

COMMUNICATIONS

CACM.ACM.ORG OF THE **ACM** 04/2017 VOL.60 NO.04

Attack of the Killer Microseconds

A Service Computing Manifesto

Uninitialized Reads

Sensors on the Brain

Wanted: Toolsmiths

Systor2017

May 22 – 24 Haifa, Israel

10 The 10th ACM International Systems and Storage Conference

We invite you to attend SYSTOR 2017, the ACM SIGOPS Systems and Storage Conference covering all aspects of systems research. Join us in Haifa from May 22 to May 24 for an exciting and enriching technical program, social events, and informal get-together with leading researchers from academia and industry.

This year we celebrate our 10th anniversary.

Registration is free of charge.

Program chairs

Peter Desnoyers, Northeastern University

Eyal de Lara, University of Toronto

Keynote speakers

Orran Krieger, Boston University

Emery Berger, University of Massachusetts Amherst

Timothy Roscoe, ETH Zürich

General chair

Doron Chen, IBM Research

Posters chair

Adam Morrison, Tel Aviv University

Steering committee head

Michael Factor, IBM Research

Steering committee

Ethan Miller, University of California Santa Cruz

Liuba Shrira, Brandeis University

Dan Tsafir, Technion

Dalit Naor, IBM Research

Erez Zadok, Stony Brook University

www.systor.org/2017/

Sponsored by



In cooperation with



Platinum sponsor



Gold sponsors



NOKIA Bell Labs

vmware

NUTANIX

Sponsors



ORACLE



The Blavatnik School of Computer Science
The Raymond and Beverly Sackler Faculty of Exact Sciences
Tel Aviv University



ICMI 2017

19th INTERNATIONAL CONFERENCE
ON MULTIMODAL INTERACTION

GLASGOW, Scotland
November 13-17, 2017

Long and Short
Paper Submission

MAY, 12th, 2017

Reviews Available
for Author Rebuttal

JUL, 21st, 2017

Paper Notification
Sent to Authors

AUG, 25th, 2017

Camera - Ready
Paper Deadline

SEP, 22nd, 2017

icmi.acm.org/2017



 SIGCHI



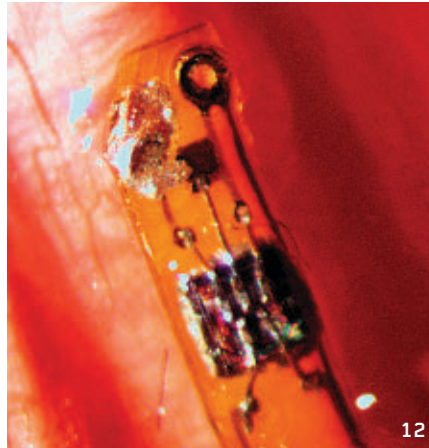
Departments

- 5 **From the ACM-W Chair**
**Gender Diversity in Computing:
Are We Making Any Progress?**
By Valerie Barr
-
- 7 **Cerf's Up**
**A Genetic Theory of
the Silicon Valley Phenomenon**
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
Consider Indirect Threats of AI, Too
-
- 10 **BLOG@CACM**
**Crafting a National Cyberdefense,
and Preparing to Support
Computational Literacy**
John Arquilla considers how
we should interpret the alleged
Russian cyberattack on the U.S.
Presidential election; Mark Guzdial
describes the potential benefits
of a 'computing lab.'
-
- 25 **Calendar**
-
- 95 **Careers**

Last Byte

- 96 **Upstart Puzzles**
Stacking the Deck
By Dennis Shasha

News



- 12 **Sensors on the Brain**
Implantable wireless monitors
give researchers a new look inside
the human body.
By Gregory Mone
-
- 15 **Digitizing the World**
Digital maps trawl for
real-time updates.
By Chris Edwards
-
- 17 **Computing the Arts**
Artists can use software to create art,
and some software creates art all
on its own.
By Esther Shein
-
- 20 **ACM Panels in Print**
Cybersecurity

Viewpoints



- 22 **Global Computing**
**Online Social Networks and Global
Women's Empowerment**
Mediating social change or
reinforcing male hegemony?
By Ineke Buskens
-
- 24 **Kode Vicious**
**The Chess Player Who Couldn't
Pass the Salt**
AI: Soft and hard, weak and strong,
narrow and general.
By George V. Neville-Neil
-
- 26 **Viewpoint**
Wanted: Toolsmiths
Seeking to use software,
hardware, and algorithmic
ingenuity to create unique
domain-independent instruments.
By William Regli
-
- 29 **Viewpoint**
**What It Means to Be
an Entrepreneur Today**
In his keynote address before the
fifth edition of the Tech Open Air
conference in Berlin in 2016,
Kickstarter's cofounder and
CEO Yancey Strickler suggests
the city's tech community faces
"a very rare opportunity."
By Yancey Strickler

Practice



32

32 **Pervasive, Dynamic Authentication of Physical Items**

The use of silicon PUF circuits.
By Meng-Day (Mandel) Yu and Srinivas Devadas

40 **Uninitialized Reads**

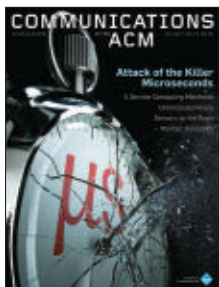
Understanding the proposed revisions to the C language.
By Robert C. Seacord

45 **Does Anybody Listen to You?**
How do you step up from mere contributor to real change-maker?
By Kate Matsudaira

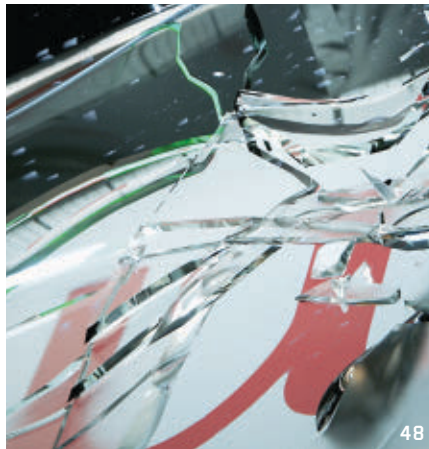
Articles' development led by **acmqueue**
queue.acm.org

About the Cover:

While techniques for addressing latencies at the millisecond and nanosecond level have proven effective, the challenges imposed by microsecond-scale latencies are growing in scope and intensity. This month's cover story (p. 48) raises awareness of microsecond issue and offers advice for research directions that should be explored. Cover illustration by Peter Crowther Associates.



Contributed Articles



48

48 **Attack of the Killer Microseconds**

Microsecond-scale I/O means tension between performance and productivity that will need new latency-mitigating ideas, including in hardware.
By Luiz Barroso, Mike Marty, David Patterson, and Parthasarathy Ranganathan



Watch the authors discuss their work in this exclusive *Communications* video.
<http://cacm.acm.org/videos/the-attack-of-the-killer-microseconds>

55 **Computational Thinking for Teacher Education**

This framework for developing pre-service teachers' knowledge does not necessarily depend on computers or other educational technology.
By Aman Yadav, Chris Stephenson, and Hai Hong



Watch the authors discuss their work in this exclusive *Communications* video.
<http://cacm.acm.org/videos/computational-thinking-for-teacher-education>

Review Articles

64 **A Service Computing Manifesto: The Next 10 Years**

Mapping out the challenges and strategies for the widespread adoption of service computing.

By Athman Bouguettaya, Munindar Singh, Michael Huhns, Quan Z. Sheng, Hai Dong, Qi Yu, Azadeh Ghari Neiat, Sajib Mistry, Boualem Benatallah, Brahim Medjahed, Mourad Ouzzani, Fabio Casati, Xumin Liu, Hongbing Wang, Dimitrios Georgakopoulos, Liang Chen, Surya Nepal, Zaki Malik, Abdelkarim Erradi, Yan Wang, Brian Blake, Schahram Dustdar, Frank Leymann, and Michael Papazoglou

Research Highlights

74 **Technical Perspective**
Proving File Systems Meet Expectations
*By Gernot Heiser*75 **Certifying a File System Using Crash Hoare Logic: Correctness in the Presence of Crashes**
*By Tej Chajed, Haogang Chen, Adam Chipala, M. Frans Kaashoek, Nikolai Zeldovich, and Daniel Ziegler*85 **Technical Perspective**
Building a Safety Net for Data Reuse
*By Jonathan Ullman*86 **Guilt-Free Data Reuse**
By Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth

Association for Computing Machinery
Advancing Computing as a Science & Profession



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
Bobby Schnabel
Deputy Executive Director and COO
Patricia Ryan
Director, Office of Information Systems
Wayne Graves
Director, Office of Financial Services
Darren Ramdin
Director, Office of SIG Services
Donna Cappel
Director, Office of Publications
Scott E. Delman

ACM COUNCIL

President
Vicki L. Hanson
Vice-President
Cherri M. Pancake
Secretary/Treasurer
Elizabeth Churchill
Past President
Alexander L. Wolf
Chair, SGB Board
Jeanna Matthews
Co-Chairs, Publications Board
Jack Davidson and Joseph Konstan
Members-at-Large
Gabriele Anderst-Kotis; Susan Dumais; Elizabeth D. Mynatt; Pamela Samuelson; Eugene H. Spafford
SGB Council Representatives
Paul Beame; Jenna Neefe Matthews; Barbara Boucher Owens

BOARD CHAIRS

Education Board
Mehran Sahami and Jane Chu Prey
Practitioners Board
Terry Coatta and Stephen Ibaraki

REGIONAL COUNCIL CHAIRS

ACM Europe Council
Dame Professor Wendy Hall
ACM India Council
Srinivas Padmanabhuni
ACM China Council
Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs
Jack Davidson; Joseph Konstan
Board Members
Ronald F. Boisvert; Karin K. Breitman; Terry J. Coatta; Anne Condon; Nikil Dutt; Roch Guerin; Carol Hutchins; Yannis Ioannidis; Catherine McGeoch; M. Tamer Ozsu; Mary Lou Soffa; Alex Wade; Keith Webster

ACM U.S. Public Policy Office
Renee Dopplick, Director
1701 Pennsylvania Ave NW, Suite 300,
Washington, DC 20006 USA
T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association
Mark R. Nelson, 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 PUBLICATIONS
Scott E. Delman
cacm-publisher@cacm.acm.org

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

Art Director
Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Director
Mia Angelica Balaquiot
Designer
Iwona Usakiewicz
Advertising Sales Account Manager
Ilia Rodriguez

Columnists
David Anderson; Phillip G. Armour;
Michael Cusumano; Peter J. Denning;
Mark Guzdial; Thomas Haigh;
Leah Hoffmann; Mari Sako;
Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

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

WEBSITE
http://cacm.acm.org

AUTHOR GUIDELINES
http://cacm.acm.org/

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

Advertising Sales Account Manager
Ilia Rodriguez
ilia.rodriguez@hq.acm.org

For display, corporate/brand advertising:
Craig Pitcher
pitcherc@acm.org T (408) 778-0300

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
eic@cacm.acm.org

NEWS

Co-Chairs
William Pullleyblank and Marc Snir
Board Members
Mei Kobayashi; Michael Mitzenmacher;
Rajeev Rastogi; François Sillion

VIEWPOINTS

Co-Chairs
Tim Finin; Susanne E. Hambrusch;
John Leslie King; Paul Rosenbloom
Board Members
William Aspray; Stefan Bechtold;
Michael L. Best; Judith Bishop;
Stuart I. Feldman; Peter Freeman;
Mark Guzdial; Rachelle Hollander;
Richard Ladner; Carl Landwehr;
Carlos Jose Pereira de Lucena;
Beng Chin Ooi; Loren Terveen;
Marshall Van Alstyne; Jeannette Wing

PRACTICE

Co-Chair
Stephen Bourne
Board Members
Eric Allman; Peter Bailis; Terry Coatta;
Stuart Feldman; Benjamin Fried;
Pat Hanrahan; Tom Killalea; Tom Limoncelli;
Kate Matsudaira; Marshall Kirk McKusick;
George Neville-Neil; Theo Schlossnagle;
Jim Waldo

The Practice section of the CACM Editorial Board also serves as the Editorial Board of [computer.org](http://www.computer.org).

