[Short]: A short geometric obstruction label λ , with bit length polynomial in n and m , exists if m is polynomial in n .

FH[Verification]: Given n , m , and a partition λ , it can be verified whether λ is a valid geometric obstruction label in time polynomial in n , m and the bit length of λ .

FH[Discovery and construction]: Given n and m , it can be decided whether a geometric obstruction exists in time polynomial in n and m . If an obstruction exists, one such geometric obstruction label λ can also be constructed in the same time. By FH[short], this discovery algorithm always succeeds if m is polynomial in n .

FH[Det]: For given m and λ , it can be verified whether $V_\lambda(G)$ occurs as a G -subrepresentation in the coordinate ring^d of $\Delta[\det, m]$ in time polynomial in m and the bit length of λ .

FH[Perm]: For given n , m and λ , it can be verified whether $V_\lambda(G)$ occurs as a G -subrepresentation in the coordinate ring of $\Delta[\text{perm}, n, m]$ in time polynomial in n , m and the bit length of λ .

The flip strategy can now be elaborated further in three steps: (1) Prove FH[Det] and FH[Perm]. This clearly implies an efficient criterion for verifying a geometric obstruction label as in FH[Verification]. (2) Use this criterion to design an efficient algorithm for discovering an obstruction as in FH[Discovery]. (3) Prove that this discovery algorithm always succeeds if m is a polynomial in n . For this strategy to succeed, it is not enough if the verification and discovery algorithms are efficient only in theory. They should also have simple enough mathematical structure to carry out step (3). Otherwise, they have to be made simpler and simpler until (3) succeeds.

Later, we discuss why FH should



The best algorithms for verification and construction of a geometric obstruction label based on general-purpose algorithms in algebraic geometry and representation theory take triple exponential time in n and m in the worst case.



hold. There is a huge gap between FH and what can be proved at present. Currently, the best algorithms for verification and construction of a geometric obstruction label based on general-purpose algorithms in algebraic geometry and representation theory take triple exponential time in n and m in the worst case. FH says this time bound can be brought down to a polynomial. This may seem impossible.

Why Go for Explicit Proofs?

If so, one may ask as to why we should go for explicit construction of obstructions when proving existence of obstructions even nonconstructively suffices in principle. The reason is provided by the strong flip theorem.²¹ It says that any proof of the arithmetic (strong) permanent vs. determinant conjecture can be converted into an explicit proof assuming a stronger form of a standard derandomization hypothesis¹² in complexity theory that is generally regarded as easier than the target lower bound. By an explicit proof, we mean that the proof also yields an algorithm for efficient construction of some proof certificate of hardness of the permanent, called an *obstruction*, that is analogous to the geometric obstruction above in the following sense: (1) its very existence for given n and m guarantees that the inclusion (1) is impossible, and (2) the family of obstructions satisfies analogues of FH[short], FH[verify], and FH. Thus, by the strong flip theorem, the strong permanent vs. determinant conjecture essentially *forces* an explicit proof, modulo derandomization.

There are similar flip theorems²¹ for other lower bound problems, such as the usual permanent vs. determinant and the arithmetic P vs. NP problems, and a certain average case stronger form of the Boolean P vs. NP problem. These results are the main reason why we are going toward explicit proofs, that is, toward explicit construction of obstructions, right from the beginning.

The derandomization hypothesis mentioned here is the following. Its importance is based on the fundamental result in Kabanets and Impagliazzo¹² that derandomization means proving circuit lower bounds. Let $Y(X)$ be an $m \times m$ matrix, whose each entry is a complex linear combination

^d Actually its dual; and similarly in FH[Perm].

(possibly nonhomogeneous) of the variable entries of X . The problem is to decide if $\det(Y(X))$, for given $Y(X)$, is an identically zero polynomial in the variable entries of X . There is a simple and efficient randomized algorithm for this test. Let A be a matrix obtained from X by substituting for each entry of X a large enough random integer of bit length polynomial in n and m . Evaluate $\det(Y(A))$ modulo a large enough random integer b . If it is nonzero, then $\det(Y(X))$ is certainly a nonzero polynomial. If it is zero, then $\det(Y(X))$ is an identically zero polynomial with a high probability. This randomized test is a black-box test in the sense it only needs to know the value of $\det(Y(X))$ for a given specialization of X to A . It does not need to know $Y(X)$. The derandomization hypothesis mentioned earlier is essentially that this randomized black-box determinant identity test can be efficiently derandomized so as to get an efficiently deterministic black-box determinant identity testing algorithm. (The required hypothesis is actually a bit stronger.²¹) This derandomization hypothesis, which is somewhat different from the one in Kabanets and Impagliazzo,¹² is essentially equivalent to proving a determinantal lower bound for a multilinear function that can be evaluated in exponential time¹. This is generally regarded as easier than proving a determinantal lower bound for the permanent since $\#P$ is conjecturally smaller than EXP , the class of functions that can be computed in exponential time.

Why Should GOH and FH Hold?

The strong flip theorem^{21,23} actually shows something much more. It shows that stronger forms of the permanent vs. determinant and derandomization conjectures together imply an analogue of FH in algebraic geometry of comparable difficulty. This reveals that formidable upper bound problems in algebraic geometry are hidden underneath the fundamental hardness and derandomization conjectures in complexity theory. This may explain why these conjectures, which look so elementary at the surface, have turned out to be so formidable. In view of the strong flip theorem, problems of comparable difficulty can be expected in *any* approach, even if the approach does not go via algebraic

The Strong Flip Theorem

To state the strong flip theorem, we need a few definitions. Let $\Delta[\det, m]$, $\Delta[\text{perm}, n, m]$, and V be as in the sidebar “Formal Definition of the Varieties.”

By a *global obstruction set* for given n and small m polynomial in n , we mean a set $S_{n,m} = \{X_1, \dots, X_l\}$ of nonnegative integral $n \times n$ matrices with the following property. Fix any point (homogeneous polynomial) $p(Y) \in \Delta[\det, m] \subseteq V$. Let $p'(X)$ denote the polynomial obtained from $p(Y)$ by substituting zero for all variables in Y other than z and X , and 1 for z . Then, for any such $p(Y) \in \Delta[\det, m]$, there exists a counterexample $X_i \in S_{n,m}$ such that $p'(X_i) \neq \text{perm}(X_i)$. Thus, $S_{n,m}$ contains a counterexample against every point in $\Delta[\det, m]$ that shows that the point does not specialize to $\text{perm}(X)$. This guarantees that the inclusion as in (1) is not possible for given n and m . We say that $S_{n,m}$ is small if l is polynomial in n .

We call a proof of the strong permanent vs. determinant conjecture *extremely explicit* if, for each n and small m polynomial in n , it shows existence of a set of bit strings called *obstructions*, which serve as proof certificates of hardness of $\text{perm}(X)$, with the following properties E0–E3.

- ▶ E0 [Short]: For every n and small m polynomial in n , there exists a short obstruction of bit length polynomial in n .
- ▶ E1 [Easy to decode]: Given any such short obstruction s for given n and small m polynomial in n , one can construct in time polynomial in n a small *global obstruction set* $S_{n,m}(s)$. Thus, each short obstruction s denotes a small global obstruction set.
- ▶ E2 [Easy to verify]: Given a bit string s and n and m , it can be decided whether s is the specification of an obstruction for n and m in time polynomial in n and m , and the bit length of s .
- ▶ E3 [Easy to construct]: For each n and small m polynomial in n , a valid obstruction can be constructed in time polynomial in n .

There are some additional technical properties that an extremely explicit proof has to satisfy; see Mulmuley^{21,23} for its details. The properties E0, E2, and E3 are analogues of the properties FH[Short], FH[Verification], and FH[Construction] of geometric obstruction labels (which we identify with geometric obstructions). The geometric obstruction labels also conjecturally satisfy the analogue of E1 for decoding (though this was not stated in the statement of FH).

We call a proof *extremely explicit in a stronger NC-sense*, if the various algorithms in the conditions E1–E3 work in polylogarithmic time using polynomial number of processors instead of sequential polynomial time.

We say that a technique for proving the strong permanent vs. determinant conjecture is a *flip* if it leads to an extremely explicit proof of the conjecture. It is called a flip because it reduces the nonexistence problem to the problem of proving existence of obstructions and the lower bound problem to the upper bound problem of finding efficient algorithms to verify, construct, and decode obstructions.

For similar definitions of explicit proofs for other lower bound problems, such as the usual permanent vs. determinant and P vs. NP problems, see Mulmuley.^{21,23}

The Strong Flip Theorem.^{21,23} Suppose the strong permanent vs. determinant conjecture holds and that black box determinant identity testing¹² can be derandomized (in a stronger form as specified in Mulmuley²¹). Then, the strong permanent vs. determinant conjecture has an extremely explicit proof in the stronger NC-sense. In particular, for any m polynomial in n , an explicit global obstruction set $S_{n,m}$ can be constructed in time polynomial in n and, more strongly, in polylogarithmic time (in n) using polynomial number of processors.

The proof of the strong flip theorem depends critically on the characterization by symmetries of the permanent as per the property (P). Alternatively, one can also use downward self-reducibility of the permanent that has several other applications in complexity theory.³

Currently, the best algorithms for the construction of a global obstruction set $S_{n,m}$ based on general purpose algorithms in algebraic geometry take triple exponential time in n and m in the worst case and the bit length of the constructed obstruction is double exponential in the worst case. Bringing this time down to polynomial may seem impossible at the surface. This situation is very similar to that for FH (see the remarks after the statement of FH). Thus, the difficulty of proving E0, E1, E2, and E3 for global obstruction sets is comparable to the difficulty of proving FH for geometric obstructions. The strong flip theorem says that the time bound can indeed be brought down to polynomial for geometric obstruction sets assuming the strong permanent vs. determinant conjecture and the derandomization hypothesis.

geometry. We refer to this as the “*law of conservation of difficulty*.”

The article by Mulmuley²² gives evidence based on the strong flip theorem and additional results in algebraic

geometry, which suggests that FH itself may be in essence an *implication* of the strong permanent vs. determinant and derandomization conjectures together. At present, this is the main evidence for

Self-Referential Difficulty

The strong flip theorem also reveals the self-referential difficulty in the permanent vs. determinant conjecture.

To see this, let us examine closely the properties E1–E3 of an explicit proof that any proof of this conjecture can be converted into modulo derandomization by the strong flip theorem. Let $S_{n,m}(s)$ be a global obstruction set denoted by an obstruction string s . To verify if $S_{n,m}(s)$ indeed contains a counterexample against a given point (homogeneous polynomial) $p(Y) \in \Delta[\det, m]$, we have to check if $p'(X) \neq \text{perm}(X)$ for some $X \in S_{n,m}(s)$. Assuming that the permanent is hard to compute, we cannot check efficiently if $p'(X) \neq \text{perm}(X)$ for general X . Yet, by E2 (in the stronger *NC*-sense), it can be checked efficiently whether $S_{n,m}(s)$ contains a counterexample against *every* $p(Y) \in \Delta_v[g, m]$ (even in parallel). This seems to contradict the very hardness of the permanent that we are trying to prove.

This self-referential paradox is only an *apparent* paradox, because assuming the permanent vs. determinant conjecture and an additional derandomization hypothesis we can construct an extremely explicit proof by the strong flip theorem. But this is a circular argument. The main difficulty is to make headway in the construction of an explicit proof *without making an assumption in any guise* that is as hard as or harder than the target lower bound assumption.

Analogous flip theorems²¹ reveal similar self-referential difficulty in other variants of the *P* vs. *NP* problem harder than derandomization, such as the arithmetic *P* vs. *NP* problem.²⁵ Intuitively, the apparent self-referential paradox arises because the *P* vs. *NP* conjecture, being a universal statement about mathematics that says that discovery is hard, can *potentially* make the discovery of its own proof hard.

The situation here is akin to (but far

harder than) the situation for another universal statement about mathematics, Gödel's Incompleteness Theorem. This result says that there are true statements that cannot be proved. This does not say that this universal statement itself cannot be proved. As we know now, it can be proved. But the crux of this proof is the resolution of this *apparent* self-referential paradox by the construction of a statement that says "I cannot be proved." Similarly, the root difficulty in the *P* vs. *NP* (and the permanent vs. determinant) problem is the resolution of the *apparent self-referential paradox* in the construction of the statement that says "I am different from *NP* (*#P*)."

In view of this self-referential paradox, the main conceptual difficulty in proving the permanent vs. determinant conjecture is to *break the circle* of self-reference by decomposing the conjecture into subproblems without the self-referential difficulty.

The Decomposition

We now describe how the circle of self-reference is broken in GCT.

Toward this end, observe that $\text{FH}[\text{Det}]$ does not have the self-referential difficulty in the sense that (1) m is not required to be a small function of n in its statement, and (2) it only depends on the properties of the determinant, and not on the relationship between the permanent and the determinant (or equivalently, between the complexity classes *#P* and *NC*). The case of $\text{FH}[\text{Perm}]$ is similar.

$\text{FH}[\text{Det}]$ and $\text{FH}[\text{Perm}]$ together imply $\text{FH}[\text{verification}]$, which says that geometric obstructions in *GOH* are easy to verify. We saw that the self-referential difficulty is the main obstacle to efficient verification of obstructions as needed in E2. Hence, once $\text{FH}[\text{Det}]$ and $\text{FH}[\text{Perm}]$ are proved, *GOH* does not have the self-referential difficulty in verification any more. This decomposes the strong permanent vs. determinant conjecture into three subproblems without the self-referential difficulty in verification, namely, $\text{FH}[\text{Det}]$, $\text{FH}[\text{Perm}]$, and *GOH*. Pictorially,

$$\begin{array}{l} \text{Strong perm. vs. det.} \leftarrow \cdots \cdots \rightarrow \\ \text{FH}[\text{Det}] + \text{FH}[\text{Perm}] + \text{GOH}. \end{array} \quad (3)$$

Here the solid arrow $\leftarrow$ denotes the formal implication—this follows trivially since *GOH* itself implies the strong permanent vs. determinant conjecture. The dotted arrow

$\cdots \rightarrow$ indicates the evidence given in Mulmuley²² for the plausible converse based on the strong flip theorem. This decomposition breaks the circle of self-reference for verification. Intuitively, the circle is broken here because the task of verifying a geometric obstruction naturally breaks into two independent tasks, one depending only on the permanent (i.e., the complexity class *#P*) and the other only on the determinant (the complexity class *NC*). This is the fundamental difference between geometric obstructions and the global obstruction sets in the strong flip theorem.

The article by Mulmuley²² also describes an approach to prove FH assuming certain *positivity hypotheses* in algebraic geometry and representation theory. The first positivity hypothesis called *PH1* basically says that, for given n, m , and λ , the number of copies of the Weyl module $V_\lambda(G)$ that occur in the coordinate ring of $\Delta[\det, m]$ (and similarly $\Delta[\text{perm}, n, m]$) has a positive (*#P*) formula without alternating signs, akin to the usual positive formula for the permanent. We do not discuss other positivity hypotheses here. These hypotheses are again supported by the strong flip theorem, which suggests²² that these hypotheses too (like FH) may be in essence *implications* of the strong permanent vs. determinant and derandomization conjectures together. Furthermore, the self-referential difficulty is absent in these positivity hypotheses for the same reason that it

is absent in $\text{FH}[\text{Det}]$ and $\text{FH}[\text{Perm}]$. The decomposition theorem^{22,23} decomposes the strong permanent vs. determinant conjecture in terms of these positivity hypotheses and a more refined form of *GOH* (called *OH*), which too is without the self-referential difficulty once the positivity hypotheses are proved. Unlike (3), this decomposition yields an approach to prove $\text{FH}[\text{Discovery}]$ in addition to $\text{FH}[\text{Verification}]$. See Mulmuley^{22,23} for its details.

The positivity hypotheses above turn out to be formidable because as explained in Mulmuley²² they encompass and go much further than the century-old plethysm problem in algebraic geometry and representation theory. Since in view of the strong flip theorem these may be essentially implications of the strong hardness and derandomization conjectures, problems of comparable difficulty can be expected in *any* approach. In this sense, positivity (like explicit construction) is a hidden root difficulty underneath the fundamental hardness conjectures of complexity theory. This provides yet another reason (in addition to the strong flip theorem) for why these conjectures have turned out to be so hard though they look so elementary at the surface.

The articles^{4,19,20,22} suggest an approach to the positivity hypotheses via nonstandard quantum groups. But this story is beyond the scope of this article.

See Mulmuley²² for the GCT approach to the arithmetic *P* vs. *NP* problem.

FH and, hence, GOH. Further evidence is provided by a recent article,⁷ which constructs explicit geometric obstructions in the analogous setting for the lower bound problem for matrix multiplication, albeit for a problem of very modest size. Explicit computation for any larger example is difficult at present due to the difficulty of the problems that arise.

The strong flip theorem for the permanent vs. determinant conjecture and analogous results in Mulmuley²¹ for other fundamental hardness conjectures in complexity theory, such as the arithmetic P vs. NP conjectures, show a fundamental difference between such hardness conjectures that are at least as hard as the derandomization conjectures and the known lower bound results in the restricted models of computation such as constant depth⁵ or monotone²⁷ circuits. The lower bounds in these restricted models are statements about the *weakness* of these models. In contrast, by the strong flip theorem, the permanent vs. determinant problem is a statement about the *strength* of the complexity class NC (or rather its arithmetic analogue²⁹ VP) for which the determinant is essentially complete. It does not say that NC (or rather VP) is small and weak, but rather that it is big and strong—strong enough to assert that “I am different from $\#P$ ” (or rather its arithmetic analogue VNP ²⁹), for the permanent is complete. Similarly, by an analogous flip theorem for the (arithmetic) P vs. NP problem, this problem is a statement about the strength of the complexity class P . It does not say that P is weak and small but rather that it is big and strong—strong enough to assert that “I am different from NP .”

It should also be remarked that FH will almost never hold for functions not characterized by their symmetries (in place of the determinant and the permanent), since the characterization by symmetries plays a crucial role in the proof of the strong flip theorem that forms the crux of the justification of FH. This is why the characterization by symmetries is so crucial for the flip strategy. It is indeed a fortunate coincidence that the fundamental complexity classes such as $\#P$ and NC have complete functions characterized by their symmetries.

Frequently Asked Questions

Can GCT be used to prove some modest lower bounds first?