CONTRIBUTED ARTICLES

Co-Chairs
Andrew Chien and James Larus
Board Members
William Aiello; Robert Austin; Elisa Bertino;
Gilles Brassard; Kim Bruce; Alan Bundy;
Peter Buneman; Peter Druschel; Carlo Ghezzi;
Carl Gutwin; Yannis Ioannidis;
Gal A. Kaminka; James Larus; Igor Markov;
Gail K. Murphy; Bernhard Nebel;
Lionel M. Ni; Kenton O'Hara; Sriram Rajamani;
Marie-Christine Rousset; Avi Rubin;
Krishan Sabnani; Ron Shamir; Yoav Shoham; Larry Snyder; Michael Vitale;
Wolfgang Wahlster; Hannes Werthner;
Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs
Azer Bestovros and Gregory Morrisett
Board Members
Martin Abadi; Amr El Abbadi; Sanjeev Arora;
Michael Backes; Nina Balcan; Andrei Broder;
Doug Burger; Stuart K. Card; Jeff Chase;
Jon Crowcroft; Alexei Efros; Alon Halevy;
Norm Jouppi; Andrew B. Kahng; Sven Koenig;
Xavier Leroy; Steve Marschner; Kobbi Nissim;
Guy Steele, Jr.; Margaret H. Wright;
Nicolai Zeldovich; Andreas Zeller

WEB Chair

James Landay
Board Members
Marti Hearst; Jason I. Hong;
Jeff Johnson; Wendy E. MacKay

ACM Copyright Notice

Copyright © 2017 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@hq.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 \$269.

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

Printed in the U.S.A.



Association for Computing Machinery



DOI:10.1145/3056417

Valerie Barr

Gender Diversity in Computing: Are We Making Any Progress?

RECENTLY I CAME across notes for a talk I gave in 1991 about women and computer science. It was depressing to read through it. Change the date and I could give the same talk today. How can that be? Hasn't the situation for women in computer science improved in the last 26 years? The answer to that question depends on what measure(s) you choose.

Yes, there are more women enrolling in and staying in computer science degree programs, but we should not overstate the improvement. All U.S. academic degree data must be analyzed against a significant shift in the overall undergraduate pool. From 1966 to 2015, the U.S. undergraduate population shifted from 42% to 57% women, requiring that we be very careful with how we evaluate disciplinary data. Published analyses of degree production in CS always state that 37% of CS degrees went to women in 1984, and only 18% in 2015. Consider a different perspective. In 1984, 2.42% of all women's degrees were earned in CS (the high point was actually 2.97% of women's degrees in 1986).

During 1989–2006, between 1% and 2% of women's degrees were earned in CS, and since then it has been less than 1% each year. For comparison, in 2014, 6% of men's degrees were earned in CS in 2015.

What about jobs? Yes, lots of women are being hired into tech. The exhibition hall at the Grace Hopper Conference and the career fairs at the ACM Celebrations of Women in Computing are full of companies eager to talk with student attendees and line up new hires. But, again, we have to temper excitement at these developments. While women are heading in the front doors of companies, they are hemorrhaging out


the side and back doors. Approximately 45% of women entering tech leave within five years while only 17% of men leave. Bringing more women into tech has not succeeded in changing the climate there, nor apparently led to significant changes in the attitudes and behavior of many people who work in tech. There are other efforts being made, such as providing employees with training on implicit bias, and carrying out large-scale salary review to ensure that wages are equitable across job titles.

But attitudinal change is slow.

It is time to ask how ACM, a membership organization that reaches millions of people every year, can contribute to efforts to make people aware of bias and to encourage them to change their attitudes and behavior. For years ACM has supported ACM-W, the Council on Women in Computing. Thanks to ACM's support, and additional funding from Google, Microsoft, and Oracle, today we have 30 Celebrations of Women in Computing worldwide, over 160 ACM-W Chapters, and we annually award over \$35,000 in scholarships for women CS students to attend research conferences. These efforts have great impact on the women involved, but little broader impact on the ACM

membership. What can ACM do to address issues of diversity in ways that will reach broadly across the membership?

In an exciting development, the ACM Executive Committee last June authorized the establishment of a working group that will formulate the charter for a new ACM Council on Diversity and Inclusion (CDI). The working group met in January 2017, and the new Council should launch at the start of the next fiscal year, July 1, 2017. No single group can possibly address all aspects of diversity and inclusion, particularly when we consider that ACM is a global organization. The CDI will foster the development of new committees, created by groups of ACM members who are passionate about particular diversity areas. This will enable ACM to, for example, improve our understanding of what diversity and inclusion issues are around the world, address LGBTQ issues, and address access as an issue for members and for those wishing to participate in ACM-sponsored events, not just as a research topic.

The working group has started to address the overarching question of what the role is of a membership organization in addressing issues of diversity and inclusion. In what ways can ACM contribute to making tech (both industry and academia) a more hospitable environment for all who are interested in the field? ACM-W has expanded our work considerably in the last five years, including increasing our collaborations with other organizations. We commit to help in whatever ways we can as ACM extends and increases its organizational commitment to diversity in computing. 

While women are heading in the front doors of companies, they are hemorrhaging out the side and back doors.

Valerie Barr, a CS professor at Union College, Schenectady, NY, is chair of ACM's Council on Women in Computing, ACM-W.

Copyright held by author.

ACM Europe Conference

Barcelona, Spain | 6 – 8 September 2017

The ACM Europe Conference, hosted by the Barcelona Supercomputing Center and located on the Universitat Politècnica de Catalunya campus, aims to bring together computer scientists and practitioners interested in exascale high performance computing and cybersecurity.

The High Performance Computing track includes a panel discussion of top world experts in HPC to review progress and current plans for the worldwide roadmap toward exascale computing. The Cybersecurity track will review the latest trends in this very hot field. High-level European Commission officials and representatives of funding agencies are participating.

Keynote Talk by ACM 2012 Turing Award Laureate **Silvio Micali**

Co-aligned meetings:

- ACM Europe Celebration of Women in Computing: **WomENCourage 2017**
(Requires registration, <https://womencourage.acm.org/>)
- The European Extreme Data & Computing Initiative (EXCDI)
- Eurolab-4-HPC
- The European Network on High Performance and Embedded Architecture and Compilation (HiPEAC)

Conference Chair: Mateo Valero, Director of the Barcelona Supercomputing Center

Registration to the ACM Europe Conference is free of charge.



<http://acmeurope-conference.acm.org>



Vinton G. Cerf

DOI:10.1145/3055094

A Genetic Theory of the Silicon Valley Phenomenon

WHAT IS IT about the residents of Silicon Valley that encourages risk taking? I have often wondered about that and have reached an interesting, if possibly controversial conclusion. Thinking more generally about immigration, I considered my own family history. In the mid-late 1800s, my father's family emigrated from the Alsace-Lorraine region (variously French and German) to Kentucky. A great many families came to the U.S. during that period. It is a family belief that my great-grandmother, born Caroline Reinbrecht, brought the idea of "kindergarten" from Germany to her new home. My grandfather, Maximilian Cerf, was an engineer and inventor.

Many Silicon Valley residents are also immigrants and their innovative talent and willingness to take risks have been abundantly demonstrated in the past several decades. On the other hand, we hear that risk taking and tolerance for (business) failure is less common in Europe despite the fact that many of the successful Silicon Valley entrepreneurs (and the rest of the U.S.) are from that region. This leads me to think that emigrants are quintessential risk takers. Moving to a new country and, potentially, a new language and culture, surely involves risk. To be sure, some emigrants, especially those coming to America in the 1600s, were fleeing persecution and that has continued to be the case to this day for a portion of those arriving here. Their emigration was and is driven as much by necessity as a willingness to take risk. Those left behind were presumably less inclined to take risk and

have passed their genetic tendencies to their descendants. Hence, the stereotypic risk averseness of the European population. I emphasize this is a stereotype that is plainly not universal and may not even be credible.

Think, however, about the westward movement of the 19th century. The families that moved to the American Midwest and the West were taking enormous risks. The journey was arduous, long, and made the more hazardous by potential encounters with Native American tribes that were understandably resistant to what they saw as invaders of their land. And yet, they came, settled, raised their families, farmed, ranched, started new businesses, and contributed to the expansion of the U.S. across North America.

So we come to a possible explanation for this phenomenon. The emigrants brought with them a gene pool that predisposed them to risk taking. That this is not entirely preposterous, is underscored by a 2009 article that I encountered that speaks directly to this topic.^a The authors conclude:

"Results demonstrate that financial risk seeking is correlated with the *5-HTTLPR* and *DRD4* functional polymorphisms."

I don't pretend to grasp all the implications of that statement other than to conclude there is evidence for a genetic component to risk seeking (or at least risk tolerant) behavior. We hear the term "Yankee Ingenuity," which was

originally associated with emigrants and settlers in the American Northeast but has come to refer more generally to a common stereotype of American inventiveness. Someone making the trek to the West, arriving where enterprises were scarce to nonexistent, had to make do with whatever was at hand or could be invented on the spot.

The 19th century was also the period of the Industrial Revolution in Europe, America, and elsewhere. The term "revolution" is appropriate given the extraordinary creativity of the period. The steam engine, railroads, telegraph, telephone, electrical power generation, distribution and use, electrical appliances including the famous light bulb, were among the many, many other inventions of that era. As we approach the end of the second decade of the 21st century, we can look back at the 20th and recognize a century of truly amazing developments, especially the transistor and the programmable computers derived from it. While it would be a vast overstatement to ascribe all this innovation to genetic disposition, it seems to me inarguable that much of our profession was born in the fecund minds of emigrants coming to America and to the West over the past century.

I celebrate this phenomenon and hope we can keep alive the daring of entrepreneurs, teaching our children to embrace risk, to tolerate failure and to learn from it, regardless of their genetic heritage. □

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.

^a C.M. Kuhnen and J.Y. Chiao. Genetic determinants of financial risk taking. *PLOS ONE* 4, 2 (Feb. 11, 2009), e4362, doi: 0.1371/journal.pone.0004362

Consider Indirect Threats of AI, Too

ALAN BUNDY'S VIEWPOINT "Smart Machines Are Not a Threat to Humanity" (Feb. 2017) was too limited in light of the recent accomplishments of artificial intelligence (AI). Reducing the entire field of AI to four "successful AI systems"—DeepBlue, Tartan Racing, Watson, and AlphaGo—does not give the full picture of the impact of AI on humanity. Recent advances in pattern recognition, due mainly to deep learning, for computer vision and speech recognition have achieved benchmarks comparable to human performance;² consider AI technologies power surveillance systems, as well as Apple's Siri and Amazon's Echo personal assistants. Looking at such AI algorithms one can imagine AI general intelligence being possible throughout our communication networks, computer interfaces, and tens of millions of Internet of Things devices in the near future. Toward this end, Deepmind Technologies Ltd. (acquired by Google in 2014) created a game-playing program combining deep learning and reinforcement learning that sees the board, as well as moves the pieces on the board.¹ Recent advances in generative adversarial learning will reduce reliance on labeled data (and the humans who do the labeling) toward machine-learning software capable of self-improvement.

It is not because four well-known AI applications are narrowly focused by design that smart machines are not a threat to humanity. This is a false premise. Smart machines are a threat to humanity in indirect ways. Intelligence runs deep.

References

1. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* 518 (Feb. 26, 2015), 529–533.
2. Xiong, W. et al. Achieving human parity in conversational speech recognition. *arXiv* (Feb. 17, 2017); <https://arxiv.org/abs/1610.05256>

Myriam Abramson, Arlington, VA

Author Responds:

Abramson misses my point. AI systems are not just narrowly focused by design,

because we have yet to accomplish artificial general intelligence, a goal that still looks distant. The four examples I included are the ones most often cited to illustrate AI progress. Her reaction illustrates my main point—that it is all too easy to erroneously extrapolate from spectacular success in a narrow area to general success elsewhere. That leads to real danger to humans, but not to humanity.

Alan Bundy, Edinburgh, Scotland

Gustafson's Law Contradicts Theory Results

The article "Exponential Laws of Computing Growth" by Peter J. Denning and Ted G. Lewis (Jan. 2017) cited Gustafson's Law (from John L. Gustafson's "Reevaluating Amdahl's Law," May 1988) to refute Amdahl's Law. Unfortunately, Gustafson's Law itself contradicts established theoretical results.

Both Amdahl and Gustafson claimed to quantify the speedup t_1/t_N achievable by N processors, where $N > 1$, t_1 is the time required to solve a computational problem by using one processor, and t_N is the time required to solve the same problem using N processors. Gustafson, in an attempt to interpret experimental results, said

$$t_1/t_N = s + N(1 - s),$$

where s , $0 < s < 1$, is the proportion of time spent by a single processor on serial parts of the program. Gustafson claimed this equation should replace Amdahl's Law as the general speedup rule for parallel and distributed computing.

The sequential running time for finding the maximum of n integers $t_1(n) \leq cn$, accounting for $n - 1$ comparisons and, in the worst case, $n - 1$ assignments, where c is a positive constant. Based on Gustafson's equation, the time to find the maximum of n integers by using N processors would be

$$t_N(n) \leq cn/(s + N(1 - s)),$$

which is bounded by a constant for any

N proportional to n , and approaches 0 for any fixed n if N approaches infinity.

However, in 1982 Cook and Dwork¹ provided an $\Omega(\log n)$ lower bound for finding the maximum of n integers allowing infinitely many processors of any parallel random-access machine (PRAM) without simultaneous writes.

In 1985 Fich et al.² proved an $\Omega(\log \log n)$ lower bound for the same problem under the priority model, where the processor with the highest priority is allowed to write in case of a write conflict. The priority model is the strongest PRAM model allowing simultaneous writes.

Nevertheless, Denning and Lewis were right that Amdahl's Law is flawed. Contrary to Amdahl's assumption, it has already been demonstrated (though without reference to Amdahl) that, in theory, no inherently sequential computations exist. Even though sequential computations (such as sequential concurrent objects and Lampport's bakery algorithm) may appear in concurrent systems, they have negligible effect on speedup if the growth rate of the parallel fraction is higher than that of the sequential fraction.

References

1. Cook, S.A. and Dwork, C. Bounds on the time for parallel RAM's to compute simple functions. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing* (San Francisco, CA, May 5–7). ACM Press, New York, 1982, 231–233.
2. Fich, F.E., Meyer auf der Heide, F., Ragde, P., and Wigderson, A. One, two, three ... infinity: Lower bounds for parallel computation. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (Providence, RI, May 6–8). ACM Press, New York, 1985, 48–58.

Ferenc (Frank) Dévai, London, U.K.

Authors Respond:

Gustafson's Law gives bounds on data-parallel processing whereby the same operation(s) is applied in parallel to different data elements. The standard serial algorithm for finding the max of n elements cannot thus be parallelized, and Amdahl's Law says no speedup. However, finding the max can be computed in parallel by applying the compare

operation to pairs of numbers in $\log(n)$ rounds, yielding a speedup of $n/\log(n) \leq n(1-p)$, consistent with Gustafson's Law. Our point was that not that all algorithms are parallelizable through data parallelism, but, rather, data parallelism currently contributes to the exponential rise in computing power because so many cloud-based operations fall into the data-parallel paradigm.

Peter J. Denning and Ted G. Lewis,
Monterey, CA

Embed the 'Social' in Application Software, Too

In their article "Industrial Scale Agile—From Craft to Engineering" (Dec. 2016), Ivar Jacobson et al. described how software development should be moved from "craft to engineering," but their proposed method completely ignored consideration of what I would generally call the "social embedding" of application software. As software is increasingly integrated into our daily lives, surrounding us in smart homes, smart cities, smart medical devices, and

self-driving cars, socio-technical concerns arise not only due to privacy and security concerns but also to legal constraints, user diversity, ergonomics, trust, inclusion, and psychological considerations. This is not just a matter of conventional requirements engineering. Many such considerations are related to abstract legal, social, ethical, and cultural norms and thus demand a difficult translation from abstract normative level to concrete technical artifacts in the software. A crucial point is such concerns of social embedding could lead to conflicting software design requirements and thus should be addressed in a systematic, integrated development process. The results of such a transformation determines the acceptance and acceptability of application software. From this perspective, Jacobson et al. remain in the "old world" of software engineering, without proper attention to the changing role of application software for users like you and me.

Kurt Geihs, Kassel, Germany

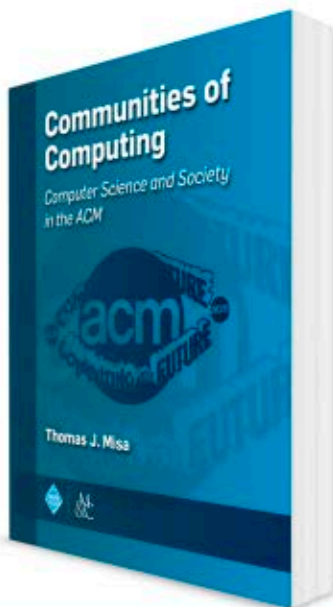
Authors Respond:

Geihs is absolutely right in that Essence, the new standard we explored in our article, is not directly concerned with what he calls "social embedding," but that was intentional. Essence represents a common ground for software engineering in general, and its developers have been very conservative in what they have included. However, Essence is designed to be extended by any known specific set of practices, including human-centric, techno-centric, and user-centric. Since Essence is small yet practical, no one familiar with it would be surprised if it also could serve as a platform for the kind of practices Geihs is looking for.

Ivar Jacobson, Ian Spence,
and **Ed Seidewitz,** Alexandria, VA

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.

© 2017 ACM 0001-0782/17/04



Your first book-length history of the ACM.

**Defining the Discipline
Broadening the Profession
Expanding Research Frontiers**

Thomas J. Misa (Editor)

Charles Babbage Institute (University of Minnesota)

Featuring insightful profiles of people who shaped ACM, such as Edmund Berkeley, George Forsythe, Jean Sammet, Peter Denning, and Kelly Gotlieb, and honest assessments of controversial episodes, this volume deals with compelling and complex issues involving ACM and computing. This is not a narrow organizational history. While much information about the SIGs and committees are presented, this book is about how the ACM defined the discipline, broadened the profession, and how it has expanded research frontiers. It is a permanent contribution to documenting the history of ACM and understanding its central role in the history of computing.



ISBN: 978-1-970001-84-6 DOI: 10.1145/2973856

<http://books.acm.org>

<http://www.morganclaypoolpublishers.com/misa>

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:10.1145/3048379

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

Crafting a National Cyberdefense, and Preparing to Support Computational Literacy

John Arquilla considers how we should interpret the alleged Russian cyberattack on the U.S. Presidential election; Mark Guzdial describes the potential benefits of a 'computing lab.'



John Arquilla The Real Lesson of the Alleged Russian Hack

<http://bit.ly/2hMB9yb>
December 19, 2016

What a pity that senior leaders in the American government and intelligence community have decided to play political football with the alleged Russian hacks of John Podesta's and other Democrats' email. By using these intrusions to gin up fears about the "integrity" of the electoral process—which is already befouled by the focus on finding and spreading dirt on the opposition—the real story is being neglected. And what is that real story? It is that, despite more than two decades of consistent public warnings that have reached the highest levels of government, cybersecurity throughout much of the world is in a shameful state of unpreparedness.

Take the U.S., for example. Since the mid-1990s, there have been ap-

proximately 200 cybersecurity bills brought before Congress. Only one has passed, quite recently at that, and it only calls for voluntary information-sharing about cyber incidents. Legislation aside, there have also been several government-sponsored commissions and top-level exercises focused on understanding and illuminating the cyber threat. Each of these has signaled that "the red light is flashing"; that is, American cybersecurity is in very poor shape. Indeed, former cyber czar Richard Clarke and Robert Knake, in their book, *Cyber War* (<http://amzn.to/2jEymX3>), list the U.S. as having the poorest cyberdefenses among the leading developed countries.

The situation around much of the rest of the world is not much better, as the cost inflicted upon societies—not to mention the wide social and political disruption caused by hack attacks—is staggering. In a speech at the American Enterprise Institute in 2012,

General Keith Alexander, then head of the National Security Agency and the Cyber Command, reckoned annual global losses at more than \$1 trillion. As he put it, this was the "largest [illicit] transfer of wealth in human history." [Full disclosure: I have worked for General Alexander, and continue to do so for Cyber Command.] The situation has only become worse.

Whatever the American role in global leadership in other areas might be, when it comes to cybersecurity, Washington has been sadly lacking. Even now, in the wake of the alleged Russian hacks, leadership, right on up to the president, has decided to focus upon retaliatory action, rather than on beefing up security. My previous post (<http://bit.ly/2enZtrl>) made the point that deterrence based on punitive threats and actions will simply not work, so I won't repeat my lines of argument. But I will reiterate that the failure of the deterrence paradigm, when applied to cy-

berspace, means that the world must move decisively toward an emphasis on improving defenses. And it's not rocket science; better use of strong encryption, moving data around in the Cloud, and increasing use of the Fog, all these can make the situation much better.

But the most important lesson to be learned from the hapless John Podesta is that you can't wait for government policy to protect you. Cyberspace is not just the world at your fingertips; it is also a wilderness, and a dangerous one at that. Much as major commercial firms and governmental bodies must improve their own cybersecurity, individuals, too, must bear responsibility for their own security. The situation is somewhat like that described by the historian Frederick Jackson Turner, who thought of the U.S. as a society defined by its long "frontier experience." Americans were always pushing on into the wilderness, and developed a great deal of self-reliance when it came to sustenance and security. So it may be now in the virtual wilderness of cyberspace.

The alternative, reliance on government, is likely to be fraught with political bickering, endless delays, and unsatisfactory results; in the world's most democratic countries, at least. Authoritarians, on the other hand, have quickly adopted strong cybersecurity policies. As Clarke and Knake see such matters, they list North Korea as having the best cyberdefenses in the world, with China and Russia not far behind.

Perhaps, then, the true lesson of the election hack kerfuffle is not to keep making hard-to-prove charges against President Putin, but to look more closely at how he, and others of his ilk, have crafted their countries' cyber defenses.