Given the difficulty of the fundamental hardness conjectures, one may ask if GCT can be used to prove some modest lower bounds first. That is indeed so. Currently, the best known lower bounds in the context of the P vs. NC and strong permanent vs. determinant problems are both based on GCT. The first lower bound is a special case of the $P \neq NC$ conjecture proved in Mulmuley.¹⁸ It says that the P -complete max-flow problem cannot be solved in polylogarithmic time using polynomially many processors in the PRAM model without bit operations. This model is quite realistic and natural in contrast to the constant depth⁵ or monotone²⁷ circuit models used for proving lower bounds earlier. This lower bound is currently the only known super-polynomial lower bound that is a nontrivial implication of a fundamental separation conjecture like the $P \neq NC$ conjecture and holds unconditionally in a natural and realistic model of computation. Its proof is geometric and quasi-explicit. No combinatorial or elementary proof is known so far. This result was the beginning of the GCT approach to the fundamental hardness conjectures. The second lower bound based on GCT constructions, specifically the varieties $\Delta[\det, m]$ and $\Delta[\text{perm}, n, m]$, is the

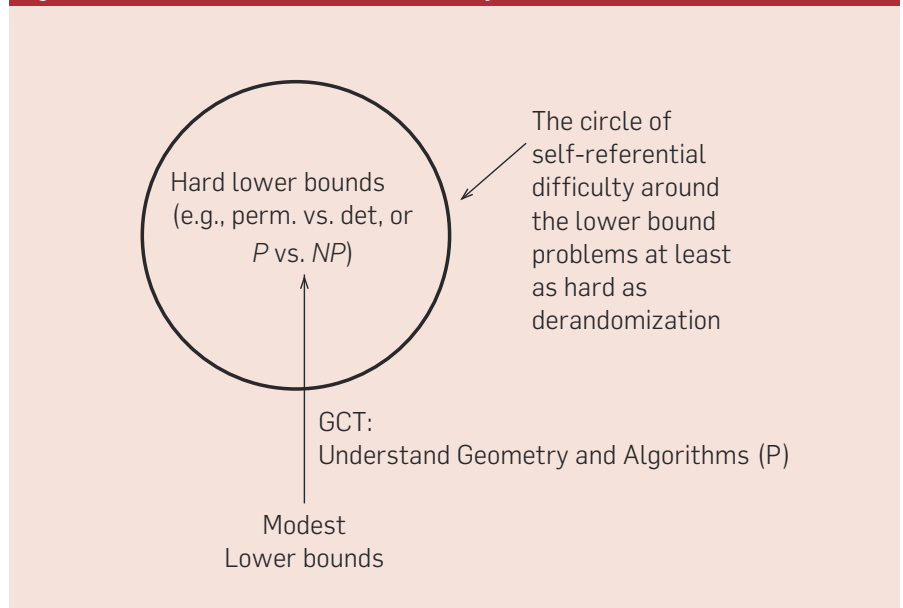
quadratic lower bound¹⁴ stated earlier in the context of the strong permanent vs. determinant conjecture. It is a stronger form of the earlier quadratic lower bound¹⁶ for the usual permanent vs. determinant problem. The proof in Mignon and Ressayre¹⁶ is elementary and does not need GCT. The difference between the strong and usual versions of the permanent vs. determinant problem in Landsberg et al.¹⁴ and Mignon and Ressayre¹⁶ is akin to the difference between the tensor rank and usual versions of the lower bound problem for matrix multiplication.⁶

See also the lower bounds for matrix multiplication based on the fundamental work²⁸ that introduced invariant theory in complexity theory.

Are explicit proofs necessary?

By the strong flip theorem, we know that any proof of the strong permanent vs. determinant conjecture leads to an explicit proof modulo derandomization. This does not say that explicit proofs are necessary. There may be nonexplicit proofs that avoid derandomization altogether. But this does suggest that if derandomization is indeed easier than the fundamental hardness conjectures¹² as the complexity theory suggests, then even such nonexplicit proofs would essentially have the necessary mathematical ingredients to construct proof-certificates of hardness efficiently a posteriori. If so, it makes sense to go toward

Figure 3. Division in the world of lower bounds by the circle of self-reference.



this efficient construction right from the beginning. This allows us to use the theory of algorithms—the main tool of complexity theory—in the study of the fundamental lower bounds. Indeed, it is unrealistic to expect that we can prove $P \neq NP$ without understanding the complexity class P and the theory of algorithms in depth first, as the flip strategy suggests.

The situation here may be compared to that for the well-known four color theorem.² In principle, this theorem may be proved nonconstructively. Yet, the fact remains that all known proofs of this theorem are explicit in the sense that they also yield efficient algorithms for finding a four coloring as a byproduct. The flip theorem suggests that the story of the fundamental hardness conjectures in complexity theory may be similar.

In this sense, these conjectures are fundamentally different from other conjectures in mathematics such as the Riemann Hypothesis. Since there is no analogous flip theorem for the Riemann Hypothesis, it may have a nonconstructive proof that gives no hint on how to test efficiently if the n -th zero of the Riemann zeta function lies on the critical line.

Is algebraic geometry necessary?

According to Valiant,²⁹ the arithmetic permanent vs. determinant conjecture over $\mathbb{Z}$ is implied by the $\#P$ vs. NC conjecture. By the strong flip theorem,^{21,23} stronger forms of the fundamental hardness and derandomization hypotheses in the arithmetic setting imply an analogue of FH in algebraic geometry of comparable difficulty. We have already argued on the basis of these results as to why it is not pragmatic to avoid algebraic geometry, even though it is not formally necessary.

Another concrete evidence for the power of algebraic geometry even in the Boolean setting is provided by the proof of the special case of the $P \neq NC$ conjecture.¹⁸ It has to be emphasized here that, unlike the earlier lower bounds in the algebraic model,⁶ this lower bound is Boolean, not algebraic. This is because it is in terms of the bit length of the input, though the PRAM model in Mulmuley¹⁸ does not allow bit operations. At present, to our knowledge, this is the only nontrivial implication

of a fundamental hardness conjecture that can be proved unconditionally in a natural and realistic model of computation. If we cannot prove even this easier implication of the $P \neq NC$ conjecture by elementary techniques, it seems unrealistic to expect that we can prove the far harder $P \neq NC$ (or NP) conjecture by elementary techniques.

When can we expect a hard lower bound?

The modest lower bounds based on GCT and the earlier modest lower bounds^{3,6} are separated from the fundamental hardness conjectures that are at least as hard as derandomization by the circle of self-referential difficulty, see Figure 3. To break into this circle, we have to show that P contains formidable explicit construction problems in algebraic geometry and representation theory, such as the ones that arise in the strong flip theorem or FH. By the law of conservation of difficulty based on the strong flip theorem, comparable understanding of P is needed in *any* approach. Unfortunately, our current understanding of P is very modest. Until we understand P (the theory of algorithms) and geometry in the required depth, we may not expect any further lower bounds that are fundamentally different from the modest lower bounds discussed previously.

Conclusion

GCT has broken the circle of self-reference around the fundamental hardness conjectures in the arithmetic setting and, in the process, has revealed deep explicit construction and positivity problems at the crossroads of algebraic geometry, representation theory, and complexity theory hidden underneath the fundamental hardness conjectures in complexity theory. Given the formidable nature of these problems, this is undoubtedly only the beginning.

Acknowledgments

The author is grateful to Josh Grochow, Jimmy Qiao, Janos Simon, and the referees for helpful comments. The work on this paper was supported by NSF grant CCF-1017760. 

References

1. Agrawal, M. Proving lower bounds via pseudo-random generators. In *Proceedings of the FSTTCS* (2005), Springer-Verlag, Berlin, Germany, 92–105.

2. Appel, K., Haken, W., Koch, J. Every planar map is four colourable. *Illinois J. Math.* 21 (1977), 439–567.
3. Arora, S., Barak, B. *Computation Complexity: a Modern Approach*, Cambridge University Press, Cambridge, England, 2009.
4. Blasiak, J., Mulmuley, K., Sohoni, M. Geometric complexity theory IV: nonstandard quantum group for the Kronecker problem. *cs. ArXiv preprint cs.CC/0703110* (2011).
5. Boppana, R., Sipser, M. The complexity of finite functions. *Handbook of Theoretical Computer Science*. J. van Leeuwen, ed. Volume A, MIT press, 1990, 757–804.
6. Bürgisser, P., Clausen, M., Shokrollahi, M. *Algebraic Complexity Theory*, Springer-Verlag, 1997.
7. Bürgisser, P. and Ikenmeyer, C. Geometric complexity theory and tensor rank. *arXiv:1011.1350 v1* (2010).
8. Bürgisser, P., Landsberg, J., Manivel, L., Weyman, J. An overview of mathematical issues arising in the geometric complexity theory approach to $VP \neq VNP$. *arXiv:0907.2850v1 [cs.CC]* (2009).
9. Cook, S. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM Symposium on Theory of Computing* (1971), ACM, New York, NY, 151–158.
10. Deligne, P. La conjecture de weil II. *Publ. Math. Inst. Ha. Étud. Sci.* 52 (1980), 137–252.
11. Fortnow, L. The status of the P versus NP problem. *CACM* 52, 9 (2009), 78–86.
12. Kabanets, V., Impagliazzo, R. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.* 13 (2004), 1–46.
13. Karp, R. Reducibility among combinatorial problems. *Complexity of Computer Computations*. R. Miller and J. Thatcher, eds. Plenum Press, New York, 1972, 85–103.
14. Landsberg, J., Manivel, L., Ressayre, N. Hypersurfaces with degenerate duals and the geometric complexity theory program. *arXiv:1004.4802 v1 [math.AG]* (2010).
15. Levin, L. Universal sequential search problems. *Probl. Inform. transm.* 9 (1973), 115–116.
16. Mignon, T., Ressayre, N. A quadratic bound for the determinant and permanent problem. *Int. Math. Res. Not.* 79 (2004), 4241–4253.
17. Mulmuley, K. *Geometric Complexity Theory IX*. Technical report, under preparation.
18. Mulmuley, K. Lower bounds in a parallel model without bit operations. *SIAM J. Comput.* 28, 4 (1999), 1460–1509.
19. Mulmuley, K. *Geometric Complexity Theory VII: Nonstandard Quantum Group for the Plethysm Problem*. Technical Report TR-2007-14, Computer Science Department, The University of Chicago, 2007.
20. Mulmuley, K. *Geometric Complexity Theory VIII: On Canonical Bases for the Nonstandard Quantum Groups*. Technical Report TR 2007-15, Computer Science Department, The University of Chicago, 2007.
21. Mulmuley, K. Explicit proofs and the flip. *arXiv:1009.0246 v1 [cs.CC]* (2010).
22. Mulmuley, K. *Geometric Complexity Theory VI: The Flip via Positivity*. Technical report, The Computer Science Department, The University of Chicago, 2010.
23. Mulmuley, K. On P vs. NP and geometric complexity theory. *JACM* 58, 2 (2011).
24. Mulmuley, K., Narayanan, H., Sohoni, M. Geometric complexity theory III: on deciding nonvanishing of a Littlewood–Richardson coefficient. *J. Algebr. Combinator.* (Nov. 2011), 1–8.
25. Mulmuley, K., Sohoni, M. Geometric complexity theory I: an approach to the P vs. NP and related problems. *SIAM J. Comput.* 31, 2 (2001), 496–526.
26. Mulmuley, K., Sohoni, M. Geometric complexity theory II: towards explicit obstructions for embeddings among class varieties. *SIAM J. Comput.* 38, 3 (2008), 1175–1206.
27. Razborov, A. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR* 281 (1985), 798–801.
28. Strassen, V. Rank and optimal computation of generic tensors. *Lin. Algebra Appl.* 53 (1983), 645–685.
29. Valiant, L. The complexity of computing the permanent. *TCS* 8 (1979), 189–201.
30. Weyl, H. *Classical Groups*, Princeton University Press, Princeton, NJ, 1946.

Ketan D. Mulmuley (mulmuley@cs.uchicago.edu) is a professor of computer science at the University of Chicago, IL.

© 2012 ACM 0001-0782/12/06 \$10.00

THIRD ANNUAL ACM CONFERENCE ON

SYSTEMS, PROGRAMMING, LANGUAGES, APPLICATIONS: SOFTWARE FOR HUMANITY

.....

LOCATION

Loews Ventana Canyon Resort
Tucson, Arizona, USA

EVENTS

- OOPSLA
- Onward!
- Dynamic Languages Symposium,
- Wavefront
- Workshops
- ACM Student Research Competition
- Pattern Languages of Programming
- ...and more!

GENERAL CHAIR

Gary T. Leavens

University of Central Florida
chair@splashcon.org

OOPSLA PROGRAM CHAIR

Matthew B. Dwyer

University of Nebraska
oopsla@splashcon.org

ONWARD! PROGRAM CHAIRS

Jonathan Edwards/Julia Steele

MIT/O'Reilly
onward@splashcon.org
essays@splashcon.org

DLS PROGRAM CHAIR

Alessandro Warth

Google
dls@splashcon.org

FOR MORE INFORMATION

info@splashcon.org
<http://splashcon.org/>

.....

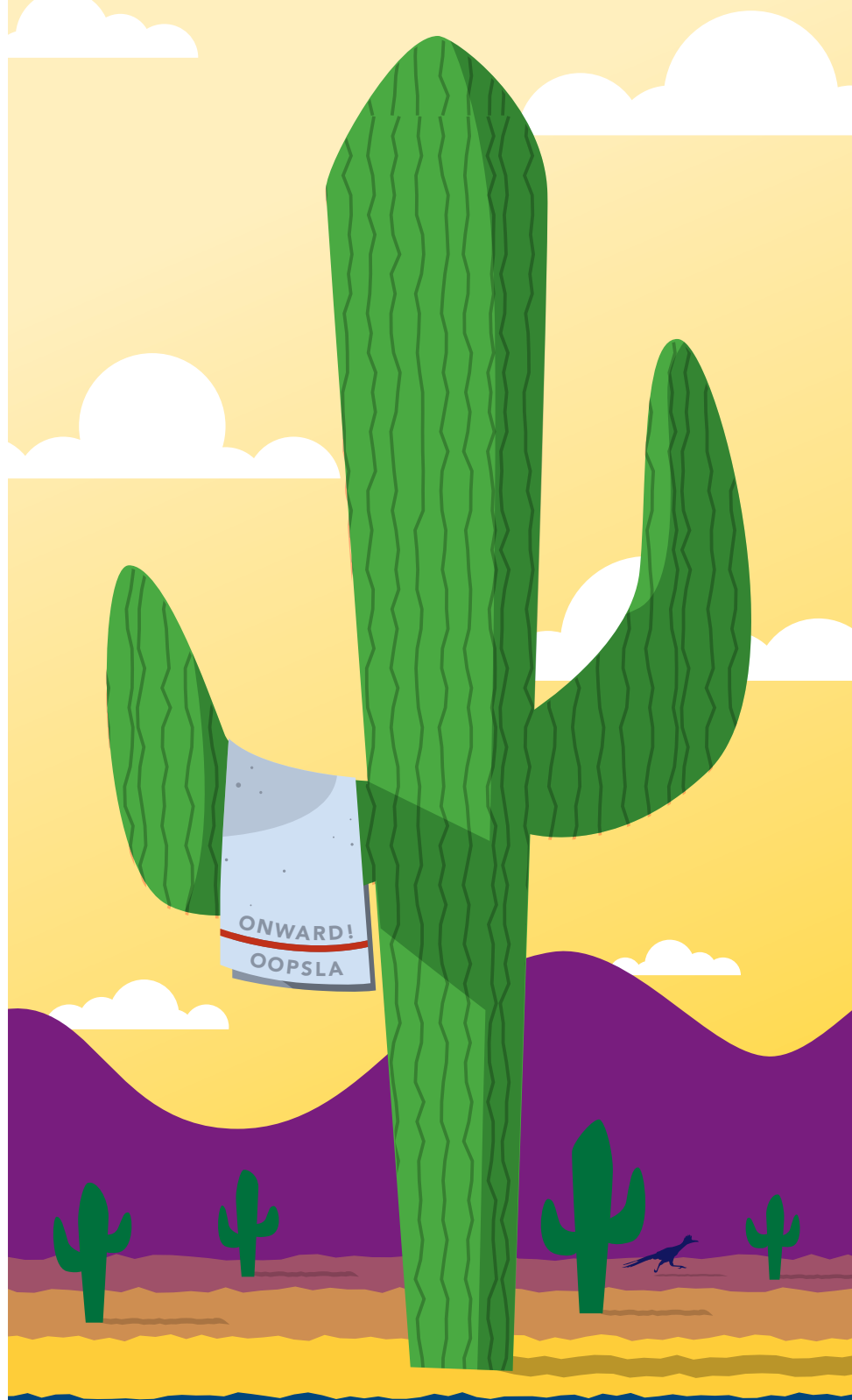
SPLASH is sponsored by
ACM SIGPLAN



SPLASH

TUCSON, ARIZONA

OCTOBER 19-26, 2012



research highlights

P. 110

**Technical
Perspective**
**Reconstructing
the Unknown,
Balancing Structure
and Uncertainty**

By Pablo A. Parrilo

P. 111

**Exact Matrix Completion
via Convex Optimization**

By Emmanuel Candès and Benjamin Recht

P. 120

**Technical
Perspective**
**The Fox and
the Hedgehog**

By Peter Lee

P. 121

**Lightweight Modular Staging:
A Pragmatic Approach to
Runtime Code Generation
and Compiled DSLs**

By Tiark Rompf and Martin Odersky

Technical Perspective

Reconstructing the Unknown, Balancing Structure and Uncertainty

By Pablo A. Parrilo

WE HAVE ALL been there. Late at night, half-sleeping on the couch in front of the TV, opening our eyes every few minutes, catching occasional glimpses of the old movie playing. Waking up, we try to reconstruct and make sense of the plot, reconciling the partial, fragmented pieces of information we have gathered (she stole the money, showers are dangerous, Mother is upstairs) against our preconceived notions (it is a psychological thriller, the blonde actress is the star, twist ending expected). Clearly, there is a tension and synergy between the “partial information” that we have acquired, and the “structured object” that is a traditional movie plot.

Another familiar example of this situation is given by constraint satisfaction puzzles such as Sudoku. Here, the partial information is given by the numbers that are initially revealed, and the structured object is the unknown, final completed pattern satisfying all the constraints. Solving Sudoku, or understanding what the movie was about, is the process of reconciling what we have seen against our priors.

As in these examples, the problem of estimating or reconstructing an *unknown structured object* from *incomplete, partial, noisy measurements* is a fundamental one in scientific and technological applications. Many aspects of signal processing, imaging, system identification, and statistical inference can be fruitfully understood from this perspective. For instance, consider understanding a speech signal, reconstructing a real-world image, or identifying a dynamical model of a robotic arm. In all these cases, we have good priors about what the objects *should* be, but also have empirical information about the concrete instance we are interested in.

The beautiful and foundational work of Candès and Recht on matrix completion addresses an interesting

and important situation of this kind, where the unknown object is a *low-rank matrix*, and the “measurements” correspond to a *random subset of its entries*. Besides the mathematical simplicity and elegance of the model, what makes this problem formulation remarkable is that it appears quite naturally in a number of important applications, including many tasks related to machine learning and collaborative filtering such as the celebrated Netflix problem.

What often makes these problems challenging is the subtle interaction between *a priori information* and *data*. In the existing literature, many different situations have been studied, which can be roughly classified as either *probabilistic* or *set-based* assumptions. In particular, the popular “compressed sensing” techniques correspond to *sparsity* assumptions on the unknown object, and have received much research attention in recent years. The low-rank models studied in this paper naturally generalize many of these ideas to the matrix setting. In combination with sparsity techniques, they have proven to be remarkably flexible and powerful for modeling complex

statistical and dynamical phenomena.

The approach to matrix completion followed in this paper uses *convex optimization* in an essential way. The nuclear norm of a matrix (the sum of its singular values) is a convex function that is intimately related to its low-rank properties. Replacing a difficult, non-convex rank minimization problem with a better-behaved convex surrogate enables the use of the well-developed machinery of numerical convex optimization to produce candidate solutions. However, the main question remains: how well does this work, if it indeed does? How many entries of the matrix do we need to observe, if we hope to exactly reconstruct the matrix? As the authors show, a surprisingly small number of entries are required to completely recover the unknown matrix with high probability.

There are a number of important technical innovations in this beautiful paper that have created much research excitement and set the stage for numerous follow-up works. In particular, the notion of “incoherence” of the row and column spaces of the matrix has proved fundamental to ensure the observed random samples are “representative enough,” and nicely generalizes earlier concepts from the compressed sensing literature. The probabilistic analysis techniques, relying on subtle duality arguments and concentration inequalities, have provided a template for a number of similar recovery results.

Like many great papers, this one not only answers an interesting question but also raises several new ones: What are the most general “structured objects” that we can efficiently recover? What are the fundamental limitations of convex optimization techniques? Is it possible to relax the convexity requirements, while keeping tractability? To what extent can these results be generalized to richer and more powerful models of structure and uncertainty? Whatever the answers to these questions may be, it is clear the path to their solution has been illuminated by this insightful piece of work. 

There are a number of important technical innovations in this beautiful paper that have created much research excitement and set the stage for numerous follow-up works.

Pablo A. Parrilo (parrilo@mit.edu) is a professor of electrical engineering and computer science at the Massachusetts Institute of Technology, Cambridge, MA.

© 2012 ACM 0001-0782/12/06 \$10.00

Exact Matrix Completion via Convex Optimization

By Emmanuel Candès and Benjamin Recht

Abstract

Suppose that one observes an incomplete subset of entries selected from a low-rank matrix. When is it possible to complete the matrix and recover the entries that have not been seen? We demonstrate that in very general settings, one can perfectly recover all of the missing entries from most sufficiently large subsets by solving a convex programming problem that finds the matrix with the minimum nuclear norm agreeing with the observed entries. The techniques used in this analysis draw upon parallels in the field of compressed sensing, demonstrating that objects other than signals and images can be perfectly reconstructed from very limited information.

1. INTRODUCTION

In many practical problems of interest, one would like to recover a matrix from a sampling of its entries. As a motivating example, consider the task of inferring answers in a partially filled out survey in which questions are asked to a collection of individuals. Then we can form a matrix where the rows index the individuals and the columns index the questions. We collect data to fill out this table, but unfortunately, many questions are left unanswered. Is it possible to make an educated guess about what the missing answers should be? How can one make such a guess? Formally, we may view this problem as follows. We are interested in recovering a data matrix $\mathbf{M}$ with n_1 rows and n_2 columns but have access to only m of its entries, where m is much smaller than the total number of entries, $n_1 n_2$. Can one recover the matrix $\mathbf{M}$ from m of its entries? In general, everyone would agree that this is impossible without some additional information.

In many instances, however, the matrix we wish to recover is known to be structured in the sense that it is low-rank or approximately low-rank. (We recall for completeness that a matrix has rank r if its rows or columns span an r -dimensional space.) Consider the following two scenarios as prototypical examples.

► *The Netflix problem.* In the area of recommender systems, users submit ratings on a subset of entries in a database, and the vendor provides recommendations based on the user's preferences.³¹ Because users only rate a few items, one would like to infer their preference for unrated items. A special instance of this problem is the now famous Netflix problem.²⁴ Users (rows of the data matrix) are given the opportunity to rate movies (columns of the data matrix), but users typically rate only very few movies so that there are very few scattered observed entries of this data matrix. Yet, one would like to complete this matrix so that

the vendor (here Netflix) might recommend titles that any particular user is likely to be willing to order. In this case, the data matrix of all user-ratings may be approximately low-rank, because only a few factors contribute to an individual's tastes or preferences.

► *Triangulation from incomplete data.* Suppose we are given partial information about the distances between objects and would like to reconstruct the low-dimensional geometry describing their locations. For example, we may have a network of low-power, wirelessly networked sensors scattered randomly across a region. Suppose each sensor only has the ability to construct distance estimates based on signal strength readings from its nearest fellow sensors. From these local distance estimates, we can form a partially observed distance matrix. We can then estimate the true distance matrix whose rank will be equal to 2 if the sensors are located in a plane or 3 if they are located in three-dimensional space.^{26, 32} In this case, we only need to observe a few distances per node to have enough information to reconstruct the positions of the objects.

These examples are of course far from exhaustive and there are many other problems which fall in this general category.

Suppose for simplicity that we wish to recover a square $n \times n$ matrix $\mathbf{M}$ of rank r . Although $\mathbf{M}$ contains n^2 numbers, our assumption that its rank is r means that it can be represented exactly by its singular value decomposition (SVD)

$$\mathbf{M} = \mathbf{U} \Sigma \mathbf{V}^T = \sum_{k=1}^r \sigma_k \mathbf{u}_k \mathbf{v}_k^T \quad (1.1)$$

where $\mathbf{V}^T$ denotes the transpose of $\mathbf{V}$. Σ is an $r \times r$ diagonal matrix with real, positive elements $\sigma_k > 0$. $\mathbf{U}$ is an $n \times r$ matrix with orthonormal columns $\mathbf{u}_1, \dots, \mathbf{u}_r$. That is, $\mathbf{u}_k^T \mathbf{u}_k = 1$ and $\mathbf{u}_i^T \mathbf{u}_j = 0$ if $i \neq j$. $\mathbf{V}$ is also $n \times r$ with orthonormal columns $\mathbf{v}_1, \dots, \mathbf{v}_r$. The column space of $\mathbf{M}$ is spanned by the columns of $\mathbf{U}$, and the row space is spanned by the columns of $\mathbf{V}$.

The number of degrees of freedom associated with a rank r matrix $\mathbf{M}$ is $r(2n - r)$. To see this, note that Σ has r nonzero entries, and $\mathbf{U}$ and $\mathbf{V}$ each have nr total entries. Since $\mathbf{U}$ and $\mathbf{V}$ each satisfy $r(r + 1)/2$ orthogonality constraints, the total number of degrees of freedom is $r + 2nr - r(r + 1) = r(2n - r)$. Thus, when r is much smaller than n , there are significantly

The original version of this paper was published in *Foundations of Computational Mathematics* 9, 6 (2009), 717–772.

fewer degrees of freedom than the size of $\mathbf{M}$ would suggest. The question is then whether $\mathbf{M}$ can be recovered from a suitably chosen sampling of its entries without collecting n^2 measurements.

In this paper, we demonstrate that most low-rank matrices can be indeed recovered from a very sparse sampling of their entries. In Section 2, we summarize the main results of our paper, highlighting the necessary assumptions, algorithmic ingredients, and theoretical foundations of reconstructing matrices from a presented collection of entries. In Section 3, we survey the subsequent developments in this area, including refinements and important extensions of our theory. We close with a discussion of further progress and advances in low-rank and sparse modeling.

2. MATRIX COMPLETION

Which matrices?

In general, one cannot hope to be able to recover a low-rank matrix from a sample of its entries. Consider the rank 1 matrix $\mathbf{M}$ equal to

$$\mathbf{M} = \mathbf{e}_1 \mathbf{x}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_{n-1} & x_n \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad (2.1)$$

where here and throughout, $\mathbf{e}_i$ is the i th canonical basis vector in Euclidean space (the vector with all entries equal to 0 but the i th equal to 1). The matrix $\mathbf{M}$ has the entries of $\mathbf{x}$ along its first row and all the other entries are 0. Clearly, this matrix cannot be recovered from a sampling of its entries unless we see *all* of the entries in the first row. As another example, the matrix $\mathbf{e}_1 \mathbf{e}_n^T$ is a matrix with a 1 in the $(1, n)$ entry and 0s everywhere else. If we do not see this upper right corner, then we cannot distinguish the matrix from the all 0s matrix.

Even if it is impossible to recover *all* low-rank matrices from a set of sampled entries, can one recover *most* of them? To investigate this possibility, we introduce a simple model of low-rank matrices.

DEFINITION 2.1. *Let $\mathbf{M}$ be a rank r matrix with SVD defined by (1.1). Then we say that $\mathbf{M}$ belongs to the random orthogonal model if the family $\{\mathbf{u}_k\}_{1 \leq k \leq r}$ is selected uniformly at random among all families of r orthonormal vectors, and similarly for $\{\mathbf{v}_k\}_{1 \leq k \leq r}$. The two families may or may not be independent of each other. We make no assumptions about the singular values, σ_k .*

If a matrix is sampled from the random orthogonal model, then we would expect most of the entries to be non-zero. This model is convenient in the sense that it is both very concrete and simple, and useful in the sense that it will help us fix the main ideas. In the sequel, however, we will consider far more general models. The question for now is whether or not one can recover such a generic matrix from a sampling of its entries.

Which sampling sets?

Clearly, one cannot hope to reconstruct any low-rank matrix $\mathbf{M}$ —even of rank 1—if the sampling set avoids any column or row of $\mathbf{M}$. Suppose that $\mathbf{M}$ is of rank 1 and of the form $\mathbf{x}\mathbf{y}^T$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ so that the (i, j) entry is given by $M_{ij} = x_i y_j$. Then, if we do not have samples from the first row, one could never infer the value of the first component x_1 as no information about x_1 is observed. There is, of course, nothing special about the first row and this argument extends to any row or column. To have any hope of recovering an unknown matrix, one needs to have access to at least one observation per row and one observation per column.

This example demonstrates that there are sampling sets where one would not even be able to recover matrices of rank 1. But what happens for typical sampling sets? Can one recover a low-rank matrix from almost all sampling sets of cardinality m ? Formally, suppose that the set Ω of locations corresponding to the observed entries ($(i, j) \in \Omega$ if M_{ij} is observed) is a set of cardinality m sampled uniformly at random. Then, can one recover a generic low-rank matrix $\mathbf{M}$, perhaps with very large probability, from the knowledge of the value of its entries in the set Ω ?

Which algorithm?

If the number of measurements is sufficiently large, and if the entries are close to uniformly distributed, one might hope that there is only *one* low-rank matrix with these entries. If this were true, one would want to recover the data matrix by solving the optimization problem

$$\begin{aligned} & \text{minimize} && \text{rank}(\mathbf{X}) \\ & \text{subject to} && X_{ij} = M_{ij} \quad (i, j) \in \Omega \\ & && \mathbf{X} \in \mathbb{R}^{n \times n}, \end{aligned} \quad (2.2)$$

where $\mathbf{X}$ is the decision variable and $\text{rank}(\mathbf{X})$ is equal to the rank of the matrix $\mathbf{X}$. The program (2.2) is a common sense approach which simply seeks the simplest explanation fitting the observed data. If there were only one low-rank object fitting the data, the solution of (2.2) would recover $\mathbf{M}$ perfectly. This is unfortunately of little practical use, because not only is this optimization problem NP-hard but also all known algorithms which provide exact solutions require time doubly exponential in the dimension n of the matrix in both theory and practice.

If a matrix has rank r , then it has exactly r nonzero singular values so that the rank function in (2.2) is simply the number of nonvanishing singular values. In this paper, we consider an alternative which minimizes the sum of the singular values over the constraint set. This sum is called the *nuclear norm*,

$$\|\mathbf{X}\|_* = \sum_{k=1}^n \sigma_k(\mathbf{X}),$$

where, here and below, $\sigma_k(\mathbf{X})$ denotes the k th largest singular value of $\mathbf{X}$. The heuristic optimization we study is then given by

$$\begin{aligned} & \text{minimize} && \|\mathbf{X}\|_* \\ & \text{subject to} && X_{ij} = M_{ij} \quad (i, j) \in \Omega. \end{aligned} \quad (2.3)$$

Whereas the rank function is equal to the number of non-vanishing singular values, the nuclear norm equals their sum. The nuclear norm is to the rank functional what the convex ℓ_1 norm is to the ℓ_0 norm in the area of sparse signal recovery. The main point here is that the nuclear norm is a convex function and can be optimized efficiently via semi-definite programming.¹⁴

There are many norms one could define for a given matrix. The *operator norm* is the largest singular value. The *Frobenius norm* is equal to the square root of the sum of the squares of the entries. This norm is akin to the standard Euclidean norm on a real vector space. Why should the nuclear norm provide lower rank solutions than either of these two more commonly studied norms?

One can gain further intuition by analyzing the geometric structure of the nuclear norm ball. The unit nuclear norm ball is precisely the convex hull of the rank 1 matrices of unit Frobenius norm. The nuclear norm minimization problem (2.3) can be interpreted as inflating the unit ball until it just touches the affine space $X_{ij} = M_{ij}$. Such an intersection will occur at an extreme point of the nuclear norm ball, and these extreme points are sparse convex combinations of rank 1 matrices. That is, the extreme points of the nuclear norm ball have low rank. This phenomenon is depicted graphically in Figure 1. There, we plot the unit ball of the nuclear norm for matrices parametrized as

$$\mathbf{X} = \begin{bmatrix} x & y \\ y & z \end{bmatrix}.$$

The extreme points of this cylindrical object are the rank 1 matrices with unit Frobenius norm. The red line in this figure is a “random,” one-dimensional, affine subspace which, as expected, intersects the nuclear norm ball at a rank 1 matrix.

As further motivation, an interesting connection exists between the nuclear norm and popular algorithms in

data-mining and collaborative filtering. In these fields, researchers commonly aim to find an explicit factorization $\mathbf{X} = \mathbf{L}\mathbf{R}^T$ that agrees with the measured entries. Here $\mathbf{L}$ and $\mathbf{R}$ are $n \times k$ matrices. Since there are many possible such factorizations that might agree with the observations, a common approach searches for matrices $\mathbf{L}$ and $\mathbf{R}$ that have Frobenius norm as small as possible, that is, the solution of the optimization problem

$$\begin{aligned} &\text{minimize} && \frac{1}{2}(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2) \\ &\text{subject to} && \mathbf{L}\mathbf{R}^T = \mathbf{X} \\ &&& X_{ij} = M_{ij} \quad (i, j) \in \Omega, \end{aligned} \quad (2.4)$$

where we are minimizing with respect to $\mathbf{L} \in \mathbb{R}^{n \times k}$, $\mathbf{R} \in \mathbb{R}^{n \times k}$, and $\mathbf{X} \in \mathbb{R}^{n \times n}$, and $\|\cdot\|_F$ denotes the Frobenius norm. Surprisingly, the optimization problem (2.4) is equivalent to minimization of the nuclear norm subject to the same equality constraints provided k is chosen to be larger than the rank of the optimum of the nuclear norm problem (2.3).³⁰

To get an intuition for this equivalence, take any matrix $\mathbf{X}$ of rank k . Suppose the SVD is $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$. If we set $\mathbf{L} := \mathbf{U}\Sigma^{\frac{1}{2}}$ and $\mathbf{R} := \mathbf{V}\Sigma^{\frac{1}{2}}$, we see that

$$\frac{1}{2}\|\mathbf{L}\|_F^2 + \frac{1}{2}\|\mathbf{R}\|_F^2 = \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^k (U_{ij}^2 \sigma_j + V_{ij}^2 \sigma_j) = \sum_{j=1}^k \sigma_j = \|\mathbf{X}\|_*,$$

because $\sum_i U_{ij}^2 = \sum_i V_{ij}^2 = 1$ for all j . Thus, the optimal solution of (2.3) is suboptimal for (2.4). The full equivalence can be seen via an appeal to semidefinite programming and can be found in Recht et al.³⁰

The main advantage of this reformulation (2.4) is to substantially decrease the number of decision variables from n^2 to $2nr$. For large problems, this leads to a significant reduction in computation time, such that very large instances can be solved on a desktop computer. On the other hand, the formulation (2.4) is nonconvex and thus potentially has local minima that are not globally optimal. Nonetheless, this factored approximation (2.4) of the nuclear norm is one of the most successful stand-alone approaches to solving the Netflix Prize problem.^{16,24} Indeed, it was one of the foundational components of the winning team’s prediction engine.

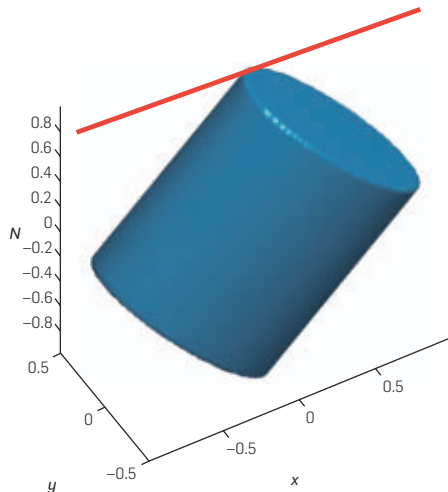
2.1. Main results

As seen in our first example (2.1), it is impossible to recover a matrix which is equal to 0 in nearly all of its entries unless we see all the entries of the matrix. This is particularly likely if the singular vectors of a matrix $\mathbf{M}$ have most of their mass concentrated in a few coordinates. For instance, consider the rank 2 symmetric matrix $\mathbf{M}$ given by

$$\mathbf{M} = \sum_{k=1}^2 \sigma_k \mathbf{u}_k \mathbf{u}_k^T, \quad \begin{aligned} \mathbf{u}_1 &= (\mathbf{e}_1 + \mathbf{e}_2)/\sqrt{2}, \\ \mathbf{u}_2 &= (\mathbf{e}_1 - \mathbf{e}_2)/\sqrt{2}, \end{aligned}$$

where the singular values are arbitrary. Then, this matrix vanishes everywhere except in the top-left 2×2 corner, and one would basically need to see all the entries of $\mathbf{M}$ to be able to recover this matrix exactly. There is an endless list of examples of this sort. Hence, we arrive at the notion that the singular vectors need to be sufficiently spread across

Figure 1. Unit ball of the nuclear norm for symmetric 2×2 matrices. The red line depicts a random one-dimensional affine space. Such a subspace will generically intersect a sufficiently large nuclear norm ball at a rank one matrix.



all components—that is, uncorrelated with the standard basis—in order to minimize the number of observations needed to recover a low-rank matrix. This motivates the following definition.

DEFINITION 2.2. Let U be a subspace of $\mathbb{R}^n$ of dimension r and $\mathbf{P}_U$ be the orthogonal projection onto U . Then the coherence of U (vis-à-vis the standard basis $(\mathbf{e}_i)$) is defined to be

$$\mu(U) \equiv \frac{n}{r} \max_{1 \leq i \leq n} \|\mathbf{P}_U \mathbf{e}_i\|^2.$$

Note that for any subspace, the smallest $\mu(U)$ can be is 1, achieved, for example, if U is spanned by vectors whose entries all have magnitude $1/\sqrt{n}$. The largest possible value for $\mu(U)$ is n/r which would correspond to any subspace that contains a standard basis element. Matrices whose column and row spaces have low coherence are likely not to vanish in too many entries and are our most likely candidates for matrices that are recoverable from a few samples. As we discuss below, subspaces sampled from the random orthogonal model (Definition 2.1) have nearly minimal coherence.

To state our main result, we introduce two assumptions about an $n_1 \times n_2$, rank r matrix $\mathbf{M}$ whose SVD is given by (1.1) and with column and row spaces denoted by U and V , respectively.

A0 The coherences obey $\max(\mu(U), \mu(V)) \leq \mu_0$ for some $\mu_0 > 0$.

A1 The $n_1 \times n_2$ matrix $\sum_{1 \leq k \leq r} \mathbf{u}_k \mathbf{v}_k^T$ has a maximum entry bounded by $\mu_1 \sqrt{r/(n_1 n_2)}$ in absolute value for some $\mu_1 > 0$.

These definitions implicitly define two critical parameters, μ_0 and μ_1 . These μ 's may depend on r and n_1, n_2 . Moreover, note that **A1** always holds with $\mu_1 = \mu_0 \sqrt{r}$ since the (i, j) th entry of the matrix $\sum_{1 \leq k \leq r} \mathbf{u}_k \mathbf{v}_k^T$ is given by $\sum_{1 \leq k \leq r} u_{ik} v_{jk}$ and by the Cauchy–Schwarz inequality,

$$\left| \sum_{1 \leq k \leq r} u_{ik} v_{jk} \right| \leq \sqrt{\sum_{1 \leq k \leq r} |u_{ik}|^2} \sqrt{\sum_{1 \leq k \leq r} |v_{jk}|^2} \leq \frac{\mu_0 r}{\sqrt{n_1 n_2}}.$$

Hence, for sufficiently small ranks, μ_1 is comparable to μ_0 . We say that a subspace $U \subset \mathbb{R}^n$ is *incoherent* with the standard basis if $\mu(U)$ is at most logarithmic in n . As we show in the full version of this paper that, for larger ranks, both subspaces selected from the uniform distribution and spaces constructed as the span of singular vectors with bounded entries are not only incoherent with the standard basis but also obey **A1** with high probability for values of μ_1 at most logarithmic in n_1 and/or n_2 .