Mark Guzdial
Designing the Activities
for a 'Computing Lab'
to Support
Computational
Literacy

<http://bit.ly/2kTtyza>

October 17, 2016

When I was growing up, my elementary school had a "Reading Lab," and later, so did my children's elementary school. If students were struggling with a particular reading difficulty, they could go to the lab and get help with just those specific aspects. It didn't matter what grade

they were in (though earlier grades were certainly most common). Reading was considered so important that it was worth having special help in reading.

The book *Proust and the Squid: The Story and Science of the Reading Brain* (<http://amzn.to/2kTvIyN>) contains interesting insights into what reading experts do to help students overcome challenges in learning to read. For example, learning to read with rhymes is easier for students because they can attend to just the initial sound and only decode the final sound once. Reading out loud rhyming words like "mat" and "rat" and "sat" are easier than "cat" or "pat" (with a hard consonant at the start) because the initial sounds (e.g., "ma") can be extended ("mmmmmmaaaaa") while the student works to decode the final sound and put it all together.

Schools provide extra help in other areas of literacy that are highly valued.

► At the Georgia Institute of Technology (Georgia Tech), we have special help in writing. For example, if a student is having trouble organizing an essay, instructors in a "Writing Lab" teach techniques like using whiteboards in novel ways to brainstorm and develop an outline.

► I am a fan of the Math Emporium at Virginia Tech (<https://www.emporium.vt.edu/>), which is not just for remedial math help, but does help students to learn mathematics at a pace that works for them.

It is becoming obvious that computing is a necessary skill for 21st-century professionals. Expressing ideas in program code, and being able to read others' program code, is a kind of literacy. Even if not all universities are including programming as part of their general education requirements yet (<http://bit.ly/29NbjFK>), our burgeoning enrollments suggest that the students see the value of computational literacy.

We also know that some students will struggle with computing classes. We do not yet have evidence of challenges in learning computation akin to dyslexia. Our research evidence so far suggests that all students are capable of learning computing (<http://bit.ly/2cqaqcD>), but differences in background and preparation will lead to different learning challenges.

One day, we may have "Computing Labs" where students will receive extra

help on learning critical computational literacy skills. What would happen in a remedial "Computing Lab"? It's an interesting thought experiment.

I predict one thing that *won't* happen: students won't just program all the time. Learning to program by programming is a high cognitive-load activity (<http://bit.ly/2ktg8fa>). Students can learn a lot about reading and writing programs by engaging in a variety of other learning activities.

Some of the activities that we might expect:

► Parson's Problems (<http://bit.ly/2jEvhGv>), which are programming problems where the solution is given but the lines of code are scrambled on "refrigerator magnets." Students have to assemble the lines into place. There are never any syntax errors, so students can focus on the meaning of the code. We know that these problems have much lower cognitive load and are useful in learning (<http://bit.ly/2kTObLI>).

► Explaining programs from one student to another, aloud. There is a reading activity called reciprocal teaching (<http://bit.ly/2ks0v8b>) in which one student reads, and the other probes the understanding of the first student. A similar activity could be constructed for developing program understanding skills.

► Tracing programs by hand with pen and pencil. We teach a variety of sketch-based techniques to facilitate learning and practice in mathematics and science classes, from long division and "borrowing/carrying" in multi-digit arithmetic, to balancing equations in algebra and chemistry and drawing free-body diagrams in physics. Certainly, we will need similar sketch-based techniques to help students make sense of their code and data structures, too.

The exercise of defining a "Computer Lab" is not just speculation about a possible future. It helps us as computing teachers to think about what else we can do in our own classes *today* to help struggling students. We need a wide variety of teaching and learning techniques to achieve the goal of "CS for All" (<http://bit.ly/2kSGKas>). □

John Arquilla is professor of defense analysis at the U.S. Naval Postgraduate School; the views expressed are his alone. **Mark Guzdial** is Director of Contextualized Support for Learning at the Georgia Tech College of Computing.

© 2017 ACM 0001-0782/17/4 \$15.00

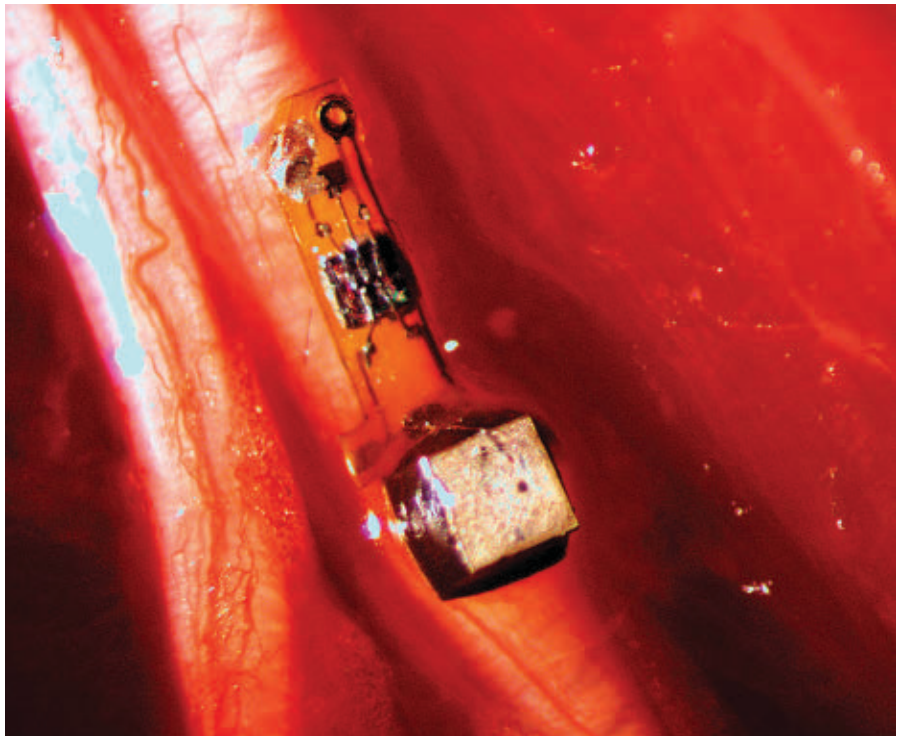
Sensors on the Brain

Implantable wireless monitors give researchers a new look inside the human body.

MORE THAN FIVE million Americans suffer from congestive heart failure, and one out of five die within the first year of contracting the disease. The most common way to treat heart disease is through medication, but the particular drugs and specific doses vary based on the patient. Each patient's condition changes with time, and doctors have limited information about what is happening inside the heart, so they are forced to adjust dosages or regimens based on the patient's behavior, exercise regimen, weight changes, and other factors. "They're trying to manage this complicated system of medications and disease in a rudimentary way," says electrical engineer Nader Najafi, founder of Integrated Sensing Systems Inc.

Najafi's company is developing the Titan implantable hemodynamic sensor (IHM), a device the size of a pencil eraser that can be implanted in the heart of a patient to measure critical variables such as temperature, and then wirelessly transmit this data to a secure database. The Titan device could give caregivers the detailed information they need to adjust medication regimens precisely and improve patient health.

The Titan is one of just two such



Neural dust: A sensor attached to a nerve fiber in a rat. Once implanted, the 3mm-long batteryless sensor is powered by ultrasound.

IHM devices that have completed clinical trials, but a wide range of wireless implantable sensors are in development inside and outside academia. Researchers are designing and testing devices that could be embedded in different parts of the body to measure pressure, temperature, acidity, and

even electrical activity. A group at the University of California, Berkeley, recently reported results in animal models demonstrating the effectiveness of its neural dust, millimeter-scale devices that reside on nerves, measuring and relaying their electrical signals.

These tiny devices may be the first in

a wave of new implantable sensors that change the way doctors and scientists gather information about the human body. “We are starting to bring to reality some of these more science-fiction ideas about how you can build electronic systems that interface with the body in fundamentally different ways than were possible in the past,” says John Rogers, an electrical engineer at the University of Illinois at Urbana-Champaign.

A Chip in the Ocean

The need for wireless implantable sensors is not just about data scarcity; it also stems in part from the flaws of their wired predecessors. In cases such as deep brain stimulation, in which an implanted electrode is wired to a device on the skull, the wires themselves can present an infection risk; often, they need to be surgically removed, too. This increases both the expense of the procedure and the risk of complications. Existing sensors are not always biocompatible, either, so they can trigger a foreign-body response, effectively inducing the host’s immune system to attack them.

The human body simply is not a hospitable environment for electronics. “We are basically giant bags of salt water,” explains electrical engineer Michel Maharbiz, one of the leaders of the neural dust project at the University of California, Berkeley. “If I were to say I’m going to take a chip and throw it into the ocean and that it has to operate under the ocean for 25 years, most people would realize that’s pretty hard.”

An effective, long-term implantable sensor therefore has to be strong enough to resist the corrosive effects of the environment. It also has to be small and biocompatible, so it will avoid interfering with the body’s normal functioning or provoking an immune response. These requirements put strong constraints on the design of the sensors, the materials used, and more.

Untethered Sensors

At the University of Illinois at Urbana-Champaign, Rogers has led a multidisciplinary and institutional team focused on building biocompatible sensors for the brain. The idea stemmed in part from neurosurgeons working with patients who suffered from severe traumatic brain in-

“If I think forward ... I think the amount of integration between what we today consider synthetic and what we consider organismal or biological is going to be extremely high.”

jury (TBI). “One of the first things the neurosurgeon will do is insert sensors for pressure and temperature because those two parameters are critically important,” he says. “If the values fall outside a narrow, healthy range, there can be brain damage.”

Current wired sensors capture the necessary data, but they need to be surgically removed, so Rogers and his group designed a device that can be left in place and absorbed into the body. The millimeter-scale device is flexible rather than rigid to be more biocompatible, and can be adapted to sense a number of variables, including fluid flow, temperature, and pressure. In the most recent study in animals, the miniature sensor is attached to degradable wires, but the group also demonstrated that the sensors would work when combined with an implanted, wireless data transmitter, eliminating the need for wires entirely.

The neural dust group has adopted a different approach to data transmission. Their package consists of a small electronic mote, which is surgically implanted near a nerve or muscle, and an external, handheld ultrasound transducer. The mote is roughly two millimeters long and one-half millimeter wide, and packs a piezoelectric crystal, a transistor, and two electrodes. There is no onboard power source. To activate the mote, the researchers press a transducer against the skin, which sends out six quick bursts of ultrasound. The transducer then switches modes to receive signals, effectively listening for the pulses to rebound. When these sound

waves strike the mote, the piezoelectric crystal captures some of the energy, while some of it bounces back toward the transducer.

In the absence of neuronal activity, those rebounded signals will look roughly the same each time. But if the nerve being monitored fires during this process, the two electrodes pick up that electrical signal. The attached transistor captures this jolt, which modulates the current already flowing through the piezoelectric crystal from the ultra-sound. Once this current changes, the amount of energy reflected back to the handheld transducer changes as well. After some computation to filter the received ultrasound pulses, the external devices tease out the strength of the neuronal signal. “Because you’re listening, you can back out what’s happening at the neurons,” Maharbiz says.

Although this version is coated with epoxy, Maharbiz and the group are now developing one coated with a biocompatible thin film that could function in the body for 10 years without degrading.

Yet another key consideration is to avoid disrupting function at the site of the sensor—to protect the body’s systems, as well as the electronic ones. Najafi and the team behind the Titan sensor say this is a major factor, since they have been testing their device in a crucial region, the left side of the heart. Their goal is to measure filling pressure as an indicator of cardiovascular health, or how well the heart is pumping. In this setting, though, building something that is biocompatible is more challenging. “One element of biocompatibility is the material you use, but the element that is much more important is whether your device is disturbing the blood flow or not.”

The Titan, a cylindrical device with a pressure-sensing module at one end, must be surgically implanted, but in a small human trial of 20 patients, the device was inserted so that only the pressure-sensing tip of the cylinder was exposed to blood flow. The rest of the device was buried in surrounding tissue, and they went through a number of design iterations to minimize possible spots where bacteria could accumulate.

Once implanted, the Titan could wirelessly trigger alerts through a

wand-like telemetry device that transmits information to a patient database. A rapid rise in filling pressure, for example, could be a warning sign for an arrhythmia, but if a healthcare professional were alerted in real time, he or she could proactively adjust the patient's medicine or schedule an appointment. "If patients have to be monitored by a specialist every day, that's not practical," Najafi says. "This way, you can look at the data over the last two weeks, and based on that, you can adjust the medications."

The Future

The timeline for when these sensors become a regular part of patient care is unclear, but the researchers paint a fascinating picture of their potential. Najafi hopes he and his group will be able to build devices that could be safely implanted in children with severe heart problems and last 30 to 50 years. Other scientists are designing

wireless implants that will be application-agnostic. A multidisciplinary group at Brown University, for example, is building a high-throughput device that could potentially work with any type of sensor.

The neural dust group has taken a similar approach, but Maharbiz says it is also moving closer to its original goal of building devices that continually read neuronal activity in key parts of the brain, and allow subjects to control their prosthetics as if they are their own limbs. Indeed, the group has published a proof-of-concept study showing the motes could be shrunk to half the width of an average human hair, which would open up a whole new realm of possibilities.

"If I think forward and dream a little bit," says Maharbiz, "I really do think the amount of integration between what we today consider synthetic and what we consider organismal or biological is going to be extremely high." ■

Further Reading

Carmena, J.M., Maharbiz, M.M., et. al.

Wireless Recording in the Peripheral Nervous System with Ultrasonic Neural Dust. *Neuron* 91, 529-539, 2016.

Yin, M., Borton, D.A., et. al.

Wireless Neurosensor for Full-Spectrum Electrophysiology Recordings during Free Behavior. *Neuron* 84, 1170-1182, 2014.

Rogers, J., et. al.

Bioresorbable Silicon Electronic Sensors for the Brain. *Nature* 530, 4 February, 2016.

Baranowski, J., Delshad, B., Ahn, H.C.

An Implantable Pressure Sensor for Long-term Wireless Monitoring of Cardiac Function – First Study in Man. *Journal of Cardiovascular Diseases & Diagnosis*, Vol. 4, Issue 4, 2016.

Ledet, E., et. al.

Elementary Implantable Force Sensor for Smart Orthopaedic Implants. *Advances in Biosensors and Bioelectronics*, Volume 2, Issue 4, 2013.

Gregory Mone is a Boston, MA-based writer and the author of the novel *Dangerous Waters*.

© 2017 ACM 0001-0782/17/4 \$15.00

Milestones

USACM on Algorithmic Bias, Accountability

Algorithms, the set of instructions computers employ to carry out a task, influence almost every aspect of society. The explosive growth of data collection, coupled with increasingly sophisticated algorithms, has yielded a significant increase in automated decision making, as well as a greater reliance on algorithms in human decision making. Industry forecasters believe software programs incorporating automated decision making will only increase in the coming years as artificial intelligence becomes more mainstream.

One of the major challenges of this emerging reality is to ensure that algorithms do not reinforce harmful and/or unfair biases.

Examples of potential algorithmic bias include:

1. Job web sites: Do these sites send more listings of high-paying jobs to men than to women?
2. Credit reporting bureaus: Does the dataset that algorithms weigh in determining credit scores contain prejudicial information?
3. Social media sites: What factors determine the news items that are served up to users?
4. The criminal justice

system: Are computer-generated reports that influence sentencing and parole decisions biased against African Americans?

Recognizing the ubiquity of algorithms in our daily lives, as well as their far-reaching impact, the ACM U.S. Public Policy Council (USACM) has issued a statement and a list of seven principles designed to address potential harmful bias. The goals of the statement include providing context for what algorithms are, how they make decisions, and the technical challenges and opportunities to prevent and mitigate potential harmful bias.

USACM is the focal point for ACM's interaction with U.S. government organizations, the computing community, and the public in matters of U.S. public policy related to computing and information technology.

The USACM statement (available in full at http://www.acm.org/binaries/content/assets/public-policy/2017_usacm_statement_algorithms.pdf) asserts that these principles should guide every phase of software system development and deployment. "Algorithmic bias can occur even

with the best of intentions," said USACM Chair Stuart Shapiro. "This is, in part, due to the fact that both software development and its products can be complex and produce unanticipated results. Following these principles cannot guarantee that there will be no biased algorithms or biased outputs. But they will serve to keep computing professionals on the lookout for ways biases could creep into systems and provide guidelines on how to minimize the potential for harm."

The Statement on Algorithmic Transparency and Accountability was designed to be consistent with ACM's Code of Ethics. The effort was initiated by USACM's Algorithmic Accountability Working Group.

USACM is organized around a committee structure. Each member of USACM serves on at least one committee. Policy statements originate at the committee level before being approved by the full USACM Council. USACM's seven committee areas are: Privacy, Security, Intellectual Property, Law, Accessibility, Digital Governance, and Voting. In June, the Council approved

the addition of three new working groups to reflect the continuing rapid evolution of the technology landscape: Internet of Things (IoT), Big Data, and AI/Algorithmic Accountability.

ACM TO CELEBRATE 50 YEARS OF TURING AWARD

During the next several months, ACM will celebrate 50 years of the ACM A.M. Turing Award and the visionaries who have received it.

The aim is to highlight the significant impact of the contributions of the Turing Laureates on computing and society, to look ahead to the future of technology and innovation, and to help inspire the next generation of computer scientists to invent and dream.

The celebration will culminate with a conference on June 23–24 at the Westin St. Francis in San Francisco, with moderated discussions (streamed in real time) exploring how computing has evolved and where the field is headed.

More information and registration for this event is available on the Turing Award 50 website, <http://www.acm.org/turing-award-50>.

Digitizing the World

Digital maps trawl for real-time updates.

REAL-TIME COMMUNICATION and collaboration lie at the heart of a new generation of high-definition (HD) digital maps that react quickly to changes in the real world. Autonomous vehicles and construction-site surveys are among the applications that are driving companies toward high-precision mapping performed almost in real time.

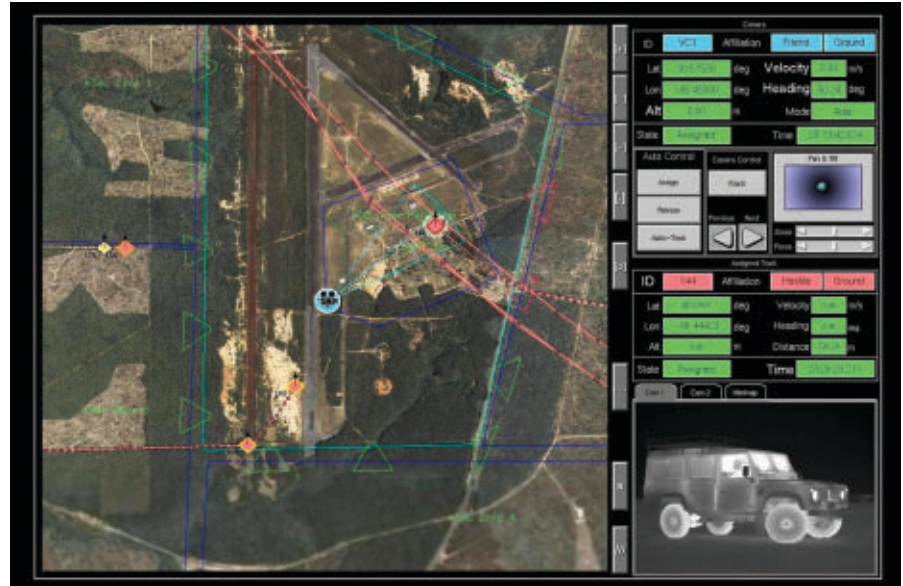
James Dean, founder and director of technology applications at London, U.K.-based startup SenSat, says, “Digitizing the world is incredibly important. We can make better, faster decisions from that digitized information than is possible with traditional means”

Early adopters of SenSat’s mapping technology come from the road-construction industry, a sector that today mainly relies on manual surveys conducted at ground level. Surveys can take as long as six weeks and, as a result, can only be performed infrequently during a project. A pillar built a foot out of place that is only discovered at a relatively late stage can set the project back weeks, but by sending out small, lightweight drones as flying cameras to scan the site on a daily basis, project managers can spot mistakes in the three-dimensional (3D) map long before they become a costly issue.

Regulations currently limit the area that drones can cover, which suits focused construction projects such as bridges over freeways and intersections. However, larger-scale applications, such as a proposed expansion of the capacity of one of the U.K.’s busiest roads—the M4 motorway—will require surveys to be conducted over many miles and months.

“Today, you have to keep the drone within the line of sight of the operator; that limits you to coverage of around 1km² per hour. For economical use, you really want 20km² per hour,” Dean says.

Still, the U.K. government is beginning to look favorably on changes



Command and Control Technologies Corp.’s C3I Surveillance Toolkit provides real-time tracking and geo-referencing of targets of interest, as well as providing control and monitoring for a network of sensor systems.

to regulations that would give survey drones greater autonomy to support projects such as the M4 expansion, he says. “At the moment, it’s a supportive environment. We think it will only get better from here.”

Users of these roads, as vehicles become more autonomous, will need similarly detailed mapping to be carried out on a near-real-time basis. Although an autonomous vehicle could scan only its surroundings to see where it can drive, practical systems will use HD maps to perform the task of localization. “Using the map, we figure out where we are on the road,” says Jen-Hsun Huang, cofounder and CEO of graphics-processor company nVidia. “We want to test whether our understanding of the world is consistent with what is around the car.”

Filipe Mutz, associate professor of information systems at Federal University of Espírito Santo (IFES) in Vitória, Brazil, says the level of detail required in the map depends on its intended use: “For feature-based localization systems, a map can be represented by a sparse set of features, and it is not

necessary to store a detailed 3D representation of the environment. For collision-free motion planning, on the other hand, a denser representation of the occupied and free areas is necessary.”

Mutz says the resolution of the grid used in the experimental vehicle-based mapping system built by a team at IFES is set to 20cm. “We plan to reduce the map resolution to 15cm in the near future, and our group considers 10cm the ideal resolution for safe operation in highly cluttered urban areas at reasonable speeds,” he adds.

Mapping companies such as Google, HERE, and TomTom are using fleets of similar vehicles armed with cameras and LIDAR sensors to map the roads they travel. Still, the level of detail needed for motion planning requires an immense effort.

Alain De Taeye, head of high-definition (HD) mapping at TomTom, said at the European leg of nVidia’s GTC technology conference in Amsterdam: “We have mapped 47.1 million kilometers of road, but only 120,000 kilometers is mapped in HD. And we are leading the pack. We have basic information

for many of the roads today—that covers 70% of developed society—but we need to go much further and faster. You don't want to miss centimeters in a self-driving car.

“At first, people believed a navigable map was unaffordable; now, they think HD mapping is unaffordable. It's not unaffordable; you have to be clever about it,” De Taeye says.