We are now in a position to state our main result: if a matrix has row and column spaces that are incoherent with the standard basis, then nuclear norm minimization can recover this matrix from a random sampling of a small number of entries.

THEOREM 2.3. Let $\mathbf{M}$ be an $n_1 \times n_2$ matrix of rank r obeying **A0** and **A1** and put $n = \max(n_1, n_2)$. Suppose we observe m entries of $\mathbf{M}$ with locations sampled uniformly at random. Then there exist constants C, c such that if

$$m \geq C \max(\mu_1^2, \mu_0^{1/2} \mu_1, \mu_0 n^{1/4}) nr (\beta \log n)$$

for some $\beta > 2$, then the minimizer to the problem (2.3) is unique and equal to $\mathbf{M}$ with probability at least $1 - cn^{-\beta}$. For $r \leq \mu_0^{-1} n^{1/5}$ this estimate can be improved to

$$m \geq C \mu_0 n^{6/5} r (\beta \log n)$$

with the same probability of success.

Theorem 2.3, proven in the full version of this paper, asserts that if the coherence is low, few samples are required to recover $\mathbf{M}$. For example, if μ_0 is a small constant and the rank is not too large, then the recovery is exact with large probability provided that

$$m \geq C n^{6/5} r \log n. \quad (2.5)$$

We give two illustrative examples of matrices with incoherent column and row spaces. This list is by no means exhaustive.

1. The first example is the random orthogonal model (see Definition 2.1). For values of the rank r greater than $\log n$, $\mu(U)$ and $\mu(V)$ are $O(1)$, $\mu_1 = O(\log n)$ both with very large probability. Hence, the recovery is exact on most sampling sets provided that $m \leq C n^{5/4} r \log n$. When $r \leq n^{1/5}$, we can strengthen this bound to $m \leq C n^{6/5} r \log n$.
2. The second example is more general and simply requires that the components of the singular vectors of $\mathbf{M}$ are small. Assume that the $\mathbf{u}_j$ and $\mathbf{v}_j$'s obey

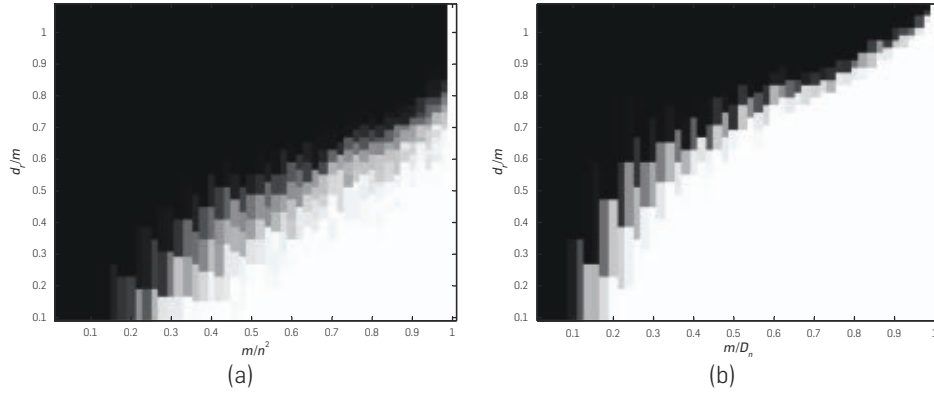
$$\max_{ij} |\langle \mathbf{e}_i, \mathbf{u}_j \rangle|^2 \leq \mu_b / n, \quad \max_{ij} |\langle \mathbf{e}_i, \mathbf{v}_j \rangle|^2 \leq \mu_b / n, \quad (2.6)$$

for some value of $\mu_b = O(1)$. Then, the maximum coherence is at most μ_b since $\mu(U) \leq \mu_b$ and $\mu(V) \leq \mu_b$. Further, we show in the full version of this paper that **A1** holds most of the time with $\mu_1 = O(\sqrt{\log n})$. Thus, for matrices with singular vectors obeying (2.6), the recovery is exact provided that m obeys (2.5) for values of the rank not exceeding $\mu_b^{-1} n^{1/5}$.

2.2. Numerical validation

To demonstrate the practical applicability of the nuclear norm heuristic for recovering low-rank matrices from their entries, we conducted a series of numerical experiments for a variety of the matrix sizes n , ranks r , and numbers of entries m . For each (n, m, r) triple, we repeated the following procedure 50 times. We generated $\mathbf{M}$, an $n \times n$ matrix of rank r , by sampling two $n \times r$ factors $\mathbf{M}_L$ and $\mathbf{M}_R$ with i.i.d. Gaussian entries and setting $\mathbf{M} = \mathbf{M}_L \mathbf{M}_R^T$. We sampled a subset Ω of m entries uniformly at random. Then, the nuclear norm minimization problem was solved using the semi-definite programming solver, SeDuMi.³³ We declared $\mathbf{M}$ to be recovered if the solution returned by the solver, $\mathbf{X}_{\text{opt}}$, satisfied $\|\mathbf{X}_{\text{opt}} - \mathbf{M}\|_F / \|\mathbf{M}\|_F < 10^{-3}$. Figure 2 shows the results of these experiments for $n = 50$. The x -axis corresponds to the fraction of the entries of the matrix that are revealed to the SDP solver. The y -axis corresponds to the ratio between the

Figure 2. Recovery of full matrices from their entries. For each (n, m, r) triple, we repeated the following procedure 50 times. A matrix $\mathbf{M}$ of rank r and a subset of m entries were selected at random. Then, we solved the nuclear norm minimization for $\mathbf{X}$ subject to $X_{ij} = M_{ij}$ on the selected entries. We declared $\mathbf{M}$ to be recovered if $\|\mathbf{X}_{\text{opt}} - \mathbf{M}\|_F / \|\mathbf{M}\|_F < 10^{-3}$. The results are shown for (a) general 50×50 matrices (b) 50×50 positive definite matrices. The color of each cell reflects the empirical recovery rate (scaled between 0 and 1). White denotes perfect recovery in all experiments, and black denotes failure for all experiments.



dimension of the rank r matrices, $d_r = r(2n - r)$, and the number of measurements m .

Note that the axes range from 0 to 1 as a value > 1 on the x -axis corresponds to an overdetermined linear system where the semidefinite program always succeeds, and a value > 1 on the y -axis corresponds to when there are an infinite number of rank r matrices with the provided entries. The color of each cell in the figures reflects the empirical recovery rate of the 50 runs (scaled between 0 and 1). Interestingly, the experiments reveal very similar plots for different n , suggesting that our theoretical upper bounds on recovery may be rather conservative.

For a second experiment, we generated random *positive semidefinite* matrices and tried to recover them from their entries using the nuclear norm heuristic. As above, we repeated the same procedure 50 times for each (n, m, r) triple. We generated $\mathbf{M}$, an $n \times n$ positive semidefinite matrix of rank r , by sampling an $n \times r$ factor $\mathbf{M}_F$ with i.i.d. Gaussian entries and setting $\mathbf{M} = \mathbf{M}_F \mathbf{M}_F^T$. We sampled a subset Ω of m entries uniformly at random. Then, we solved the nuclear norm minimization problem with an additional constraint that the decision variable be positive definite. Figure 2(b) shows the results of these experiments for $n = 50$. The x -axis again corresponds to the fraction of the entries of the matrix that are revealed to the solver, but, in this case, the number of measurements is divided by $D_n = n(n + 1)/2$, the number of unique entries in a positive-semidefinite matrix, and the dimension of the rank r matrices is $d_r = nr - r(r - 1)/2$. The color of each cell is chosen in the same fashion as in the experiment with full matrices. Interestingly, the recovery region is much larger for positive semidefinite matrices, and future work is needed to investigate if the theoretical scaling is also more favorable in this scenario of low-rank matrix completion.

These phase transition diagrams reveal a considerably smaller region of parameter space than the Gaussian models studied in Recht et al.³⁰ In the experiments in Recht et al.,³⁰ $\mathbf{M}$ was generated in the same fashion as above, but, in the

place of sampling entries, we generated m random Gaussian projections of the data (see the discussion in Section 2.4). In these experiments, the recovery regime is far larger than that in the case of sampling entries, but this is not particularly surprising as each Gaussian observation measures a contribution from every entry in the matrix $\mathbf{M}$.

2.3. More general bases

Our main result (Theorem 2.3) extends to a variety of other low-rank matrix completion problems beyond the sampling of entries. Indeed, suppose we have two orthonormal bases $\mathbf{f}_1, \dots, \mathbf{f}_n$ and $\mathbf{g}_1, \dots, \mathbf{g}_n$ of $\mathbb{R}^n$, and that we are interested in solving the rank minimization problem

$$\begin{aligned} & \text{minimize} && \text{rank}(\mathbf{X}) \\ & \text{subject to} && \mathbf{f}_i^T \mathbf{X} \mathbf{g}_j = \mathbf{f}_i^T \mathbf{M} \mathbf{g}_j, \quad (i, j) \in \Omega. \end{aligned} \quad (2.7)$$

The machine learning community's interest in specialized algorithms for *multiclass* and *multitask* learning provides a motivating example (see, e.g., Amit et al.¹ and Argyriou et al.²). In multiclass learning, the goal is to build multiple classifiers with the same training data to distinguish between more than two categories. For example, in face recognition, one might want to classify whether an image patch corresponds to an eye, nose, or mouth. In multitask learning, we have a large set of data and a variety of different classification tasks, but, for each task, only partial subsets of the data are relevant. For instance, in activity recognition, we may have acquired sets of observations of multiple subjects and want to determine if each observed person is walking or running. However, a different classifier is desired for each individual, and it is not clear how having access to the full collection of observations can improve classification performance. Multitask learning aims to take advantage of access to the full database to improve performance on individual tasks. A description of how to apply our results to the multiclass setting can be

found in the full version of this paper.

To see that our theorem provides conditions under which (2.7) can be solved via nuclear norm minimization, note that there exist unitary transformations $\mathbf{F}$ and $\mathbf{G}$ such that $\mathbf{e}_j = \mathbf{F}\mathbf{f}_j$ and $\mathbf{e}_j = \mathbf{G}\mathbf{g}_j$ for each $j = 1, \dots, n$. Hence,

$$\mathbf{f}_i^T \mathbf{X} \mathbf{g}_j = \mathbf{e}_i^T (\mathbf{F} \mathbf{X} \mathbf{G}^T) \mathbf{e}_j.$$

Then, if the conditions of Theorem 2.3 hold for the matrix $\mathbf{F} \mathbf{X} \mathbf{G}^T$, it is immediate that nuclear norm minimization finds the unique optimal solution of (2.7) when we are provided a large enough random collection of the inner products $\mathbf{f}_i^T \mathbf{M} \mathbf{g}_j$. In other words, all that is needed is that the column and row spaces of $\mathbf{M}$ be, respectively, incoherent with the bases $(\mathbf{f}_i)$ and $(\mathbf{g}_j)$.

2.4. Connections, alternatives, and prior art

Nuclear norm minimization is a recent heuristic introduced by Fazel¹⁴ and is an extension of the trace heuristic often used in control theory; see, for example, Beck and D'Andrea.³ Indeed, when the matrix variable is symmetric and positive semidefinite, the nuclear norm of $\mathbf{X}$ is the sum of the (nonnegative) eigenvalues and thus equal to the trace of $\mathbf{X}$. Hence, for positive semidefinite unknowns, $\mathbf{X}$, (2.3) becomes the semidefinite program

$$\begin{array}{ll} \text{minimize} & \text{trace}(\mathbf{X}) \\ \text{subject to} & X_{ij} = M_{ij} \quad (i, j) \in \Omega \\ & \mathbf{X} \succeq 0. \end{array}$$

Even for the general matrix $\mathbf{M}$, which may not be positive definite or even symmetric, the nuclear norm heuristic can be formulated in terms of semidefinite programming. The program (2.3) is equivalent to

$$\begin{array}{ll} \text{minimize} & \text{trace}(\mathbf{W}_1) + \text{trace}(\mathbf{W}_2) \\ \text{subject to} & X_{ij} = M_{ij} \quad (i, j) \in \Omega \\ & \begin{bmatrix} \mathbf{W}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{W}_2 \end{bmatrix} \succeq 0, \end{array}$$

with optimization variables $\mathbf{X}$, $\mathbf{W}_1$, and $\mathbf{W}_2$ (see, e.g., Fazel¹⁴). There are many efficient algorithms and high-quality software packages available for solving these types of problems.

Our work is inspired by results in the emerging field of *compressive sampling* or *compressed sensing*, a new paradigm for acquiring information about objects of interest from what appears to be a highly incomplete set of measurements.^{8, 13} In practice, this means that high-resolution images can be captured with fewer sensors or that signal acquisition can be accelerated by orders of magnitude in biomedical applications, simply by taking far fewer specially coded samples. Mathematically speaking, we wish to reconstruct a signal $\mathbf{x} \in \mathbb{R}^n$ from a small number of measurements $\mathbf{y} = \Phi \mathbf{x}$, $\mathbf{y} \in \mathbb{R}^m$ with m much smaller than n ; that is, we have far fewer equations than unknowns. In general,

one cannot hope to reconstruct $\mathbf{x}$ but assume now that the object we wish to recover is known to be structured in the sense that it is sparse (or approximately sparse). This means that the unknown object depends upon a smaller number of unknown parameters. Then, it has been shown that ℓ_1 minimization—minimizing the sum of the absolute values of $\mathbf{x}$, subject to the linear constraints $\mathbf{y} = \Phi \mathbf{x}$ —allows recovery of sparse signals from remarkably few measurements.⁸ If Φ is chosen randomly from a suitable distribution, then with very high probability, all sparse signals with about k nonzero entries can be recovered from on the order of $k \log n$ measurements. For instance, if $\mathbf{x}$ is k -sparse in the Fourier domain, that is, $\mathbf{x}$ is a superposition of k sinusoids, then it can be perfectly recovered with high probability—by ℓ_1 minimization—from the knowledge of about $k \log n$ of its entries sampled uniformly at random.

From this viewpoint, the results in this paper greatly extend the theory of compressed sensing by showing that other types of interesting objects or structures, beyond sparse signals and images, can be recovered from a limited set of measurements. Moreover, the techniques for proving our main results build upon ideas from the compressed sensing literature together with powerful probabilistic tools for bounding norms of operators between Banach spaces.

Also, our notion of coherence generalizes the concept of the same name in compressive sensing. Notably, the authors Candès and Romberg⁷ introduce the notion of the coherence of a unitary transformation $\mathbf{U}$; the coherence of $\mathbf{U}$ is simply proportional to $\max_{j,k} |U_{j,k}|^2$. This quantity plays a crucial role in determining the minimal sampling rate necessary to recover a k -sparse signal by ℓ_1 minimization.

In Recht et al.,³⁰ the authors studied the nuclear norm heuristic applied to a related problem where partial information about a matrix $\mathbf{M}$ is available from m equations of the form

$$\langle \mathbf{A}^{(k)}, \mathbf{M} \rangle = \sum_{ij} A_{ij}^{(k)} M_{ij} = b_k, \quad k = 1, \dots, m. \quad (2.8)$$

Here, for each k , $\{A_{ij}^{(k)}\}_{ij}$ is an i.i.d. sequence of Gaussian or Bernoulli random variables and the sequences $\{\mathbf{A}^{(k)}\}$ are also independent of each other (the sequences $\{\mathbf{A}^{(k)}\}$ and $\{b_k\}$ are available to the analyst). Building on the concept of *restricted isometry* in the context of sparse signal recovery, Recht et al.³⁰ establish the first sufficient conditions for which the nuclear norm heuristic returns the minimum rank element in the constraint set. The authors prove that the heuristic succeeds with large probability whenever the number m of available measurements is greater than a constant times $2nr \log n$ for $n \times n$ matrices of rank r . These results do not generalize to the matrix completion problem of interest to us in this paper. The measurements in (2.8) give some information about *all* the entries of $\mathbf{M}$, whereas in our problem information about most of the entries is simply not available. As a consequence, our methods are quite different and require more involved probabilistic analysis.

Our work also has close connections with the study of stochastic algorithms for low-rank matrix *approximation*. In this body of work, one is interested in sampling some entries of a matrix in order to construct an approximate factorization of this matrix. Typically, it is assumed that one may sample any subset of entries but would like to minimize the computational complexity involved in constructing an approximation. Pioneering work in this area appears in Frieze et al.¹⁵ and Liberty et al.,²⁵ and an extensive survey of these methods can be found in Halko et al.¹⁹ While this body of work also uses similar foundational theory of random matrices, our modeling assumptions are fundamentally different. Here, we are primarily concerned with the scenario where we have very limited control over which entries of the matrix we can observe. In the examples described in the introduction, one does not have access to all of the entries of the matrix due to systemic constraints. Surprisingly, our results demonstrate that low-rank matrices can be recovered *exactly* from almost all sufficiently large subsets of entries. However, when we have the ability to sample entries at will, the algorithmic recovery schemes become considerably more efficient. We see our results as complementary extremes of the sort of access one may have to the entries of a matrix.

Indeed, when the sampling can be chosen in specially designed patterns, the exact recovery problem becomes dramatically simpler. For example, suppose that $\mathbf{M}$ is generic and that we precisely observe every entry in the first r rows and columns of the matrix.

Write $\mathbf{M}$ in block form as

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix},$$

with $\mathbf{M}_{11}$ an $r \times r$ matrix. In the special case that $\mathbf{M}_{11}$ is invertible and $\mathbf{M}$ has rank r , it is easy to verify that $\mathbf{M}_{22} = \mathbf{M}_{21} \mathbf{M}_{11}^{-1} \mathbf{M}_{12}$. One can prove this identity by forming the SVD of $\mathbf{M}$. That is, if $\mathbf{M}$ is generic, the upper $r \times r$ block is invertible, and we observe *every* entry in the first r rows and columns, we can recover $\mathbf{M}$. This result immediately generalizes to the case where one observes r rows and r columns and the $r \times r$ matrix at the intersection of the observed rows and columns is invertible. Algorithms in the stochastic low-rank matrix approximation literature are essentially no more complicated than this simple algorithm. They use randomness to add numerical robustness and to guarantee that the sampled entries span the row/column space of the matrix to be acquired.

3. RECENT ADVANCES IN LOW-RANK MODELING

Our original article announced the possibility of various refinements and extensions, and invited researchers to develop the new field of matrix completion. We are pleased to see that the area of low-rank modeling and matrix completion has been quite active, and the field is growing at a very fast pace. In fact, there are so many new and exciting results recently developed that it is unfortunately impossible to review them all here. Below, we

survey selected progress that has occurred since our original submission.

3.1. Improvements and other approaches

The results discussed in Section 2.1 show that under suitable conditions, one can reconstruct an $n \times n$ matrix of rank r from a small number, m , of its sampled entries provided that m is on the order of $n^{1.2} r \log n$, at least for moderate values of the rank. One would like to know whether better results hold, in the sense that exact matrix recovery would be guaranteed with a reduced number of measurements. In particular, recall that an $n \times n$ matrix of rank r depends on $(2n - r)r$ degrees of freedom; is it possible to recover most low-rank matrices from on the order of nr randomly selected entries? Can the sample size be merely proportional to the true complexity of the low-rank object we wish to recover?