Through a deal with nVidia, TomTom aims to accelerate HD map creation by applying more advanced artificial intelligence (AI) algorithms running on graphics processors in the vehicles themselves, as well as in the cloud. Huang says nVidia's approach employs AI to work out the difference between trees, buildings, and other vehicles. “We detect all these for two reasons. Number one: we want to continuously update our map, and there are several different types of marker we can use to figure out where we are. Number two: we're detecting where it's safe to drive [in real time].”

With the front end of the mapping technology deployed in vehicles, TomTom and others want to gather mapping data from many vehicles to support a continuously updated map. “Crowdsourcing will be helpful. Otherwise, it would take many thousands of one's own vehicles, like Google's, and many miles of driving, resulting in high cost, to achieve these HD maps—and they would not be up to date, either,” says Kevin Mak, senior analyst for Strategy Analytics' global automotive practice.

Marco Lisi, engineering manager for global navigation systems at the European Space Agency (ESA), points to handheld gadgets as rich sources of mapping data. “Whenever we carry around a smartphone, we are carrying around several sensors: a compass, accelerometer, gyroscope, and GPS. We are collecting data on our location all the time with several sensors at once,” Lisi says.

Such real-time data streams already feed back into location-driven applications such as the StreetBump app used by the City of Boston, which collects data on potholes from users—the app forwards data from the motion sensors in smartphone handsets when the vehicle they are riding in hits a bump. The Waze app uses regular updates on

the location of smartphones carried by its clients as they drive around in their cars; data from the app not only highlights traffic jams, but lets the host servers estimate the number of lanes on a freeway based on the geographic spread of pings across the road collected over time.

Eric Gunderson, CEO of MapBox, sees similar data aggregated on a large scale being used to perform precision mapmaking. “What we're getting every single day is data that represents 100 million miles of traveling. As it comes in, it just looks like noise, but as I analyze the data, the algorithm can discern the actual lanes on the highway. You can drill down this crowdsourced data all the way to put center lines on the road so I can drive the car as if it was on rails.”

A lack of standards for representing sensor data even for dedicated vehicle systems will make crowdsourcing more difficult in the short term, Mutz says. “Nowadays, most vehicles have different sensor setups and there are no established standards for the storage and distribution of that data.”

Although mapping organizations are likely to embrace standards to allow them to incorporate data from many sources, a fully open ecosystem seems unlikely, says Strategy Analytics' Mak, who says the companies involved will prefer to maintain semi-closed ecosystems.

Some of the standards used to exchange mapping data will be mandated by government: agencies such as the U.S. National Highway Transportation Safety Administration (NHTSA) see the potential for crowdsourced data to improve traffic safety, as well as map accuracy. A car can send messages to nearby vehicles if it detects a pedestrian moving toward the road, or passes a vehicle signaling that it intends to turn across oncoming traffic.

Working with the U.S. Department of Transportation, the NHTSA said in 2015 it was speeding up plans to mandate the adoption of vehicle-to-vehicle (V2V) communication. Based on a version of the Wi-Fi local-area network protocol adapted to work in a dedicated frequency band around 5.9GHz to limit interference, V2V allows cars to share data on the environment around them. If a car detects a pedestrian moving toward the road or passes a vehicle

signaling that it intends to turn across oncoming traffic, it can send messages to the vehicles following behind.

The communication need not be limited to vehicles. Says Maurice Gerjets, senior director of chipmaker NXP Semiconductors, “There will be cameras at intersections that send V2X signals.” Such smart intersections will be able to indicate whether nearby cars need to slow down for a red signal or warn that a car has blocked an exit. Temporary roadside beacons will alert vehicles to the presence of roadside workers, and that can be added to local maps temporarily.

“Collaboration across many technologies is very important,” says Dean.

Yet the vehicles and their mapping software will need to be alert to the possibility of hacking, and of collaborators in data being less than honest. Lars Reger, chief technology officer of NXP's automotive division, says without effective security, “I could put a little beacon in front of my house and transmit to the world that an accident has happened, and clear the road outside.” **■**

Further Reading

Seif, H.G., and Hu, X.

Autonomous Driving in the iCity – HD Maps as a Key Challenge of the Automotive Industry, *Engineering: The Official Journal of the Chinese Academy of Engineering and Higher Education Press*, Vol. 2, Issue 2, June 2015, pp159-162

Carrera, F., Guerin, S., and Thorp, J.

By the People, for the People: the Crowdsourcing of 'StreetBump,' an Automatic Pothole Mapping App, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, Volume XL-4/W1, 29th Urban Data Management Symposium (2013)

Harding, J., Powell, G., R., Yoon, R., Fikentscher, J., Doyle, C., Sade, D., Lukuc, M., Simons, J., and Wang, J.

Vehicle-to-vehicle communications: Readiness of V2V technology for application, *National Highway Traffic Safety Administration Report No. DOT HS 812 014.*

Mutz, F., Veronese, L.P., Oliveira-Santos, T., de Aguiar, E., Auat Cheein, F.A., and De Souza, A.F.
Large-Scale Mapping in Complex Field Scenarios Using an Autonomous Car, *Expert Systems with Applications*. Vol. 46, 15 March 2016, pp439-462

Chris Edwards is a Surrey, U.K.-based writer who reports on electronics, IT, and synthetic biology.

© 2017 ACM 0001-0782/17/4 \$15.00

Computing the Arts

Artists can use software to create art, and some software creates art all on its own.

IT IS NOT unusual to hear a student is taking an advanced placement computer science (AP CS) course these days, but eyebrows raise when Jacqueline Mendez tells people about it, because Mendez is a senior at Boston Arts Academy where, as the name implies, the emphasis is on the arts.

“I had a free block, and I was surprised how [computer science is] more than just systems and machines and the Internet,” explains Mendez, who plans to major in physics in college. “People have a mind set that it’s machines, but it’s really not. It’s what the world is right now. We use computer science for everything.”

Her friend and classmate Michelle Romero is also bullish on AP CS, thanks primarily to their teacher, Nettrice Gaskins, who is also director of the school’s STEAM (science, technology, engineering, art, and mathematics) lab. The lab’s mission is to help teachers and students explore the connections between the arts, science, and math, and to incorporate new technology into their projects.

Romero, who plans to major in theater in college, says she became interested in computer science after seeing Gaskins help with projection and lighting from her laptop for a school play. “I thought ‘oh wow, theater can be three dimensions,’ and adding projections and sensors made it really cool and made the audience enjoy it more,” Romero recalls. “It sparked my interest in wanting to know how ... a sensor connected to her computer could make movement and projections.”

Mendez and Romero recently created a short video (<https://www.youtube.com/watch?v=uQp-5DdyKGQ>) blending technology, physics, and jazz improvisation. The idea was to interpret Einstein’s Theory of Relativity with musician John Coltrane’s jazz using animation and film to create a music visualization based on that intersection. “Anything can be simplified, so instead



Algorithmic innovation engines are capable of producing images that are not only given high confidence scores by a deep neural network, but are also qualitatively interesting and recognizable.

of doing the mathematical equation where your brain gets confused, we did animations and used computer science technology,” says Mendez. Using Adobe Flash, “Our video explained how Coltrane used his saxophone to explore the connection of jazz and velocity and acceleration, which are vectors. Then we said any moving objects, including sound waves, can be described by the magnitude of its velocity and acceleration as well as direction.”

Gaskins’ class includes students studying visual arts, theater, and music, and all “also have an interest in computer science and are courageous enough to take the step to enter something they’re not familiar with,” she says. “There are new ways to expand computer science and the arts in ways I can’t imagine, and

that’s the type of flexibility we have.”

With an increasing number of software programs available to help people create art, the question becomes: is it culturally acceptable for a neural network or machine learning system to produce original artwork?

Jeff Clune thinks so. Clune, an assistant CS professor and director of the Evolving Artificial Intelligence Lab at the University of Wyoming, says there are many reasons why we might want neural networks to produce art. “The main one is because we consider artistic expression as one of the most uniquely human traits. If we want to produce artificial intelligence that rivals human intelligence, that should include art.”

Another reason for artificial intelligence (AI) to produce art is that “it might



Images produced with innovation engines were not only accepted to a selective art competition and displayed at the University of Wyoming Art Museum, but they also were among the 21% of submissions that won an award.

fuel us to be more creative.” Being exposed to extremely different influences can be inspiring, and is often essential to producing breakthroughs, he adds. “What better way, save for visiting alien cultures, is there to experience truly different ways of thinking and artistic expression than letting AI produce art? I predict it will unlock vast new cultural and scientific landscapes for humans to explore.”

The art produced by networks has greatly informed our scientific understanding of how deep neural networks see the world, says Clune, and they do so very differently than humans do.

He references the Turing Test, and how computers must not only produce art, but “produce art at a level of quality and originality such that it is indistinguishable from art produced by a human” to pass the test. Clune decided to test that theory by submitting art the lab’s deep neural networks (DNNs) produced to the University of Wyoming’s 40th Annual Juried Student Exhibition, which accepted 35.5% of the submissions.

Clune and his team developed an AI “artist” agent and a “critic” or judge agent. The purpose was for the artist to try and produce art the critic/judge would like, he explains. To do that, the team created an innovation engine algorithm with the artist and judge agents inside. The critic was a DNN trained to recognize 1,000 categories of images, Clune says, “and it’s the AI artist’s job to produce images that the critic agrees count as each type of im-

age,” like a motorcycle.

“We have found that it’s important for artificial intelligence, while it’s learning, to have to constantly switch goals and have many goals,” he explains. That is why they challenged the AI artist to make 1,000 different types of images, instead of just one. The team went through the images the artist agent produced and submitted a handful for the competition they thought appeared to show interesting artistic interpretations of concepts, such as a beacon or a prison cell.

There were no rules forbidding such entries, he notes, “largely because I’m sure the competition organizers didn’t realize it was possible for AI to produce high-quality art. In fact, if the rules for art competitions begin to be rewritten to limit submissions to humans, we will know that AI has become such a great artist that many humans are afraid to

“We have found that it’s important for artificial intelligence, while it’s learning, to have to constantly switch goals and have many goals.”

compete against it.”

The outcome? Not only were the images accepted, but they were also among the 21.3% of submissions to receive an award. The work was then displayed at the university’s art museum. Clune adds that the judges were evaluating all submitted art on its own merits to see which was good enough to be accepted. “I imagine the judges never considered the fact that the art might have been made by AI.”

Music to the Ears

Perry Cook, professor emeritus of computer science at Princeton University, believes art has been combined with technology longer than many people realize. “The march of music has been about technology. Music has, in some cases, brought about new technologies, and certain music has taken advantage of the technology of the day,” says Cook, who is also a research coordinator and IP Strategist for SMule, a social network based on creating and sharing music, and co-founder and executive vice president of Kadenze, an online arts and technology education startup. In a blog post (<http://bit.ly/2feVCyd>), Cook cited numerous artists who have created artwork using machine learning, which he says opens up all kinds of new possibilities.

Although a common perception is that computer music came along late in the evolution of electronic music, according to Cook, people were looking at using computers to store and analyze music in the 1950s. “Forward to today, the computer is just another technology, in my opinion. It’s somewhat comparable to putting a valve on a bugle so you could play more notes, [which] allowed music to get louder and softer.”

The arts are an especially good way to introduce people who normally would not have thought they should or would want to learn concepts about computer science, he notes, specifically artists, children, and underrepresented minority groups “who think they’re kind of locked out or aren’t interested and don’t want to be engineers or computer scientists. Getting to them through the arts is a way to open up [computer science] to a whole new community.”

Making CS “Culturally Responsive”

That is exactly what Gaskins is doing at

Gaskins' approach to her AP CS class is "culturally responsive computer science," using examples from different cultures to create computational artifacts.

the academy. She feels strongly that the way to engage her students who come from various ethnic and socioeconomic backgrounds, in science and math is through STEAM. Her approach to her AP CS class is "culturally responsive computer science," a concept that uses examples from different cultures to create computational artifacts, she says. Gaskins got the idea from a program at Rensselaer Polytechnic Institute (RPI) that shows teachers and students how to see algorithms from a cultural perspective, using Culturally Situated Design Tools (CSDT), and thus, learn principles of engineering, science, math, and computing.

During research Gaskins conducted as a doctoral student, she noticed "there were certain groups that were less represented" in STEAM or STEM, women being key among them. There were also several ethnic groups that were underrepresented: African Americans, Latinos, and native Americans or indigenous groups, she says.

A colleague of Gaskins' created a software tool to look at local or cultural knowledge and merge it with CS, and saw positive results, "which led me to believe that the content is the problem; the content isn't engaging the populations that aren't well represented." Teaching CS with culturally relevant or responsive content, she believes, makes a huge difference.

Her course is not designed to be a traditional computer course, she emphasizes, and she teaches the principles using 3D modeling, visual arts, music—anything a student might find relevant.

Ron Eglash, a professor in the departments of Science and Technology Studies and Computer Science at RPI,

developed CSDTs as a way to teach anyone how to create their own simulations and artifacts. In a TED Talk ("The Fractals at the heart of African designs," <http://bit.ly/2e5l0ne>), Eglash explained his focus in the U.S. was on African-American, Native American, and Latino students. "We've found statistically significant improvement with children using this software in a mathematics class in comparison with a control group that did not have the software. So it's really very successful teaching children that they have a heritage that's about mathematics, that it's not just about singing and dancing."

For Gaskins, creating art through culturally response computer science is not only acceptable, it is an imperative if the U.S. is to stay competitive in STEM, as well as STEAM.

"If a student has no exposure to computers and they enter a situation where there's nothing in a course that's familiar to them, they're likely to drop the course," she says. "There's a lot of people who don't know they can be interested because they don't know anything about it, and if you unlock that for them and show them how relevant it is no matter their culture ... you're opening the door for them to come into the subject." **■**

Further Reading

Nguyen, A., Yosinski, J., and Clune, J.
Deep neural networks are easily fooled: High confidence predictions for unrecognizable images, *Computer Vision and Pattern Recognition (CVPR '15)*, IEEE, 2015

Ahmed, S.U., Camerano, C., Fortuna, L., Frasca, M., and Jaccheri, L.
Information Technology and Art: Concepts and State of the Practice, Norwegian University of Science and Technology, Norway, <http://bit.ly/2eYgBBq>

Bond, G.,
Software as Art, Communications of the ACM 48(8), 2005, pp. 118-124

Mitchell, W.J., Inouye, A.S., and Blumenthal, M.S.
Beyond Productivity: Information, Technology, Innovation, and Creativity, The National Academies Press, 2003.

Machin, C.
Digital artworks: bridging the Technology Gap, Proceedings of the 20th Eurographics UK Conference, IEEE Computer Society, 2002.

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

© 2017 ACM 0001-0782/17/4 \$15.00

ACM Member News

LOOKING TO COMPUTERS TO HELP MAKE SENSE OF THE WORLD



"I first encountered computers in high school when my physics teacher got a

Commodore PET and I started playing with it," says Paul Dourish, Chancellor's Professor of Informatics at the University of California, Irvine. "I had been headed for a medical degree, but became so fascinated by the computer that I switched directions and enrolled in computer science at the University of Edinburgh" in Glasgow, Scotland.

After earning his undergraduate degree with a double major in Artificial Intelligence and Computer Science from the University of Edinburgh, Paul went to work for Rank Xerox EuroPARC in London, while working on a doctorate in computer science from University College, London.

After completing his Ph.D., Dourish moved to the U.S., where he took on research positions at Apple, and then Xerox PARC, before joining the Information and Computer Science faculty at the University of California, Irvine, in 2000.

Dourish is focused on human-computer interactions, particularly the social implications of information technology. "Silicon Valley and start-up culture gives us a certain type of idea about what innovation looks like," Paul says.

He is particularly interested in how innovation gets imported into different spaces, and different countries, such as India and China. The ways in which new models of data-driven innovation pertain to managing urban development and social informatics are now his focal point.

"My work is getting more and more into the sociology and anthropology of data and computation, looking at computers and computation as a way of people making sense of their world and acting in it."

—John Delaney

Cybersecurity

SINCE ITS INAUGURATION in 1966, the ACM A.M. Turing Award has recognized major contributions of lasting importance to computing. Through the years, it has become the most prestigious award in computing. To help celebrate 50 years of the ACM Turing Award and the visionaries who have received it, ACM has launched a campaign called “Panels in Print,” a collection of responses from Turing laureates, ACM award recipients and other ACM experts on a given topic or trend.

ACM’s celebration of 50 years of the ACM Turing Award will culminate with a conference June 23–24, 2017, at the Westin St. Francis in San Francisco. This unique event will highlight the significant impact of ACM Turing laureates’ achievements on computing and society, to look ahead to the future of technology and innovation, and to help inspire the next generation of computer scientists to invent and dream.

For our second Panel in Print, we invited 2002 ACM Turing laureate **LEN ADLEMAN**, 2014 ACM Prize in Computing recipient **DAN BONEH**, 2015 ACM Grace Murray Hopper Award recipient **BRENT WATERS**, and ACM Fellows **PATRICK MCDANIEL** and **PAUL VAN OORSCHOT** to discuss current issues in cybersecurity.

The cybersecurity discipline has developed rapidly. Do you think we are staying ahead of, or falling behind, the threats?

LEN ADLEMAN: I think that we are behind. Cybersecurity is a cat-and-

mouse game. There can never be a final victory. The Internet is developing so quickly, along so many paths, that while we address current problems, we cannot even anticipate those that are emerging.

BRENT WATERS: In the research realm of cryptography, we have made significant leaps in the past 15 years in terms of which new functionalities we can realize. These include solutions to problems such as identity-based encryption, attribute-based encryption, and fully homomorphic encryption, and potentially can realize an exciting primitive called indistinguishability obfuscation.

Where we seem to be facing problems is filling in the gap between sound cryptography and sound deployments. These deployments can fail for any number of reasons from bad software implementation, to poor design of new cryptography, to use of legacy cryptographic protocols.

What do you see as the top cybersecurity threats in 2017 and why?

DAN BONEH: Social engineering attacks remain one of the top cybersecurity issues in 2017. Phishing and related attacks are still effective at stealing user credentials. Targeted emails continue to be effective at fooling end users into installing unwanted software such as adware, malware, or ransomware. These are common occurrences, and are often the easiest way to gain a foothold on a targeted system. Two-factor authentication and application whitelisting can make the attacker’s

job harder. They are tools to be used as part of a broad defense strategy.

PATRICK MCDANIEL: There’s been an interesting transition of threats and attacks over the last 10 years, and what we’re seeing more frequently is professional attacks that more effectively monetize the vulnerabilities in computers. In particular we have seen the rise in things like ransomware, which has become a very serious problem for businesses, government agencies, and organizations that don’t have full-time professional cybersecurity staff.

Just looking at what is happened over the last six months in the U.S., it’s clear that misinformation has become a major weapon in the cybercriminal’s arsenal. I think we will see even more attacks where misinformation is used to try and shape public policy, sway public opinion or even to alter people’s behaviors. Obviously the use of misinformation is nothing new—we’ve seen it before with stock market manipulation, etc.—but I think we’re going to see much newer and inventive uses of misinformation as a means of cyberattack.

PAUL VAN OORSCHOT: It’s a long-standing problem: software vulnerabilities allowing compromise of user devices and remote control of these compromised machines. This is easily addressed in theory, but harder to fix in the real world. Problems stem from long-ago and deeply entrenched architectural choices in operating systems; use of software applications which favor rich functionality over conservative



Brent Waters



Dan Boneh



Len Aldeman



Patrick McDaniel



Paul Van Oorschot

design; and the freedom allowing users to download and install software with a single click makes us all vulnerable to being socially engineered to install software without any sound evidence or knowledge of what the software will do. It's not likely that we will be able to retrain 10 million software developers, nor to motivate them to voluntarily spend extra time on security when the economic benefit of doing so falls to others. That's an economics of information security challenge—as is that fact that historically, almost all software is sold on an as-is basis without liability for consequences.

Why are cyberthreats/attacks becoming more sophisticated with each passing year?

BRENT WATERS: One can attribute this to two basic reasons. First, technology in general becomes better and more sophisticated over time. One would expect the sophistication of cyberattacks to also flow in that same direction. Another important factor is that with more and more data stored on computing devices, the value in launching attacks increases. For example, there have been multiple attacks that exposed private communications and photos of celebrities. Ten years ago, without smartphones, these photos either wouldn't be taken or wouldn't be accessible. As another example, in this election cycle we have seen that attacks (for example, the DNC emails) have the potential to shake up organizations and possibly shift outcomes in elections. This type of power will not only interest the usual attackers, but will also attract extremely well-funded state sponsored adversaries.

With the public more concerned about cybersecurity than ever before, what should be the top cybersecurity priorities for the new U.S. administration?

DAN BONEH: The highest priority for the new U.S. administration is to shore up the cyber defenses of government systems. Events like the 2015 attack on the office of personnel management that exposed the personnel records of over 21 million people, or the compromise of the IRS systems that may have exposed personal data from over 700,000 taxpayers, should not happen again.

PATRICK MCDANIEL: Our federal IT systems, as we have learned repeatedly,

“The Internet is developing so quickly, along so many paths, that while we address current problems, we cannot even anticipate those that are emerging.”

are very much antiquated, due to things like underfunding. But if our society is to become more secure, then we need to focus on updating and fixing those federal systems. One way in which we could do this would be for the current administration to immediately prioritize creating a national two-factor authentication system, either for federal employees or even more broadly. Although that sounds somewhat boring, I think that is the single simplest thing we can do to reduce the threats against the information systems we have in this country. A good friend and colleague of mine, Farnam Jahanian, the Provost of Carnegie Mellon University, has said, “We are not good at doing the easy things, and we need to get better at them.”

What are the biggest challenges faced by industry in defending against cyberattacks, and what technologies/approaches can help them overcome these challenges?

LEN ADLEMAN: I think the issues raised in the question transcend industry. From my forthcoming book, *Memes: How Genes, Brenes and Cenes Shape Your Life and Will Shape the Future of Humanity:*

We will soon see religions, nations, and economies rise and fall in cyberspace. These entities will be no less powerful and have no less impact on our lives than their current “brick and mortar” counterparts. Political, economic, and even military power will be

diffuse; the physical locations of like-minded people will be less important than their numbers and connectivity.

If the U.S., Russian, and Chinese governments are not working on black hat programs that, in the event of war, will knock out the computational infrastructure of the other two, they aren't doing their jobs. Such programs are weapons of mass destruction, and, if used, the death toll could be colossal. A first world country with no computational infrastructure is a country with no economy, no food, no power and ultimately not a country at all.

What are your biggest security concerns as they relate to the influx of connected devices in the Internet of Things (IoT)?

PATRICK MCDANIEL: When it comes to IoT and the future of security, I have a vision of two possible futures. The first scenario comes at a significant cost. But I believe this to be the more optimistic future, because we will understand the trade-off between cost and value, and we're going to pay for it so we can live in a world in which we have much better security than we do today.