In this direction, we would like to emphasize that there is nothing in the approach of our original paper that stands in the way of stronger results. Our proof architecture requires bounding an infinite matrix series in the operator norm. We develop a bound on the spectral norm of each of the first four terms of this series and a general argument to bound the remainder of the series in the full version of this paper. Presumably, one could bound higher order terms by the same techniques. Getting an appropriate bound on the fifth term would lower the exponent of n from $6/5$ to $7/6$. The appropriate bound on the sixth term would further lower the exponent to $8/7$, and so on. To obtain an optimal result, one would need to bound $O(\log n)$ terms.

Following this main idea, the authors Candès and Tao⁹ reduced the upper bound on the number of required measurements to $O(nr \log^6(n))$ using a combinatorial argument to bound precisely this particular series. Their results depend on some additional assumptions, including a “strong incoherence condition” that is more restrictive than the one defined in Section 2.1. However, they also show that *no algorithm* could succeed with high probability if less than $\Theta(nr \log(n))$ entries were observed.

An unexpected and clever method for approximating this infinite matrix series was invented in Gross et al.¹⁸ This new approach used powerful large deviation bounds from quantum information theory combined with an iterative construction that circumvented much of the combinatorics necessary for the proof in Candès and Tao.⁹ This approach is dramatically simpler than previous approaches, and, using this technique, it was shown that $O(nr \log^2(n))$ entries were sufficient for exact matrix completion in Gross¹⁷ and Recht.²⁹ In Recht,²⁹ the leading constant was even upper bounded by 64.

From a very different perspective, the authors in Keshavan et al.²¹ provided a non-convex algorithm for low-rank matrix recovery. Here, the authors analyze a gradient descent scheme over the Grassmannian manifold of subspaces. Using some of the techniques developed in the full version of this paper, the authors show that this nonconvex problem is actually convex in a neighborhood of the true low-rank matrix provided the number of observed entries exceeds

$O(n \log(n))$ and the rank is less than $\log(n)$. This provides the asymptotically tightest bound on the number of entries required for recovery, but the authors need to assume that the singular values of the unknown low-rank matrix are all of order unity and that the rank is less than $\log(n)$ for their results to hold.

3.2. Toward a more general theory

More general measurement models. In our original work, we anticipated in Section 1.3 that our results would extend to the case where one observes a small number of arbitrary linear functionals of a hidden matrix $\mathbf{M}$. Set $N = n^2$ and let $\mathbf{A}_1, \dots, \mathbf{A}_N$ be an orthonormal basis for the linear space of $n \times n$ matrices with the usual inner product $\langle \mathbf{X}, \mathbf{Y} \rangle = \text{trace}(\mathbf{X}^T \mathbf{Y})$. Then, we predicted that our results should also apply to the rank minimization problem

$$\text{minimize rank}(\mathbf{X}) \quad \text{subject to } \langle \mathbf{A}_k, \mathbf{X} \rangle = \langle \mathbf{A}_k, \mathbf{M} \rangle \quad k \in \Omega, \quad (3.1)$$

where $\Omega \subset \{1, \dots, N\}$ is selected uniformly at random. In fact, (3.1) is (2.2) when the orthobasis is the canonical basis $(\mathbf{e}_i \mathbf{e}_j^T)_{1 \leq i, j \leq n}$. We conjectured that those low-rank matrices that have small inner product with all the basis elements $\mathbf{A}_k$ may be recoverable by nuclear norm minimization.

This conjecture was proven to be true by Gross,¹⁷ where a general definition of coherence was provided, and it was shown that the same number of measurements sufficed for reconstruction under this modified definition. Additionally, in Gross et al.,¹⁸ it was shown that the *Pauli basis*, studied in quantum information theory, was incoherent with *any* basis. This fact follows because all of the matrices in the Pauli basis are mutually orthogonal and unitary. Hence, the Pauli basis is a deterministic collection of matrices such that a random subset of these matrices can be used to reconstruct any low-rank matrix. This result has been applied to propose new methods in quantum-state tomography where one aims to determine the state of some quantum mechanical system with as few measurements or experiments as possible.

Matrix completion with noise. All of the results described above concern the problem of *exact* matrix completion, where we have perfect information about the entries of the matrix to be reconstructed. Of course, in almost all real-world problems, we can only gain access to noisy samples of the entries of the matrices we would like to recover. Fortunately, many authors have investigated the stability of matrix completion when the observations are noisy. The first such result⁶ uses a stability argument based on convexity to guarantee accurate recovery from noisy data. Several subsequent works studied this problem under different matrix models. The work in Keshavan et al.²² gives near-optimal bounds provided the unknown matrix obeys additional assumptions which say that the singular values are all about the same size. In Negahban and Wainwright,²⁸ error bounds are derived provided the

matrix is not spiky; that is to say, assuming that all the entries have about the same magnitude. We additionally invite interested readers to peruse Koltchinskii et al.,²³ which very recently introduced powerful results with yet a slightly different matrix model.

Algorithmic innovations. While it was known that the nuclear norm problem could be efficiently solved by semidefinite programming, the results of Recht et al.³⁰ and the full version of this paper have inspired the development of many special purpose algorithms to rapidly minimize the nuclear norm. For example, in Cai et al.⁴ and Ma et al.,²⁷ the authors show that many of the fast first order methods developed for compressed sensing problems can be adapted to solve large-scale matrix completion problems. These algorithms are projected gradient algorithms which operate by alternately correcting the predictions on the observed entries and soft-thresholding the singular values of the iterate. Using accelerated gradient schemes, the authors Ji and Ye²⁰ and Toh and Yun³⁴ have improved these algorithms to provide very fast implementations of nuclear norm minimization. These codes enable the solution of problems with hundreds of thousands of rows and columns in a few hours on a standard workstation. Moreover, the numerical experiments in these works confirm that nuclear minimization can successfully recover very large low-rank matrices with on the order of 3 to 5 times the number of latent degrees of freedom. Additional experiments demonstrate that low-rank matrices can be robustly recovered under significant additive noise using nuclear-norm minimization.

3.3. From sparsity to rank and beyond

In concert with Recht et al.,³⁰ our work on matrix completion crystallized some of the foundational ideas of compressed sensing. We were able to extend the notion of sparsity to the much more general concept of matrix rank, and situate the main ideas of compressed sensing in a dramatically broader context.

One exciting new development since the appearance of our original paper shows that the notions of sparsity and rank are in some sense orthogonal. If a matrix can be written as a sum of a low-rank matrix and a sparse matrix, then these two matrices can be identified and deconvolved from their sum. Deterministic conditions required for such an algorithm to work were provided in Chandrasekaran et al.¹² A randomized analysis in Candes et al.⁵ provided sharper recovery guarantees and furnished a new method for Robust Principal Components Analysis, demonstrating that principal components could be constructed even in the presence of a large number of outliers. Moreover, the results of Chandrasekaran et al.¹² were extended to provide convex algorithms for identifying Gaussian graphical models in Chandrasekaran et al.¹⁰ Prior art in this area had resorted to nonconvex heuristics based on Expectation–Maximization with no provable guarantees. It is quite surprising that, under very modest assumptions, a convex algorithm can solve a hidden variable estimation problem in multivariate statistics.

This subsequent research has shown that there is much more work to be done in this area. Work in compressed sensing, matrix completion, and their generalizations have shown that convex optimization can be used to solve a myriad of hard identification problems at nearly optimal rates. But the picture is likely much broader than what we currently understand. There are likely notions of simplicity beyond rank and sparsity that can also be leveraged in high-dimensional data analysis to open new frontiers in low-rate sampling. New work in Chandrasekaran et al.¹¹ develops a unified program for recovering simple signals and objects from incomplete information, illustrating a general approach for translating expert domain knowledge into convex optimization algorithms. This work not only generalizes prior art on compressed sensing and matrix completion but also provides several new models where low-rate sampling can recover specially structured models. Such new developments suggest that we have only begun to scratch the surface of the types of models and objects that may be recovered from highly incomplete information. 

References

1. Amit, Y., Fink, M., Srebro, N., Ullman, S. Uncovering shared structures in multiclass classification. In *Proceedings of the Twenty-fourth International Conference on Machine Learning* (2007).
2. Argyriou, A., Evgeniou, T., Pontil, M. Multi-task feature learning. In *Neural Information Processing Systems*, 2007.
3. Beck, C., D'Andrea, R. Computational study and comparisons of LFT reducibility methods. In *Proceedings of the American Control Conference* (1998).
4. Cai, J.F., Candès, E.J., Shen, Z. A singular value thresholding algorithm for matrix completion. *SIAM J. Optim.* 20, 4 (2008), 1956–1982.
5. Candès, E.J., Li, X., Ma, Y., Wright, J. Robust principal component analysis? Submitted for publication. Preprint available at <http://www-stat.stanford.edu/~candes/papers/RobustPCA.pdf>, 2009.
6. Candès, E.J., Plan, Y. Matrix completion with noise. *Proc. IEEE* 98, 6 (2009), 925–936.
7. Candès, E.J., Romberg, J. Sparsity and incoherence in compressive sampling. *Inverse Probl.* 23, 3 (2007), 969–985.
8. Candès, E.J., Romberg, J., Tao, T. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inform. Theor.* 52, 2 (2006), 489–509.
9. Candès, E.J., Tao, T. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theor.* 56, 5 (2009), 2053–2080.
10. Chandrasekaran, V., Parrilo, P.A., Willsky, A.S. Latent variable graphical model selection via convex optimization. Submitted for publication. Preprint available at http://ssg.mit.edu/~venkate/cpw_lgm_preprint10.pdf, 2010.
11. Chandrasekaran, V., Recht, B., Parrilo, P.A., Willsky, A. The convex geometry of linear inverse problems. Submitted for publication. Preprint available at arxiv.org/abs/1012.0621, 2010.
12. Chandrasekaran, V., Sanghavi, S., Parrilo, P.A., Willsky, A.S. Rank-sparsity incoherence for matrix decomposition. *SIAM J. Optim.* 21, 2 (2011), 572–596.
13. Donoho, D.L. Compressed sensing. *IEEE Trans. Inform. Theor.* 52, 4 (2006), 1289–1306.
14. Fazel, M. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, 2002.
15. Frieze, A., Kannan, R., Vempala, S. Fast Monte-Carlo algorithms for finding low-rank approximations. *J. ACM* 51, 6 (2004).
16. Funk, S. Netflix update: Try this at home, December 2006. <http://sifter.org/~simon/journal/20061211.html>.
17. Gross, D. Recovering low-rank matrices from few coefficients in any basis. *IEEE Trans. Inform. Theor.* 57 (2011), 1548–1566.
18. Gross, D., Liu, Y.K., Flammia, S.T., Becker, S., Eisert, J. Quantum state tomography via compressed sensing. *Phys. Rev. Lett.* 105, 15 (2010), 150401.
19. Halko, N., Martinsson, P.G., Tropp, J.A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* 53, 2 (2011), 217–288.
20. Ji, S., Ye, J. An accelerated gradient method for trace norm minimization. In *Proceedings of the ICML* (2009).
21. Keshavan, R.H., Montanari, A., Oh, S. Matrix completion from a few entries. *IEEE Trans. Inform. Theor.* 56, 6 (2009), 2980–2998.
22. Keshavan, R.H., Montanari, A., Oh, S. Matrix completion from noisy entries. *J. Mach. Learn. Res.* 11 (2009), 2057–2078.
23. Koltchinskii, V., Tsybakov, A.B., Lounici, K. Nuclear norm penalization and optimal rates for noisy low rank matrix completion. Submitted for publication. Preprint available at arxiv.org/abs/1011.6256, 2010.
24. Koren, Y., Bell, R., Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
25. Liberty, E., Woolfe, F., Martinsson, P.G., Rokhlin, V., Tygert, M. Randomized algorithms for the low-rank approximation of matrices. *Proc. Nat. Acad. Sci.* 104, 51 (2007), 20167–20172.
26. Linial, N., London, E., Rabinovich, Y. The geometry of graphs and some of its algorithmic applications. *Combinatorica* 15 (1995), 215–245.
27. Ma, S., Goldfarb, D., Chen, L. Fixed point and Bregman iterative methods for matrix rank minimization. *Math. Program.* 128, 1 (2011), 321–353.
28. Negahban, S., Wainwright, M.J. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. Submitted for publication. Preprint available at arxiv.org/abs/1009.2118, 2010.
29. Recht, B. A simpler approach to matrix completion. *J. Mach. Learn. Res.* (2009). To Appear 2012. Preprint available at <http://arxiv.org/abs/0910.0651>.
30. Recht, B., Fazel, M., Parrilo, P. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. *SIAM Rev.* 52, 3 (2010), 471–501.
31. Rennie, J.D.M., Srebro, N. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the International Conference of Machine Learning* (2005).
32. So, A.M.C., Ye, Y. Theory of semidefinite programming for sensor network localization. *Math. Program., Ser. B* 109 (2007), 367–384.
33. Sturm, J.F. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Meth. Software* 11–12 (1999), 625–653.
34. Toh, K.-C., Yun, S. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pac. J. Math.* 6 (2010), 615–640.

Emmanuel Candès (candes@stanford.edu), Departments of Mathematics and Statistics, Stanford University, Stanford, CA.

Candès was partially supported by a National Science Foundation grant CCF-515362, by the 2006 Waterman Award (NSF) and by an ONR grant.

Benjamin Recht (brecht@cs.wisc.edu) Department of Computer Sciences, University of Wisconsin-Madison, WI.

This work was performed while Recht was a post-doc at the Center for the Mathematics of Information at the California Institute of Technology. He was partially funded by ONR grant N00014-08-1-0749.

Technical Perspective

The Fox and the Hedgehog

By Peter Lee

“THE FOX KNOWS many things, but the hedgehog knows one big thing.” Philosophers have used this line, which is attributed to the ancient Greek poet Archilochus, to capture the notion that a person can be either a *generalist* or a *specialist*. While the former may be broad-minded and capable of coping reasonably with many different kinds of problems, the latter is someone who can apply deep expertise in an important, though perhaps narrow, domain.

This distinction exists in software, too. Programs can be written to be either general-purpose or special-purpose. General-purpose code has the advantage of being usable in a variety of situations, whereas special-purpose code might be written in a way that takes advantage of unique characteristics of the execution environment and thus gain efficiency at the cost of reusability. Perhaps the classic example is the problem of matrix multiplication. While this may be considered a toy problem, it turns out to be instructive nonetheless. The general-purpose algorithm for matrix multiply is easy to write and handles any two matrices as inputs; however, a significantly more efficient algorithm would be preferable if we just happen to know that one of the matrices is sparse. The problem, of course, is that we write the code and compile it during an early stage when we do not yet know what the inputs look like; it is only in the later execution stage when the inputs become available.

On the surface, this might appear to be just a matter of trading between programmer effort and computational efficiency. But a key part of what makes well-written software *well written* is that it makes as few assumptions about the external environment as possible. Unfortunately, software written with such generality can be too costly in terms of computational efficiency. Thus, programmers are often forced to write specialized code that takes advantage of the particulars of a specific set of assumptions.

We can ask the question: Is it possible to write code so that it is general-purpose but then have it automatically specialize itself to the situation at hand during execution?

This is a question that researchers have thought about for many years. While ideas related to “self-modifying code” have been around since the earliest days of computing, one of the first significant demonstrations that automatic code specialization might have important practical benefits on a practical scale was given in 1988 by Pu, Massalin, and Ioannidis.^a In this work, they presented the Synthesis operating system and showed it could take advantage of dynamic data values in order to generate, at runtime, specialized kernel routines. This enabled a system design with exceptionally clean, high-level interfaces and yet the high performance of a more complicated system. On the other hand, the implementation of Synthesis itself was difficult, and in general, the problem of writing programs that can generate specialized code at runtime has continued to prove challenging. Variations on the Lisp-like concepts of program quotation and macro expansion are both limiting and error-prone.

As a result, an important direction of research has involved the search for language and compiler technology that can allow programmers to write general-purpose code that is then correctly and efficiently turned into high-performance specialized code at runtime. This would allow, for example, a programmer to write a general matrix-multiply code but then still get much better performance if one of the inputs turns out to be sparse. In this line of research, one of the core ideas is the concept of *staging*. We imagine the execution of a program proceeding in a series of stages, each one calculating values that are used by later stages. What we seek, then, is to write the program code

so that somehow these stages are made apparent. If this is accomplished, then we can arrange for the later-stage code to be compiled into *code generators* that optimize against the results of earlier-stage computations.

This is easier said than done, of course. Among the many approaches that have been proposed in recent years, in my view the most promising are those that make use of sophisticated type systems to manage the identification of stages, as Tiark Rompf and Martin Odersky have proposed in the following paper on *Lightweight Modular Staging* (LMS). By using a type system, the programmer can indicate her intent with regard to the program staging, and have much of that intent typechecked for various kinds of consistency, for example, to make sure that later-stage computations really do come after early-stage computations. Furthermore, programs are not restricted to specialized code generation at just compile time or runtime. In fact, LMS programs can express arbitrarily many computation stages, obtaining the benefits of code generation at every stage. No matter how many stages, the type system provides the guarantee that, in every stage, all values that are needed for code generation will have been computed in time.

Besides providing guarantees on the consistency of staging, the language also supports good program structure and modularity. In particular, through the use of language features such as method inheritance and overriding, important aspects of the code specialization can be encapsulated neatly in program libraries.

With ideas such as LMS, we get closer to the day when all of our programs can be written with elegant generality and yet take advantage of runtime information for the best possible efficiency and user experience. 

a C. Pu, H. Massalin, and J. Ioannidis. The Synthesis kernel. *Computing Systems 1* (1988), 11–32.

Peter Lee (petelee@microsoft.com) is corporate vice president of Microsoft Research, Redmond, WA.

Lightweight Modular Staging: A Pragmatic Approach to Runtime Code Generation and Compiled DSLs

By Tiark Rompf and Martin Odersky

Abstract

Good software engineering practice demands generalization and abstraction, whereas high performance demands specialization and concretization. These goals are at odds, and compilers can only rarely translate expressive high-level programs to modern hardware platforms in a way that makes best use of the available resources.

Generative programming is a promising alternative to fully automatic translation. Instead of writing down the target program directly, developers write a program generator, which produces the target program as its output. The generator can be written in a high-level, generic style and can still produce efficient, specialized target programs. In practice, however, developing high-quality program generators requires a very large effort that is often hard to amortize.

We present *lightweight modular staging* (LMS), a generative programming approach that lowers this effort significantly. LMS seamlessly combines program generator logic with the generated code in a single program, using only types to distinguish the two stages of execution. Through extensive use of component technology, LMS makes a reusable and extensible compiler framework available at the library level, allowing programmers to tightly integrate domain-specific abstractions and optimizations into the generation process, with common generic optimizations provided by the framework.

LMS is well suited to develop embedded domain-specific languages (DSLs) and has been used to develop powerful performance-oriented DSLs for demanding domains such as machine learning, with code generation for heterogeneous platforms including GPUs. LMS has also been used to generate SQL for embedded database queries and JavaScript for web applications.

1. INTRODUCTION

Building and managing complex software systems is only possible by generalizing functionality and abstracting from particular use cases. Achieving performance, on the other hand, requires concretizing configurations and specializing code to its particular environment. Generative programming can bridge this gap by translating away abstraction overhead and effectively specializing generic programs.

Generative programming can be broadly classified as static or dynamic. Static code generation happens at compile time; a widely used example is the C++ template language or macro systems in other languages. Dynamic code generation that takes place at program runtime brings additional flexibility, because code can be specialized with respect to parameters not yet available at compile time.

Many computations can naturally be separated into stages distinguished by frequency of execution or availability of information. Staging transformations aim at executing certain pieces of code less often or at a time when performance is less critical. In the context of program generation, multistage programming (MSP, *staging* for short) as established by Taha and Sheard²² allows programmers to explicitly delay evaluation of a program expression to a later stage (thus, *staging* an expression). The present stage effectively acts as a code generator that composes (and possibly executes) the program of the next stage. A nice property of this approach is that generator and generated code are expressed in a single program, with the aim that programmers can construct a multistage program from a naive implementation of the same algorithm by adding staging annotations in a selective way.

Basic mechanisms for composing program fragments at runtime have existed for a long time in the form of Lisp's "code is data" model and its use of *quasi-quotation*: syntactic annotations to denote expressions that should remain unevaluated and to mark holes within them, to be filled in with expressions computed elsewhere. Dedicated MSP languages such as MetaML²² and MetaOCaml¹ add well-scoping and well-typing guarantees about the generated code. Despite these advances, incorporating dynamic code generation into larger systems in the form of self-optimizing "active" libraries or adding compilation to embedded domain-specific languages (DSLs) that are implemented as libraries remains a significant challenge.

1.1. Contributions

In this paper, we present *lightweight modular staging*

A previous version of this paper was published in the *Proceedings of the 9th International Conference on Generative Programming and Component Engineering* (Oct. 10–13, Eindhoven, The Netherlands).

(LMS), a dynamic code generation approach that further reduces the effort to develop sophisticated program generators. The presentation uses the Scala programming language, but the core concepts are not tied to any language in particular.

The classical introductory staging example is to specialize the power function for a given exponent, assuming that a program will take many different numbers to the same power. Considering the usual implementation,

```
def power(b: Double, n: Int): Double =
  if (n == 0) 1.0 else b * power(b, n - 1)
```

we want to turn the base *b* into a *staged* expression. Following Carette et al.² and Hofer et al.,⁸ and in contrast to the quasi-quotation approach of syntactic annotations, the central idea of LMS is to use types to distinguish the computational stages. In particular, we change *b*'s declared type from `Double` to `Rep[Double]`. The meaning of having type `Rep[Double]` is that *b* *represents* a computation that will yield a `Double` in the next stage. We also change the function's return type accordingly.

Now we need to regain the ability to do arithmetic on *b*, which is no longer a plain `Double`. The second idea of LMS is to package operations on staged types as components. To make its required functionality explicit, we wrap the power function itself up as a component (a trait):

```
trait Power { this: Arith =>
  def power(b: Rep[Double], n: Int): Rep[Double] =
    if (n == 0) 1.0 else b * power(b, n - 1)
}
```

In Scala, traits are similar to classes but they can be used in mix-in composition, a limited form of multiple inheritance.¹⁵ The self-type annotation `this: Arith` denotes a “requires” relationship: whenever an instance of `Power` is created, an instance of a concrete (but unspecified) subclass of `Arith` (see Figure 3) that provides staged double arithmetic must be mixed in, too.

LMS shares many benefits with earlier staging approaches:

- ▶ Code generators and generated code are expressed in the same program.
- ▶ Objects that are live within the generator's heap can be accessed from generated code if the code is invoked directly (cross-stage persistence).
- ▶ Staged expressions inherit the static scope of the generator, and if the generator is well-typed, so is the generated code.
- ▶ Data types representing staged expressions are inaccessible to the program itself (making optimizations that preserve only semantic but not structural equality safe).

At the same time, LMS differs from previous approaches in the following important ways:

- ▶ Staging is driven entirely by types, and no special syntax is required.

- ▶ Given a sufficiently expressive programming language, the whole framework can be implemented as a library (hence lightweight).
- ▶ Staged code fragments are composed through explicit operations, in particular, lifted variants of the usual operators and control flow statements extended with optimizing symbolic rewritings (semantic composition instead of syntactic expansion).
- ▶ Using component technology, operations on staged expressions, data types to represent them, and optimizations (both generic and domain specific) can be extended and composed in a flexible way (hence modular).
- ▶ Different code generation targets can easily be supported; their implementations can share common code.
- ▶ The relative evaluation order of expressions is preserved across stage boundaries. There is no danger of accidentally omitting, reordering, or duplicating computation.

The last item deserves more explanation, as it is arguably the most prevalent difficulty that programmers face with other staging approaches. Since freely composing code fragments voids any evaluation order guarantees (such as call-by-value semantics), adding quasi-quotation annotations can easily change the result of a program in unforeseen ways or slow down the program by duplicating computation. In practice, quasi-quotation is therefore often used as a low-level tool, like an “assembly language” for code generation, and combined with application-specific front-end layers that apply high-level code optimizations and ensure correct evaluation order. LMS provides a predictable execution order and makes the composition of staged expressions extensible, and thus removes the need for extra front ends.

LMS is a key constituent of Delite,^{13,18} an open-source framework for high-performance parallel DSLs that has been used to develop DSLs for demanding application areas such as machine learning, graph processing or mesh-based partial differential equation solvers. In this context, LMS enables “abstraction without regret.” DSL programmers can use arbitrary Scala features to structure their programs in the generator stage, with the comforting knowledge that LMS guarantees to remove these abstractions during staging and no runtime price will need to be paid. Coupled with advanced compiler optimizations such as data parallel loop fusion and architecture-specific data structure transformations, Delite generates efficient code for a variety of parallel platforms like multi-core CPUs and GPUs.

LMS has also been used to generate SQL statements for queries embedded in Scala programs and to generate JavaScript from staged Scala fragments, allowing Web applications to execute parts of their logic in the client's browser.

1.2. Organization

The rest of this paper is structured as follows. Section 2 presents an end-to-end example, turning a naive algorithm into an efficient code generator, while introducing the

major LMS components on the way. Section 2.1 discusses representations of staged code, Section 2.2 is concerned with optimizations, Section 2.3 is concerned with target code generation, and Section 2.4 concludes the example by showing how the generated code can be integrated in larger programs. Section 3 describes how additional features can be added, in particular functions and recursion. Section 4 discusses related work. An earlier version of this paper appeared at GPCE 2010. The original version contains a few additional examples and a more thorough discussion of how effectful statements are represented, while the present version has been updated with some new material in Section 3.

2. AN END-TO-END EXAMPLE

In the same way as the simple power function shown above, we can stage far more interesting and practically relevant programs, for example, the fast Fourier transform (FFT). A staged FFT, implemented in MetaOCaml, has been presented by Kiselyov et al.¹² Their work is a very good showcase for how staging allows one to transform a simple, unoptimized algorithm into an efficient program generator. Achieving this in the context of MetaOCaml, however, required restructuring the program into monadic style and adding a front-end layer for performing symbolic rewritings. Using our approach of just adding `Rep` types, we can go from the naive textbook-algorithm to the staged version (shown in Figure 1) by changing literally two lines of the code:

```
trait FFT { this: Arith with Trig =>
  case class Complex(re: Rep[Double], im:
    Rep[Double])
  ...
}
```

All that is needed is adding the self-type annotation to import arithmetic and trigonometric operations and changing the type of the real and imaginary components of complex numbers from `Double` to `Rep[Double]`.

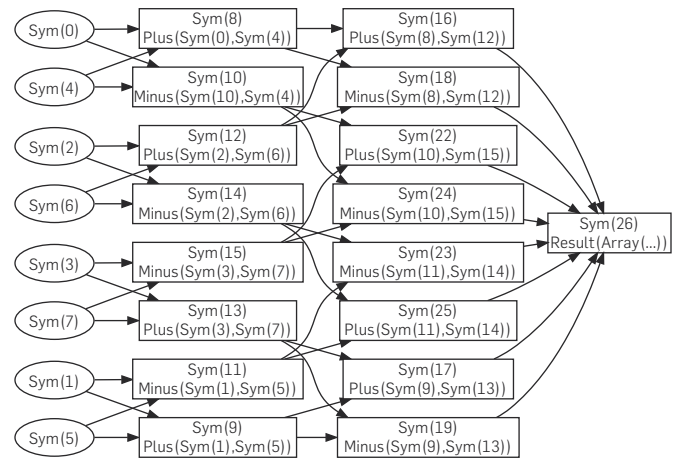
Merely changing the types will not provide us with the desired optimizations yet. We see below how we can add the transformations described by Kiselyov et al. to generate the same fixed-size FFT code, corresponding to the famous FFT butterfly networks (see Figure 2). Despite the seemingly naive algorithm, this staged code is free from branches, intermediate data structures, and redundant computations. The important point here is that we can add these transformations without any further changes to the code in Figure 1, just by mixing in the trait `FFT` with a few others.

In the remainder of this section, we present the LMS framework that is used to generate the code in Figure 2 from the algorithm in Figure 1. Before considering specific optimizations, let us have a closer look at the definition of `Rep` and the traits `Arith` and `Trig`. The definitions are given in Figure 3. In trait `Base`, the declaration `type Rep[+T]` defines an abstract type constructor

Figure 1. FFT code. Only the real and imaginary components of complex numbers need to be staged.

```
trait FFT { this: Arith with Trig =>
  case class Complex(re: Rep[Double], im: Rep[Double]) {
    def +(that: Complex) =
      Complex(this.re + that.re, this.im + that.im)
    def *(that: Complex) = ...
  }
  def omega(k: Int, N: Int): Complex = {
    val kth = -2.0 * k * Math.Pi/N
    Complex(cos(kth), sin(kth))
  }
  def fft(xs: Array[Complex]): Array[Complex] = xs match {
    case (x :: Nil) => xs
    case _ =>
      val N = xs.length // assume it's a power of two
      val (even0, odd0) = splitEvenOdd(xs)
      val (even1, odd1) = (fft(even0), fft(odd0))
      val (even2, odd2) = (even1 zip odd1 zipWithIndex)
        map {
          case ((x, y), k) =>
            val z = omega(k, N) * y
            (x + z, x - z)
        }.unzip;
      even2 ::: odd2
  }
}
```

Figure 2. Computation graph for size-4 FFT. Autogenerated from staged code in Figure 1.



(also called a higher kinded type) `Rep` which we take to range over possible representations of staged expressions. Since `Rep` is abstract, no concrete representation is defined yet; the declaration merely postulates the existence of *some* representation.

Trait `Arith` extends trait `Base` and contains only abstract members. These postulate the existence of an

Figure 3. Interface traits defining staged operations. For simplicity, operations are defined for `Double` only.

```
trait Base {
  type Rep[+T]
}

trait Arith extends Base {
  implicit def unit(x: Double): Rep[Double]
  def infix_+(x: Rep[Double], y: Rep[Double]):
    Rep[Double]
  def infix_*(x: Rep[Double], y: Rep[Double]):
    Rep[Double]
  ...
}

trait Trig extends Base {
  def cos(x: Rep[Double]): Rep[Double]
  def sin(x: Rep[Double]): Rep[Double]
}
```

implicit lifting of `Doubles` to staged values and the usual arithmetic operations on staged expressions of type `Rep[Double]`. The restriction to `Doubles` is just to keep the presentation concise. Any suitable means to abstract over numeric types, such as the “type class” `Numeric` from the Scala standard library, could be used to define `Arith` in a generic way for a range of numeric types. Analogous to `Double`, we could define arithmetic on matrices and vectors and implement optimizations on those operations in exactly the same way. Trait `Trig` is similar to `Arith` but defines trigonometric operations.

One way to look at `Base`, `Arith`, and `Trig` is as the definition of a typed embedded language (or its syntax). The embedding is *tagless* (i.e., method resolution happens at compile time without additional runtime dispatch overhead)² and *polymorphic*,⁸ in the sense that we are free to pick any suitable concrete implementation that fulfills the given interface.

From a safety point of view, keeping the actual representation inaccessible to the program generator is very important. Otherwise, the program generator could execute a different code depending on the exact structure of a staged expression. Optimizations that replace staged code with simpler but semantically equivalent expressions would risk changing the meaning of the generated program.

2.1. Representing staged code

With the aim of generating code, we could represent staged expressions directly as strings. But for optimization purposes we would rather have a structured intermediate representation that we can analyze in various ways.

We choose a representation on the basis of expression trees or, more exactly, a “sea of nodes” representation that is in fact a directed (and for the moment, acyclic) graph but can be accessed through a tree-like interface. The necessary infrastructure is defined in trait `Expressions`, shown in Figure 4.

Figure 4. Expression representation (method implementations have been omitted).

```
trait Expressions {
  // expressions (atomic)
  abstract class Exp[+T]
  case class Const[T](x: T) extends Exp[T]
  case class Sym[T](n: Int) extends Exp[T]

  def fresh[T]: Sym[T]

  // block scopes
  abstract class Block[T]

  def reifyBlock[T](x: => Exp[T]): Block[T]

  // definitions (composite, subclasses provided
  // by other traits)
  abstract class Def[T]

  def findDefinition[T](s: Sym[T]): Option[Def[T]]
  def findDefinition[T](d: Def[T]): Option[Sym[T]]
  def findOrCreateDefinition[T](d: Def[T]): Sym[T]

  // bind definitions to symbols automatically
  implicit def toAtom[T](d: Def[T]): Exp[T] =
    findOrCreateDefinition(d)

  // pattern match on definition of a given symbol
  object Def {
    def unapply[T](s: Sym[T]): Option[Def[T]] =
      findDefinition(s)
  }
}
```

There are three categories of objects involved: expressions, which are atomic (subclasses of `Exp`: constants and symbols; with a “gensym” operator `fresh` to create fresh symbols), definitions, which represent composite operations (subclasses of `Def`, to be provided by other components), and blocks, which model nested scopes.

The guiding principle is that each definition has an associated symbol and refers to other definitions only via their symbols. In effect, this means that every composite value will be named, similar to administrative normal form (ANF). The trait `Expressions` provides methods to find a definition given a symbol, or vice versa. The extractor object `Def` allows one to pattern-match on the definition of a given symbol, a facility that is used for implementing rewritings (see below).

The framework ensures that the code that contains staging operations will always be executed within the dynamic scope of at least one invocation of `reifyBlock`, which returns a block object and takes as call-by-name argument the present-stage expression that will compute the staged block result. Block objects can be part of definitions, for example, for loops or conditionals.

The implicit conversion method `toAtom` converts a definition to an atomic expression and links it to the scope being built up by the innermost enclosing `reifyBlock`

call. When the definition is known to be side-effect free, `toAtom` will search the already encountered definitions for a structurally equivalent one. If a matching previous definition is found, its symbol will be returned, possibly moving the definition to a parent scope to make it accessible. If the definition has side effects or it is seen for the first time, it will be associated with a fresh symbol and saved for future reference. This simple scheme provides a powerful global value numbering (common subexpression elimination) optimization that effectively prevents generating duplicate code. More complicated optimization schemes can be added, too.

Since all operations in interface traits such as `Arith` return `Rep` types, defining `Rep[T] = Exp[T]` in trait `BaseExp` (see Figure 5) means that conversion to symbols will take place within the constructor methods (e.g., `cos` or `infix_*`). This fact is important because it establishes our correspondence between the evaluation order of the program generator and the evaluation order of the generated program: at the point where the generator calls `toAtom`, the composite definition is turned into an atomic value, that is, its evaluation is recorded now and played back later in the same relative order with respect to others within the closest `reifyBlock` invocation.

We observe that there are no concrete definition classes provided by the trait `Expressions`. Providing meaningful data types is the responsibility of other traits that implement the interfaces defined previously (`Base` and its descendants). For each interface trait, there is one corresponding core implementation trait. Shown in Figure 5, we have traits `BaseExp`, `ArithExp`, and `TrigExp` for the functionality required by the FFT example. The trait `BaseExp` installs atomic

expressions as the representation of staged values by defining `Rep[T] = Exp[T]`. Traits `ArithExp` and `TrigExp` define one definition class for each operation defined by `Arith` and `Trig`, respectively, and implement the corresponding interface methods to create instances of those classes.

2.2. Implementing optimizations

Some profitable optimizations, such as the global value numbering described above, are very generic. Other optimizations apply only to specific aspects of functionality, for example, particular implementations of constant folding (or more generally, symbolic rewritings) such as replacing computations like `x * 1.0` with `x`. Yet, other optimizations are specific to the actual program being staged. In the FFT case, Kiselyov et al.¹² describe a number of rewritings that are particularly effective for the patterns of code generated by the FFT algorithm but not as much as for other programs.

What we want to achieve again is modularity, so that optimizations can be combined in a way that is most useful for a given task. To implement a particular rewriting rule (whether specific or generic), say, `x * 1.0` $\rightarrow$ `x`, we can provide a specialized implementation of `infix_*` (overriding the one in trait `ArithExp`) that will test its arguments for a particular pattern. How this can be done in a modular way is shown by the traits `ArithExpOpt` and `ArithExpOptFFT`, which implement some generic and program-specific optimizations (see Figure 6). Note that the use of `x*y` within the body of `infix_*` will apply the optimization recursively.

Figure 6. Extending the implementation from Figure 5 with generic (top) and specific (bottom) optimizations (analog of `TrigExp` has been omitted).

Figure 5. Implementing the interface traits from Figure 3 using the expression types from Figure 4.

```
trait BaseExp extends Base with Expressions {
  type Rep[+T] = Exp[T]
}

trait ArithExp extends Arith with BaseExp {
  implicit def unit(x: Double) = Const(x)
  case class Plus(x: Exp[Double], y: Exp[Double])
    extends Def[Double]
  case class Times(x: Exp[Double], y: Exp[Double])
    extends Def[Double]

  def infix_+(x: Exp[Double], y: Exp[Double]) =
    Plus(x, y)
  def infix_*(x: Exp[Double], y: Exp[Double]) =
    Times(x, y)
}

trait TrigExp extends Trig with BaseExp {
  case class Sin(x: Exp[Double]) extends Def[Double]
  case class Cos(x: Exp[Double]) extends Def[Double]
  def sin(x: Exp[Double]) = Sin(x)
  def cos(x: Exp[Double]) = Cos(x)
}
```

```
trait ArithExpOpt extends ArithExp {
  override def infix_*(x: Exp[Double], y: Exp[Double]) =
    (x, y) match {
      case (Const(x), Const(y)) => Const(x * y)
      case (x, Const(1)) => x
      case (Const(1), y) => y
      case _ => super.infix_*(x, y)
    }
}

trait ArithExpOptFFT extends ArithExp {
  override def infix_*(x: Exp[Double], y: Exp[Double]) =
    (x, y) match {
      case (Const(k), Def(Times(Const(1), y))) =>
        Const(k * 1) * y
      case (x, Def(Times(Const(k), y))) =>
        Const(k) * (x * y)
      case (Def(Times(Const(k), x)), y) =>
        Const(k) * (x * y)
      ...
      case (x, Const(y)) => Times(Const(y), x)
      case _ => super.infix_*(x, y)
    }
}
```

In essence, we are confronted with the classic “expression problem” of independently extending a data model to new data variants and new operations. There are many solutions to this problem but most of them are rather heavyweight. More lightweight implementations are possible in languages that support multi-methods, that is, dispatch method calls dynamically based on the actual types of all the arguments. Figure 6 shows how we can achieve essentially the same (plus *deep* inspection of the arguments) using pattern matching and mix-in composition, making use of the fact that composing traits is subject to linearization.¹⁵ We package each set of arithmetic optimizations into its own trait that inherits from `ArithExp` and overrides the desired methods (e.g., `infix_*`). When the arguments do not match the rewriting pattern, the overridden method will invoke the “parent” implementation using `super`. When several such traits are combined, the `super` calls will traverse the overridden method implementations according to the linearization order of their containing traits.

Implementing multi-methods in a statically typed setting usually poses three problems: separate type checking/compilation, ensuring nonambiguity, and ensuring exhaustiveness. The described encoding supports separate type-checking and compilation as far as traits do. Ambiguity is ruled out by always following the linearization order and the first-match semantics of pattern matching. Exhaustiveness is ensured at the type level by requiring a default implementation, although no guarantees can be made that the default will not choose to throw an exception at runtime. In the particular case of applying optimizations, the default is always safe as it will just create an expression object.

2.3. Generating code

Code generation is an explicit operation. For the common case where generated code is to be loaded immediately into the running program, the trait `Compile` provides a suitable interface in form of the abstract method `compile` (see Figure 8). The contract of `compile` is to “unstage” a function from staged to staged

values into a function operating on present-stage values that can be used just like any other function object in the running program.

For generating Scala code, an implementation of the compilation interface is provided by trait `CompileScala`. This trait extends another trait, `ScalaGenBase`, whose subclasses are responsible for traversing nested blocks and generating Scala code for individual definition nodes. Of course, the traversal code can also be factored out and shared by multiple code generation targets. Subclasses of `ScalaGenBase` are structured in a similar way as those of `Base`, that is, one for each unit of functionality (see Figure 9). Code generation can omit side-effect free graph nodes that are unreachable from the final result, effectively performing a dead code elimination optimization.

The overall compilation logic of `CompileScala` is relatively simple: emit a class and apply-method declaration header, emit instructions for each definition node according to the schedule, close the source file, invoke the Scala compiler, load the generated class file, and return a newly instantiated object of that class.

2.4. Putting it all together

In the previous sections, we have discussed the major ingredients of lightweight modular staging, focusing mostly on individual components. Figure 7 shows an overview of the traits encountered so far and their relationships.

Using the staged FFT implementation as part of some larger Scala program is straightforward but requires us to interface the generic algorithm with a concrete data representation. The algorithm in Figure 1 expects an array of `Complex` objects as input, each of which contains fields of type `Rep[Double]`. The algorithm itself has no notion of staged arrays but uses arrays only in the generator stage, which means that it is agnostic to how data is stored. The enclosing program, however, will store arrays of complex numbers in some native format which we will need to feed into the algorithm. A simple choice of representation is to use `Array[Double]` with the complex numbers flattened into adjacent slots. When applying `compile`, we will thus receive input of

Figure 7. Component architecture. Arrows denote extends relationships and dashed boxes represent units of functionality.

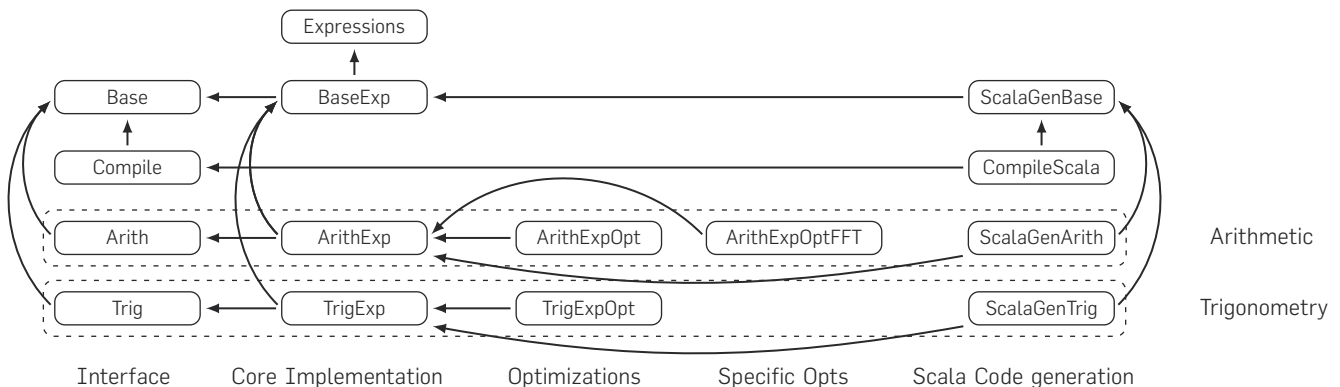


Figure 8. Code generation interface and skeleton of Scala compilation component.

```
trait Compile extends Base {
  def compile[A,B] (f: Rep[A] => Rep[B]): A=>B
}

trait CompileScala extends Compile with ScalaGenBase =>
  def compile[A,B] (f: Exp[A] => Exp[B]) = {
    val x = fresh[A]
    val y = reifyBlock { f(x) }
    // emit header
    emitBlock(y)
    //
    //
    //
    //
  }
}
```

Figure 9. Scala code generation for selected expressions.

```
trait ScalaGenBase extends BaseExp {
  def emitBlock[T] (Block[T]) = ...
  def emitNode[T] (sym: Sym[T], node: Def[T]) =
    throw new Exception("node " + node + " not supported")
}

trait ScalaGenArith extends ScalaGenBase with ArithExp {
  override def emitNode(sym: Sym[T], node: Def[T]) =
    node match {
      case Plus(a,b) => println("val %s = %a + %b".format(sym,a,b))
      case Times(a,b) => println("val %s = %a * %b".format(sym,a,b))
      case _ => super.emitNode(sym, rhs)
    }
}
```

Figure 10. Extending the FFT component from Figure 1 with explicit compilation.

```
trait FFTC extends FFT { this: Arrays with Compile =>
  def fftc(size: Int) = compile {
    input: Rep[Array[Double]] =>
    assert(<size is power of 2>) // staging time
    val arg = Array.tabulate(size) {i =>
      Complex(input(2*i), input(2*i+1))
    }
    val res = fft(arg)
    updateArray(input, res.flatMap {
      case Complex(re,im) => Array(re,im)
    })
  }
}
```

type `Rep[Array[Double]]`. Figure 10 shows how we can extend the trait `FFT` to `FFTC` to obtain compiled FFT implementations that realize the necessary data interface for a fixed input size.

We can then define a code that creates and uses compiled FFT “codelets” by extending `FFTC`:

```
trait TestFFTC extends FFTC {
  val fft4: Array[Double] => Array[Double] =
    fftc(4)
  val fft8: Array[Double] => Array[Double] =
    fftc(8)
  // embedded code using fft4, fft8, ...
}
```

Constructing an instance of this subtrait (mixed in with the appropriate LMS traits) will execute the embedded code:

```
val OP: TestFFC = new TestFFTC with CompileScala
  with ArithExpOpt with ArithExpOptFFT with
    ScalaGenArith
  with TrigExpOpt with ScalaGenTrig
  with ArraysExpOpt with ScalaGenArrays
```

We can also use the compiled methods from outside the object:

```
OP.fft4(Array(1.0,0.0, 1.0,0.0, 2.0,0.0, 2.0,0.0))
↪ Array(6.0,0.0, -1.0,1.0, 0.0,0.0, -1.0,-1.0)
```

Providing an explicit type in the definition `val OP:TestFFC= ...` ensures that the internal representation is not accessible to the outside, but only the members defined by `TestFFC` are accessible.

3. ADDING MORE FEATURES

Many features can be added in a way that is analogous to what we have seen above, but some require a bit more thought. In this section, we take a closer look at staged functions. Basic support for staged function definitions and function applications can be defined in terms of a simple higher order abstract syntax (HOAS)¹⁶ representation, similar to those of Carette et al.² and Hofer et al.⁸ (see Figure 11). The idea is to provide a lambda operation that transforms present-stage functions over staged values (type `Rep[A] => Rep[B]`) to staged function values (type `Rep[A=>B]`). Note how this is similar to the signature of `compile`. To give an example, the staged recursive factorial function will look like this:

```
def fac: Rep[Int => Int] = lambda {n =>
  if (n == 0) 1
  else n * fac(n - 1)
}
```

Figure 11. Representation of λ -abstractions as Scala function values (higher order abstract syntax).

```

trait Functions extends Base {
  def lambda[A,B] (f: Rep[A] => Rep[B]): Rep[A=>B]
  def infix_apply[A,B] (f: Rep[A=>B], x:
Rep[A]):
    Rep[B]
}

```

As opposed to the earlier power example, an invocation `fac(m)` will not inline the definition of `fac` but will result in an actual function call in the generated code.

However, the HOAS representation has the disadvantage of being opaque: there is no immediate way to “look into” a Scala function object. If we want to treat functions in the same way as other program constructs, we need a way to transform the HOAS encoding into our graph representation. We can implement `lambda(f)` to call

```

reifyBlock { f(fresh[A]) }

```

which will unfold the function definition into a `Block` that represents the entire computation defined by the function. But eagerly expanding function definitions is problematic. For recursive functions, the result would be infinite, that is, the computation will not terminate. What we would like to do instead is to detect recursion and generate a finite representation that makes the recursive call explicit. However, this is difficult because recursion might be very indirect:

```

def foo(x: Rep[Int]) = {
  val f = (x: Rep[Int]) => foo(x + 1)
  val g = lambda(f)
  g(x)
}

```

Each incarnation of `foo` creates a new function `f`; unfolding will thus create unboundedly many different function objects.

To detect cycles, we have to *compare* those functions. This, of course, is undecidable in the general case of taking equality to be defined extensionally, that is, saying that two functions are equal if they map equal inputs to equal outputs. By contrast, the standard reference equality, which compares memory addresses of function objects, is too weak for our purpose:

```

def adder(x: Int) = (y: Int) => x + y
adder(3) == adder(3)
↪ false

```

However, we can approximate extensional equality by intensional (i.e., structural) equality, which is sufficient in most cases because recursion will cycle through a well-defined code path in the program text. Testing intensional equality amounts to checking whether two functions are defined at the same syntactic location in the source program and whether all data referenced by their free variables is equal. Fortunately, the implementation of first-class functions as closure objects offers (at least in principle) access to a “defunctionalized” data type representation on which equality can easily be checked. A bit of care must be taken though, because the structure can be cyclic. On the Java virtual machine (JVM), there is a particularly neat trick. We can serialize the function objects into a byte array and compare the serialized representations:

```

serialize(adder(3)) == serialize(adder(3))
↪ true

```

With this method of testing equality, we can implement *controlled* unfolding. Unfolding functions only once at the definition site and associating a fresh symbol with the function being unfolded allows us to construct a block that contains a recursive call to the symbol we created. Thus, we can create the expected representation for the factorial function above.

3.1. Guarantees by construction

Making staged functions explicit through the use of `lambda` enables tight control over how functions are structured and composed. For example, functions with multiple parameters can be specialized for a subset of the parameters. Consider the following implementation of Ackermann’s function:

```

def ack(m: Int): Rep[Int=>Int] = lambda { n =>
  if (m == 0) n+1 else
  if (n == 0) ack(m-1)(1) else
  ack(m-1)(ack(m)(n-1))
}

```

Calling `ack(m)(n)` will produce a set of mutually recursive functions, each specialized to a particular value of `m`. For `ack(2)(n)`, for example, we get

```

def ack_2(n: Int) = if (n == 0) ack_1(1) else
  ack_1(ack_2(n-1))
def ack_1(n: Int) = if (n == 0) ack_0(1) else
  ack_0(ack_1(n-1))
def ack_0(n: Int) = n+1
ack_2(n)

```

In essence, this pattern implements what is known as “polyvariant specialization” in the partial evaluation community. But unlike automatic partial evaluation, which might or might not be able to discover the *right*

specialization, the use of staging provides strong guarantees about the structure of the generated code. In this case, we are guaranteed that specialization will happen for each value of m (but never for n), statically evaluating tests on values of m and inserting constants for all occurrences of m in the generated code.

Other strong guarantees can be achieved by restricting the interface of function definitions. Being of type $\text{Rep } [A \Rightarrow B]$, the result of `lambda` is a first-class value in the generated code that can be stored or passed around in arbitrary ways. However, we might want to avoid higher order control flow in generated code for efficiency reasons, or to simplify subsequent analysis passes. In this case, we can define a new function constructor `fundef` as follows:

```
def fundef [A,B] (f: Rep [A] => Rep [B]): Rep [A] =>
  Rep [B] =
  (x: Rep [A]) => lambda (f).apply (x)
```

The use of `fundef` instead of `lambda` produces a restricted function that can only be applied but not passed around in the generated code (`type Rep [A] => Rep [B]`). At the same time, a result of `fundef` is still a first class value in the code *generator*. If we do not expose `lambda` and `apply` at all to client code, we obtain a guarantee that each function call site unambiguously identifies the function definition being called and no closure objects will need to be created at runtime.

4. RELATED WORK

Static program generation approaches include C++ templates, and Template Haskell.²⁰ Building on C++ templates, customizable generation approaches are possible through Expression Templates.²³ An example of runtime code generation in C++ is the TaskGraph framework. Active libraries were introduced by Veldhuizen²⁴ and telescoping languages by Kennedy et al.¹¹ Specific toolkits using domain-specific code generation and optimization include FFTW,⁶ SPIRAL,¹⁷ and ATLAS.²⁵

This paper draws a lot of inspiration from the work of Kiselyov et al.¹² on a staged FFT implementation. Performing symbolic rewritings by defining operators on lifted expressions and performing common subexpression elimination on the fly is also central to their approach. LMS takes these ideas one step further by making them a central part of the staging framework itself.

Immediately related works on embedding typed languages include those of Carette et al.² and Hofer et al.⁸ Lee et al.^{13,18} describe how LMS is used in the development of DSLs for high-performance parallel computing on heterogeneous platforms.

Multistage Programming Languages such as MetaML²² and MetaOCaml¹ have been proposed as a disciplined approach to building code generators. These languages provide three syntactic annotations: *brackets*, *escape*, and *run*, which together provide a syntactic quasi-quotation facility that is similar to that found in Lisp but statically scoped and

statically typed.

MSP languages make writing program generators easier and safer, but they inherit the essentially syntactic notion of combining program fragments, which incurs the risk of duplicating or reordering computation.^{3, 21} Code duplication can be avoided systematically by writing the generator in continuation-passing or monadic style, using appropriate combinators to insert `let`-bindings in strategic places. However, this is often impractical since this style significantly complicates the generator code. Another possibility is to make extensive use of side effects in the generator part, but side effects invalidate some of the static guarantees of MSP languages. This dilemma has been described as an “agonizing trade-off,” because of which one “cannot achieve clarity, safety, and efficiency at the same time.”¹⁰

By contrast, lightweight modular staging prevents code duplication by handling the necessary side effects inside the staging primitives, which are semantic combinators instead of syntactic expanders. Therefore, code generators can usually be written in purely functional direct style and are much less likely to invalidate safety assurances.

Another characteristic of some MSP languages is that staged code cannot be inspected because of safety considerations. Thus, domain-specific optimizations must happen on an external intermediate representation, before using the MSP primitives to generate code.¹² The burden of choosing and implementing a suitable representation is on the programmer, and it is not clear how different representations can be combined or reused.

Lightweight modular staging provides a systematic interface for adding symbolic rewritings. Safety is maintained by exposing the internal code structure only to rewriting modules but keeping it hidden from the client generator code.

Compiled embedded DSLs, as studied by Leijen and Meijer¹⁴ and Elliott et al.,⁵ can also be implemented using MSP languages by writing an explicit interpreter and adding staging annotations in a second step.^{4, 7, 19} This is simpler than writing a full compiler, but compared to constructing explicit interpreters, “purely” embedded languages that are implemented as plain libraries have many advantages.⁹ LMS allows a simpler approach, by starting with a pure embedding instead of an explicit interpreter. In many cases, adding some type annotations in strategic places is all that is needed to get to a staged embedding. If domain-specific optimizations are needed, new AST classes and rewriting rules are easily added.

Acknowledgments

The development of lightweight modular staging has benefited greatly from the work on Delite in collaboration with the Stanford Pervasive Parallelism Lab, in particular Arvind Sajeeth, Hassan Chafi, Kevin Brown, HyoukJoong Lee and Kunle Olukotun. A number of members of the authors’ group at EPFL have applied LMS to interesting use cases and contributed new insights or valuable extensions to the framework. 

References

1. Calcagno, C., Taha, W., Huang, L., Leroy, X. Implementing multi-stage languages using asts, gensym, and reflection. *GPCE*, F. Pfenning and Y. Smaragdakis, eds. Volume 2830 of *Lecture Notes in Computer Science* (2003). Springer, 57–76.
2. Carette, J. Kiselyov, O., chieh Shan, C., Finally tagless, partially evaluated: Tagless staged interpreters for simpler typed languages. *J. Funct. Program.* 19, 5 (2009), 509–543.
3. Cohen, A., Donadio, S., Garzarán, M.J., Herrmann, C.A., Kiselyov, O., Padua, D.A. In search of a program generator to implement generic transformations for high-performance computing. *Sci. Comput. Program.* 62, 1 (2006), 25–46.
4. Czarnecki, K., O'Donnell, J.T., Striegnitz, J., Taha, W. Dsl implementation in metaocaml, template haskell, and c++. In *Domain-Specific Program Generation* (2003), 51–72.
5. Elliott, C., Finne, S., de Moor, O. Compiling embedded languages. *J. Funct. Program.* 13, 3 (2003), 455–481.
6. Frigo, M. A fast Fourier transform compiler. In *PLDI* (1999), 169–180.
7. Guerrero, M., Pizzi, E., Rosenbaum, R., Swadi, K.N., Taha, W. Implementing dsls in metaocaml. *OOPSLA Companion*, J. M. Vlissides and D. C. Schmidt, eds. (2004). ACM, 41–42.
8. Hofer, C., Ostermann, K., Rendel, T., Moors, A. Polymorphic embedding of dsls. *GPCE*, Y. Smaragdakis and J.G. Siek, eds. (2008). ACM, 137–148.
9. Hudak, P. Modular domain specific languages and tools. In *Proceedings of Fifth International Conference on Software Reuse* (June 1998), 134–142.
10. Kameyama, Y., Kiselyov, O., chieh Shan, C. Shifting the stage: staging with delimited control. *PEPM*, G. Puebla and G. Vidal, eds. (2009). ACM, 111–120.
11. Kennedy, K., Broom, B., Cooper, K.D., Dongarra, J., Fowler, R.J., Gannon, D., Johnsson, S.L., Mellor-Crummey, J.M., Torczon, L. Telescoping languages: A strategy for automatic generation of scientific problem-solving systems from annotated libraries. *J. Parallel Distrib. Comput.* 61, 12 (2001), 1803–1826.
12. Kiselyov, O., Swadi, K.N., Taha, W. A methodology for generating verified combinatorial circuits. *EMSOFT*, G.C. Buttazzo, ed. (2004). ACM, 249–258.
13. Lee, H., Brown, K.J., Sujeeth, A.K., Chafi, H., Rompf, T., Odersky, M., Olukotun, K. Implementing domain-specific languages for heterogeneous parallel computing. *IEEE Micro* 31, 5 (2011), 42–53.
14. Leijen, D., Meijer, E. Domain specific embedded compilers. In *DSL* (1999), 109–122.
15. Odersky, M., Zenger, M. Scalable component abstractions. *OOPSLA*, R. E. Johnson and R.P. Gabriel, eds. (2005). ACM, 41–57.
16. Pfenning, F., Elliott, C. Higher-order abstract syntax. In *PLDI* (1988), 199–208.
17. Püschel, M., Moura, J.M.F., Singer, B., Xiong, J., Johnson, J., Padua, D.A., Veloso, M.M., Johnson, R.W. Spiral: A generator for platform-adapted libraries of signal processing algorithms. *IJHPCA*, 18, 1 (2004), 21–45.
18. Rompf, T., Sujeeth, A.K., Lee, H., Brown, K.J., Chafi, H., Odersky, M., Olukotun, K. Building-blocks for performance oriented dsls. In *DSL* (2011), 93–117.
19. Sheard, T., Benaissa, Z.E.A., Pasalic, E. Dsl implementation using staging and monads. In *DSL* (1999), 81–94.
20. Sheard, T., Jones, S.L.P. Template meta-programming for haskell. *SIGPLAN Not.* 37, 12 (2002), 60–75.
21. Swadi, K.N., Taha, W., Kiselyov, O., Pasalic, E. A monadic approach for avoiding code duplication when staging memoized functions. *PEPM*, J. Hatcliff and F. Tip, eds. (2006). ACM, 160–169.
22. Taha, W., Sheard, T. Metaml and multi-stage programming with explicit annotations. *Theor. Comput. Sci.* 248(1–2) (2000), 211–242.
23. Veldhuizen, T. "Expression Templates," in Stanley B. Lippman (Editor), *C++ Gems* (SIGS Books and Multimedia, 1996), pp. 475–488.
24. Veldhuizen, T.L. Active libraries and universal languages. PhD thesis, Indiana University Computer Science (May 2004).
25. Whaley, R.C., Petitet, A., Dongarra, J. Automated empirical optimizations of software and the atlas project. *Parallel Comput.* 27(1–2) (2001), 3–35.

Tiark Rompf (tiark.rompf@epfl.ch), EPFL, Lausanne, Switzerland.

Martin Odersky (martin.odersky@epfl.ch), EPFL, Lausanne, Switzerland.

© 2012 ACM 0001-0782/12/06 \$10.00



Association for
Computing Machinery

Advancing Computing as a Science & Profession



MentorNet

You've come a long way.
Share what you've learned.



ACM has partnered with MentorNet, the award-winning nonprofit e-mentoring network in engineering, science and mathematics. MentorNet's award-winning **One-on-One Mentoring Programs** pair ACM student members with mentors from industry, government, higher education, and other sectors.

- Communicate by email about career goals, course work, and many other topics.
- Spend just **20 minutes a week** - and make a huge difference in a student's life.
- Take part in a lively online community of professionals and students all over the world.



Make a difference to a student in your field.
Sign up today at: www.mentornet.net
Find out more at: www.acm.org/mentornet

MentorNet's sponsors include 3M Foundation, ACM, Alcoa Foundation, Agilent Technologies, Amylin Pharmaceuticals, Bechtel Group Foundation, Cisco Systems, Hewlett-Packard Company, IBM Corporation, Intel Foundation, Lockheed Martin Space Systems, National Science Foundation, Naval Research Laboratory, NVIDIA, Sandia National Laboratories, Schlumberger, S.D. Bechtel, Jr. Foundation, Texas Instruments, and The Henry Luce Foundation.

Grove City College Professor of Computer Science

Grove City College announces a faculty opening in the Department of Computer Science beginning in August 2012. A Ph.D. in Computer Science or a closely related field is required. Responsibilities include teaching both lower-level and upper-level Computer Science courses, guiding student undergraduate-research projects, and developing innovative pedagogy.

Candidates must evidence a strong record of classroom teaching, scholarship within their field, and a commitment to instructing highly motivated students (1244/1600 average SAT for entering freshmen). Rank and salary are commensurate with qualifications. Grove City College is a nationally recognized, truly independent, highly selective college of liberal arts, sciences, and engineering where intellectual inquiry remains open to the questions religion raises and affirms the answers Christianity offers.

Send letter of application, vita, transcripts, names of four references (three professional and one pastoral), a brief statement of how you would engage undergraduates in your research plans, and a four to five paragraph essay relating your philosophy of a liberal arts education and teaching Computer Science to the College's unique mission as a Christian college (see www.gcc.edu) to: **William P. Anderson, Jr., Ph.D., Provost and Vice President for Academic Affairs, Grove City College, 100 Campus Drive, Grove City, PA 16127-2104** or electronically to laklaiber@gcc.edu. Review of applications commences immediately. In a continuing effort to enrich its academic and social environment, the college actively encourages applications from members of all ethnic groups.

Grove City College is a private educational institution noted for its academic excellence where scholarship is informed by Christian principles. It does not discriminate on the basis of age, race, color, creed, sex, marital status, disability, or national or ethnic origin in the administration of its educational policies, admission policies, scholarship and loan programs, and athletic and other college-administrative programs.

The Max Planck Institute for Informatics

Junior Research Groups Leaders in the Max Planck Center for Visual Computing and Communication

The Max Planck Institute for Informatics, as the coordinator of the Max Planck Center for Visual Computing and Communication (MPC-VCC), invites applications for

Junior Research Groups Leaders in the Max Planck Center for Visual Computing and Communication

The Max Planck Center for Visual Computing and

Communications offers young scientists in information technology the opportunity to develop their own research program addressing important problems in areas such as

- ▶ image communication
- ▶ computer graphics
- ▶ geometric computing
- ▶ imaging systems
- ▶ computer vision
- ▶ human machine interface
- ▶ distributed multimedia architectures
- ▶ multimedia networking
- ▶ visual media security

The center includes an outstanding group of faculty members at Stanford's Computer Science and Electrical Engineering Departments, the Max Planck Institute for Informatics, and Saarland University.

The program begins with a preparatory 1-2 year postdoc phase (**Phase P**) at the Max Planck Institute for Informatics, followed by a two-year appointment at Stanford University (**Phase I**) as a visiting assistant professor, and then a position at the Max Planck Institute for Informatics as a junior research group leader (**Phase II**). However, the program can be entered flexibly at each phase, commensurate with the experience of the applicant.

Applicants to the program must have completed an outstanding PhD. Exact duration of the preparatory postdoc phase is flexible, but we typically expect this to be about 1-2 years. Applicants who completed their PhD in Germany may enter Phase I of the program directly. Applicants for Phase II are expected to have completed a postdoc stay abroad and must have demonstrated their outstanding research potential and ability to successfully lead a research group.

Reviewing of applications will commence on **May 1, 2012**. The final deadline is **June 30, 2012**. Applicants should submit their CV, copies of their school and university reports, list of publications, reprints of five selected publications, names of references, a brief description of their previous research and a detailed description of the proposed research project (including possible opportunities for collaboration with existing research groups at Saarbrücken and Stanford) to:

Prof. Dr. Hans-Peter Seidel
Max Planck Institute for Informatics,
Campus E 1 4, 66123 Saarbrücken, Germany;
Email: mpc-vec@mpi-inf.mpg.de

The Max Planck Center is an equal opportunity employer and women are encouraged to apply. Additional information is available on the website <http://www.mpc-vec.org>



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@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 be 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:

acmm mediasales@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
acmm mediasales@acm.org**

Computing Reviews is on the move



Our new URL is

ComputingReviews.com



COMPUTING REVIEWS

A daily snapshot of what is new and hot in computing



DOI:10.1145/2184319.2184346

Peter Winkler

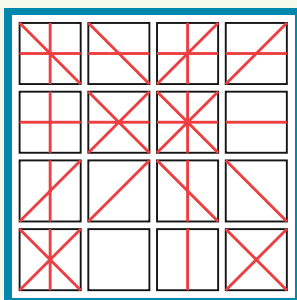
Puzzled Solutions and Sources

Last month (May 2012) we posted a trio of brainteasers concerning designs on square grids. Here, we offer solutions to all three. How did you do?

1. Tiling a 4×4 Grid.

Solution. You were given 16 unit squares, each containing a different combination of vertical crossing line, horizontal crossing line, SW-NE diagonal, and SE-NW diagonal. Your object was to tile a 4×4 grid with these squares in such a way that no line ends before it hits the edge of the grid—or prove it cannot be done. Figure 1 shows a solution, though there are several; it is indeed possible to arrive at one without much trial and effort, by, say, reasoning about how the long diagonal lines should be arranged. Note that if you identify the left and right boundaries of the square grid to form a cylinder, the lines continue to match up. Moreover, if you identify the top and bottom boundaries as well, so you now have a doughnut, or torus, all the lines will be endless loops. The puzzle's composer, Barry Cipra, noted the curious fact that all solutions work on the torus; see <http://www.funmathclub.com/students/sollewitt.html>. It is not surprising that the horizontal and vertical lines match up on the torus. They have to. But why do the diagonals happen to match up as well?

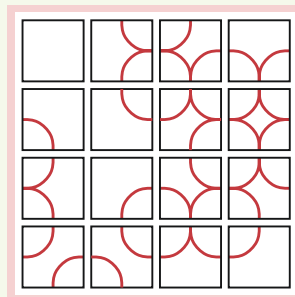
Figure 1.



2. Circles on a Torus.

Solution. In a second version of Puzzle 1, each unit square contained one of the 16 possible combinations of four quarter-circles, each of radius 1/2 and centered at a corner. You were asked to tile a 4×4 grid with these squares so no path ends before it hits the edge of the grid or, better, so an even number of quarter-circles meet at each edge shared by two squares. Figure 2 shows a solution satisfying both conditions, with the additional property that the curves form a collection of circles and semicircles. But you can do better. See if you can find a solution that matches left-right and top-bottom edges so you get nothing but circles on a torus.

Figure 2.

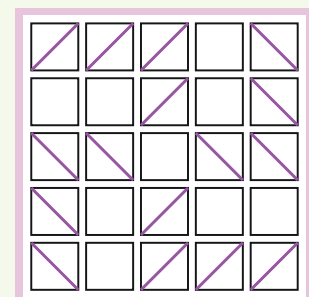


3. Diagonals in a 5×5 Grid.

Solution. You were asked to place as many diagonals as you can into a 5×5 grid with no two meeting at a corner (or crossing inside a square); Figure 3 shows you can achieve 16 diagonals. To see that more are impossible, note first that there were only $6 \times 6 = 36$ vertices in

the grid, with each diagonal using two of them, so you certainly cannot get more than 18 diagonals in the grid. Moreover, along each side of the grid are six vertices and only five squares, so, on each side, some vertex will go without a diagonal. If three or more vertices are left without diagonals, the maximum number of diagonals is 16. The only way the four sides can be handled with fewer than three empty vertices is if two are at opposite corners of the grid (at, say, the SW and NE corners) with no other empty vertices along the grid. But then every square touching a side of the grid would have to contain a SE-NW diagonal—an impossibility, because diagonals would then meet at the vertices diagonally opposite the empty grid corners.

Figure 3.



All readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is William Morrill Professor of Mathematics and Computer Science, at Dartmouth College, Hanover, NH.

© 2012 ACM 0001-0782/12/06 \$10.00

2012 - Big Data: End of the World or End of BI?



Thursday, June 28, 12 PM EST, 4 PM UTC

Register at learning.acm.org/webinar/bi

According to some, the world will end at the Winter Solstice this year. If not, it's certainly time to take a closer look at Business Intelligence (BI) and ask: "After 20 years, is it still fit for purpose? Can it deliver the type of support needed for decision-making in the next decade?"

We are witnessing the birth of a new "biz-tech ecosystem," where business and technology have become symbiotic and collaborative behavior is the norm. In this environment, decision-making is very different from what is supported by today's BI. This webinar explores the emerging ecosystem and

poses some interesting challenges for BI vendors and implementers in 2013... Assuming we're all still here... You will learn about:

- The meaning and emergence of the biz-tech ecosystem
- The re-convergence of operational and informational systems
- Data, information and knowledge – reopening Pandora's Box
- Millennials and team decision making – the whys and wherefores
- Architectural models for 2013 and beyond



➤ **Barry Devlin, 9sight Consulting** Barry Devlin is among the foremost authorities on business insight and one of the founders of data warehousing, having published the first architectural paper on the topic in 1988. With over 30 years of IT experience, including 20 years with IBM as a Distinguished Engineer, he is a widely respected analyst, consultant, lecturer, author of the seminal book *Data Warehouse—from Architecture to Implementation* and numerous White Papers, and an editor of the ACM Business Intelligence & Data Management Tech Pack.

Barry is founder and principal of 9sight Consulting. He specializes in the human, organizational, and IT implications of deep business insight solutions that combine operational, informational, and collaborative environments. A regular contributor to BeyeNETWORK, Focus, SmartDataCollective, and TDWI, Barry is based in Cape Town, South Africa and operates worldwide.



Association for
Computing Machinery

Advancing Computing as a Science & Profession

Space for this **free** webinar is limited. Register Today! ♦ Contact timanovsky@hq.acm.org for more information.

[CONTINUED FROM P. 136] quite powerfully, everything we associate with thought processes.

So you turned your attention to artificial intelligence, and to using probability for the representation and acquisition of knowledge.

On the one hand, I felt very strongly that we think probabilistically. On the other hand, we couldn't put probability into the computer without exceeding memory capacity. And you ask yourself, If we can't put it into a computer—if it takes exponential storage—how do we do it in our minds? The answer is that we utilize knowledge about irrelevance. We decompose problems into chunks that are only loosely connected.

This is what gets represented, in the model of belief propagation you then developed, through Bayesian networks.

If you have a network of loosely connected components, you can reason probabilistically without encountering exponential complexity. You can represent it parsimoniously and you can update it swiftly, and, moreover, you can update it in a distributed fashion. And that's very important, because we don't have a supervisor in our brain telling each neuron when to fire.

Your work on the topic was influenced by the writings of psychologist David Rumelhart, who proposed that, while we read, our brain's neural modules each perform simple, repetitive tasks, and that these modules use both top-down and bottom-up modes of inference to collaborate with one another.

If you pose these features as an architecture for doing things probabilistically, you ask yourself, When can we do it distributedly and still get the probabilistically correct answer to every question? And that led to a tree architecture and a proof that it converges eventually to the answers that orthodox probability theory dictates. And then came polytrees and the ultimate question of how we can do it when we have a general loopy network. Here I conjectured that the mind simply ignores the loops and allows every processor to act as if it was embedded in a polytree—and this worked miraculously well.

At this point, the practitioners took over, and they were able to do it much

"I called it do-calculus because it allows you to reduce questions about the effect of interventions to symbolic manipulations."

better than I. Even the theoreticians did better than I—they proved convergence under various conditions, and essentially I left this area to more talented and motivated researchers.

You left probability to work on causation.

Yes, primarily because it became clear that people encode world knowledge through causal, not probabilistic, relationships, and all those fancy notions of relevance and irrelevance come from causal, not probabilistic, considerations.

Among your best-known accomplishments is the creation of a calculus of intervention that enables us to compute the consequences of various actions.

The idea was to treat actions and observations as distinct symbols situated within the same formal sentence. This allows you to infer the consequences of actions from a combination of data and qualitative knowledge encoded in the form of a causal diagram.

In other words, it is where correlation and causation meet.

Yes, they meet in the calculus. I called it do-calculus because it allows you to reduce questions about the effect of interventions to symbolic manipulations. You want to predict what will happen if you do something based on what you observe. So you express this question in symbolic algebra and you can ask the question "What if I do x ?" or "What if I see y ?" as well as any other combination of doing and seeing. Then you submit the query to the inference engine and let it grind through until it gets you the right results.

Simulating intervention, by the way, was an idea that was thought of by economists in 1943. Trygve Haavelmo had this idea that economics models are a guide to policy-making, and that you can predict what will happen when the government intervenes and raises taxes or imposes duties by modifying the equations in the model. And that was taken on by other economists, but it didn't catch, because they had very lousy models of the economy, so they couldn't demonstrate success. And because they couldn't demonstrate success, the whole field of economics regressed and became a hotbed for statistical predictions. Economists have betrayed causality. I never expressed it this way before, but in all honesty this is what it boils down to. In computer science, we remain faithful to logic and try to improve our models, while economists compromised on logic to cover up for bad models.

Your work on causality culminated in counterfactuals.

There are three levels of causal relationships. The zero level, which is the level of associations, not causation, deals with the question "What is?" The second level is "What if?" And the third level is "Why?" That's the counterfactual level. Initially, I thought of counterfactuals as something for philosophers to deal with. Now I see them as just the opposite. They are the building blocks of scientific understanding.

Does your research inform your work at the Daniel Pearl Foundation, especially in conducting interfaith dialogues?

I have an advantage over my dialogue partners in that I'm an atheist, and I understand religious myths are just metaphors, or poetry, for genuine ideas we find difficult to express otherwise. So, yes, you could say I use computer science in my religious dialogues, because I view religion as a communication language. True, it seems futile for people to argue if a person goes to heaven from the East Gate or the West Gate. But, as a computer scientist, you forgive the futility of such debates, because you appreciate the computational role of the gate metaphor. 

Leah Hoffmann is a technology writer based in Brooklyn, NY.

© 2012 ACM 0001-0782/12/06 \$10.00

DOI:10.1145/2184319.2184347

Leah Hoffmann

Q&A

A Sure Thing

Artificial intelligence pioneer Judea Pearl discusses probability, causation, the calculus of intervention, and counterfactuals.

ACM A.M. TURING Award winner Judea Pearl took a somewhat unusual path to the field that now celebrates his pioneering achievements in artificial intelligence. Born in Israel in 1936, he grew up in Bnei Brak, a biblical town his grandfather refounded in 1924. After serving in the Israeli army and joining a kibbutz, he decided to study electrical engineering at the Technion-Israel Institute of Technology. (A profile of Pearl, “Game Changer,” appears on p. 22.)

He came to the U.S. in 1960, where he did graduate studies in physics and electrical engineering at Rutgers University and the Polytechnic Institute of Brooklyn, and worked at RCA Research Laboratories. In 1970, Pearl joined the University of California, Los Angeles’s (UCLA’s) computer science department and began the research for which he has become fa-

mous, creating a representational and computational foundation for reasoning with uncertain information.

In 2002, he founded the Daniel Pearl Foundation in honor of his son, a *Wall Street Journal* reporter who was kidnapped and murdered by anti-American militants in Pakistan.

You began your career in the 1960s at RCA Research Laboratories. What were you doing there?

I was doing something totally different than what I’m doing now: memory systems. At that time, computers used magnetic-core memories, which were really clumsy. You had to string magnetic donuts with wires, and hundreds of girls strung them one by one. People realized that this was a bottleneck of computers, so everyone was searching for new mechanisms to store information. Some worked on new magnetic

configurations, others worked on photochromic memory, and I worked on superconductors, where one could store information in the form of localized circulating vortices.

How did you move to academia?

Eventually, everything was wiped out by semiconductors, and people in this arena had to find new jobs. Nothing came out of my Ph.D. dissertation except one piece of analysis that physicists later named “Pearl vortex,” surpassing my wildest dream for immortality. So, looking for another endeavor, I came to the UCLA computer science department and offered my expertise with memory systems. But nothing was done in academia to utilize my experience on the hardware side, so I looked for another challenging arena. And who doesn’t want to emulate him- or herself?

You have said that computer scientists are driven to the field out of a desire to emulate and understand themselves.

It’s also why psychologists go into psychology. But we are luckier because we have a mechanism that can really do it. We have a machine that can emulate, [CONTINUED ON P. 135]



PHOTOGRAPHS BY RICHARD MORGENSTEIN

IFIP World Computer Congress

Towards an innovative, secure and sustainable information society

WCC 2012 • 24 - 26 September 2012

Amsterdam • the Netherlands • RAI Congress Centre

We invite you to an extremely exciting
event in Amsterdam



Register now!

on www.wcc-2012.org and benefit from the

Early Bird Fee!

Don't miss
this event!



This congress is supported by

City of Amsterdam, European Patent Office, International Telecommunication Union & UNESCO

twitter.com/wcc_2012



With the support of
Communication and
Information Sector

<http://www.facebook.com/wcc2012>





Join the Conversation

GAMES ART/DESIGN FILM/TV PRODUCTION RESEARCH
PRODUCT DEVELOPMENT EDUCATION STUDENT OTHER

Bring your artistic ability, scientific innovation, and everything in between to inspire and be inspired by the most diverse gathering in computer graphics and interactive techniques.

You Are
SIGGRAPH2012



The **39th** International
Conference and **Exhibition**
on **Computer Graphics** and
Interactive Techniques

Conference 5–9 August 2012
Exhibition 7–9 August 2012
Los Angeles Convention Center



Sponsored by ACM SIGGRAPH

www.siggraph.org/s2012