The second, and in my view, the more pessimistic scenario is a world in which we have just become used to insecurity. There is a kind of really toxic resignation among some members of the cybersecurity research community as well as industry and government, that today's systems are unfixable and that we don't have the technology, time or resources to make ourselves more secure. I think this is a particularly dim and uncomfortable scenario, not only because the kinds of benefits we see from technology would be greatly diminished, but our potential for changing life on this planet—from healthcare, to society, to communications, to quality of life to energy efficiency, to protecting the environment—will be vastly diminished, if we just accept insecurity. ■



DOI:10.1145/3055275

Ineke Buskens

Global Computing Online Social Networks and Global Women's Empowerment

Mediating social change or reinforcing male hegemony?

SOCIAL NETWORKS LIKE Facebook hold promise for women regarding personal growth and social emancipation that physical spaces do not offer, yet virtual and physical spaces are intertwined in intricate ways. Explorations into new forms of selfhood and social life that emerge in and through social networks will be inevitably brought into relationship with traditional dispositions and practices that may be hostile to that change. When online discourses represent a challenge to 'traditional' gender relations, the way in which Facebook management mediates online disputes can have profound offline consequences for sexual and social emancipation.

In this regard, the story Kiss Brian Abraham tells about Zambian women creating Facebook groups to initiate conversations about sex is pertinent.¹ Zambia's culture of male hegemony is defined by the supremacy of Cisgender heterosexual masculinity; Christian principles of women's chastity have merged with traditional understandings of women's submissiveness to men to frame female sexuality as a

means to satisfying husbands' needs and not as a woman's human right. Furthermore, in the time that these Facebook pages in Zambia were created (2010–2013), women were attacked and stripped naked in the streets by mobs of male assailants who were citing Christian and cultural principles whilst claiming that the women were wearing sexually provocative clothing.¹

In their Facebook groups the Zambian moderators laid down their own cyberspace rules: persons who solicited sex or posted pornography would be deleted, the use of foul and insulting language was forbidden, mindfulness of each other and respect for others' opinions was encouraged and discriminatory gender norms that aimed to subordinate women were explicitly critiqued.¹ In creating spaces for women to experience and express their sexuality in ways that would not be possible in the traditional private and public sphere, these Facebook pages were therefore nothing short of revolutionary. Abraham asserts that these Facebook pages, apart from being pertinent to gender equality and women empowerment, have done

something revolutionary for Zambian society as a whole: "engaging with each other on the basis of shared interests instead of traditional identity is an act of critiquing the traditional status quo and opens pathways for others to follow."

Inevitably these pages evoked a strong response and Facebook itself was among those to respond. Linda Waleka Manda, the moderator of *Real Adults Talk with Waleka*, shared her experiences: "I started a page in 2010 and I would openly talk about sex, but then certain people found it offensive for a woman openly discussing sexual issues in public on the Internet, so they reported me to Facebook and my first account was closed."¹ Because several of her pages were reported to Facebook and she feared them to be closed again, despite the fact that these pages did not violate Facebook's community standards, Linda started to create secret groups.

It can be argued that Facebook, in giving in to the objectors' demands, sided with Zambia's male hegemonic social order and implicitly supported that order's perspective that women should not have sexual agency. This



nist twist” to friends’ icons.⁶ To be more than ‘window dressing’, social networking managers must appreciate that it is impossible to have a gender-neutral stance in a gender-imbalanced world and that commitment to social justice requires commitment to becoming gender aware. It might not be the goal but it can be the outcome of social networking sites to exacerbate gender discrimination. A gender-aware stance requires changes in design of moderation policies and systems. Facebook and other social media platforms should align their actions and systems to support the empowerment of women, and not to support the patriarchal social order. To do this they must distinguish postings in terms of intent. Sexual content posted for emancipatory purposes is not comparable to sexual content posted for soliciting or selling sex that is often degrading of women and female bodies, even when it seems similar. Intent is underdetermined by words and images. The meaning and purpose of postings should be informed by the values that drive the posting and the context in which it happens. Such design might require an upgrade of the systems that screen postings and reporting to involve human, social and computing elements and their interrelationships. Designing for social change is more challenging than designing for social harmony. Why should that hold computer scientists back? **□**

References

1. Abraham, K.B. Sex, respect and freedom from shame: Zambian women create space for social change through social networking. In I. Buskens and A. Webb, Eds., *Women and ICT in Africa and the Middle East: Changing Selves, Changing Societies*. Zed Books, London, 2014, 195–207.
2. Best, M.L. The Internet that Facebook built. *Commun. ACM* 57, 12 (Dec. 2014), 21–23.
3. Chemaly, S. Facebook’s big misogyny problem. *The Guardian* (2013); <http://bit.ly/2lzsvUN>
4. Dent, G. The day Facebook banned Clementine Ford. *Women’s Agenda* (June 22, 2015); <http://bit.ly/2l2rmMV>
5. Facebook. Facebook Community Standards (2015); <http://bit.ly/1fcql6I>
6. Herr, A. Facebook gets a feminist twist with new friends icons, (July 8, 2015); <http://bit.ly/1eFpZrK>
7. Women, Action, and the Media. ‘Open Letter to Facebook’ (May 21, 2013); <http://bit.ly/2m9hVa9>

Ineke Buskens (ineke@unu.edu) is an independent research, evaluation and gender consultant working internationally with emphasis on Africa and the Middle East. She has been the main editor of the two books published by the Gender Research into ICT for Women Empowerment in Africa and the Middle East (GRACE) Network. Kiss Brian Abraham is the GRACE Zambia team lead.

Copyright held by author.

would make Facebook complicit in the real-world crimes that happened in the aftermath of the pages’ closures. In one case, a male involved the police, found out where the owner of the site in question worked, and attempted to scandalize her at her workplace. In another case, a student at the National Institute for Public Administration had her Facebook identity taken over, and naked pictures were posted. Lewd behavior is a crime under Zambian law, so she could have been physically arrested. There is evidence that this attack was orchestrated by a man whose sexual advances she refused. She had to change her real name, losing part of her identity and sense of historical self.

Facebook’s response to the Zambian pages is not an isolated event. Other cases have been reported where Facebook took down pages critical of women’s sexualization and discrimination, while sustaining pages that normalize gender-based violence.⁴ It is alleged that Facebook only responds to take-

down requests that challenge male hegemony if not responding would involve economic risk, for instance when advertisers distance themselves from Facebook—something that happened after a coalition of women’s organizations released an open letter in 2013⁷ stating that they did not want to be associated with images depicting violence against women.³ Facebook may have been motivated in the Zambian case by respect for cultural norms and practices, taking the lead as to what postings were acceptable or not from what it considered as community consensus. But many cultural contexts are sexist, so such a stance could also amount to colluding with gender discrimination. Facebook and other social networking sites say they want to contribute to making the “world more open and connected.”^{2,5} If so, they must be sensitive to how deeply male hegemony impacts online spaces, and how online sexism forms and informs offline lived realities.

It is a nice gesture to give a “femi-



Kode Vicious

The Chess Player Who Couldn't Pass the Salt

AI: Soft and hard, weak and strong, narrow and general.

Dear KV,

Our company is looking at handing much of our analytics to a company that claims to use “Soft AI” to get answers to questions about the data we have collected via our online sales system. I have been asked by management to evaluate this solution, and throughout the evaluation all I can see is that this company has put a slick interface on top of a pretty standard set of analytical models. I think what they really mean to say is “Weak AI” and that they’re using the term *Soft* so they can trademark it. What is the real difference between soft (or weak) AI and AI in general?

Feeling Artificially Dumb

Dear AD,

The topic of AI hits the news about every 10 to 20 years, whenever a new level of computing performance becomes so broadly deployed as to enable some new type of application. In the 1980s it was all about expert systems. Now we see advances in remote control (such as military drones) and statistical number crunching (search engines, voice menus, and the like).

The idea of artificial intelligence is no longer new, and, in fact, the thought that we would like to meet and interact with non-humans has existed in fiction for hundreds of years. Ideas about AI that have come out of the 20th century have some well-known sources—including the writings of Alan Turing and Isaac Asimov.



Turing’s scientific work generated the now-famous Turing test, by which a machine intelligence would be judged against a human one; and Asimov’s fiction gave us the Three Laws of Robotics, ethical rules that were to be coded into the lowest-level software of robotic brains. The effects of the latter on modern culture, both technological and popular, are easy to gauge, since newspapers still discuss advances in computing with respect to the three laws. The Turing test is, of course, known to anyone involved in computing, perhaps better known than the halting problem (https://en.wikipedia.org/wiki/Halting_problem), much to the chagrin of those of us who deal with people wanting to write “compiler-checking compilers.”

The problem inherent in almost all nonspecialist work in AI is that hu-

mans actually do not understand intelligence very well in the first place. Now, computer scientists often think they understand intelligence because they have so often been the “smart” kid, but that’s got very little to do with understanding what intelligence actually is. In the absence of a clear understanding of how the human brain generates and evaluates ideas, which may or may not be a good basis for the concept of intelligence, we have introduced numerous proxies for intelligence, the first of which is game-playing behavior.

One of the early challenges in AI—and for the moment I am talking about AI in the large, not *soft* or *weak* or any other marketing buzzword—was to get a computer to play chess. Now, why would a bunch of computer scientists want to get a computer to play

chess? Chess, like any other game, has a set of rules, and rules can be written in code. Chess is more complicated than many games, such as tic-tac-toe (a game that is used to demonstrate to another fictional computer in the 1983 film *WarGames* that nuclear war is unwinnable), and has a large enough set of potential moves that it is interesting from the standpoint of programming a winning set of moves or a strategy. When computer programs were first matched against human players in the late 1960s, the machines used were, by any modern concept, primitive and incapable of storing a large number of moves or strategies. It was not until 1996 that a computer, the specially built Deep Blue, beat a human Grandmaster at the game.

Since that time, hardware has continued its inexorable march toward larger memories, higher clock speeds, and now, more cores. It is now possible for a handheld computer, such as a cellphone, to beat a chess Grandmaster. We have had nearly 50 years of human/computer competition in the game of chess, but does this mean that any of those computers are intelligent? No, it does not—for two reasons. The first is that chess is not a test of intelligence; it is the test of a particular skill—the skill of playing chess. If I could beat a Grandmaster at chess and yet not be able to hand you the salt at the table when asked, would I be intelligent? The second reason is that thinking chess was a test of intelligence was based on a false cultural premise that brilliant chess players were brilliant minds, more gifted than those around them. Yes, many intelligent people excel at chess, but chess, or any other single skill, does not denote intelligence.

Shifting to our modern concepts of soft and hard AI—or weak and strong, or narrow and general—we are now simply reaping the benefits of 50 years of advancements in electronics, along with a small set of improvements in applying statistics to very large datasets. In fact, improvement in the tools that people think are AI is, in no small part, a result of the vast amount of data that it is now possible to store.

Papers on AI topics in the 1980s often postulated what “might be possible” once megabytes of storage were

commonly available. The narrow AI systems we interact with today, such as Siri and other voice-recognition systems, are not intelligent—they cannot pass the salt—but they can pick out features in human voices and then use a search system, also based on stats run on large datasets, to somewhat simulate what happens when we ask another person a question. “Hey, what’s that song that’s playing?” Recognizing the words is done by running a lot of stats on acoustic models, and then running another algorithm to throw away the superfluous words (“Hey,” “that,” “that’s”) to get “What song playing?” This is not intelligence, but, as Arthur C. Clarke famously quipped, “Any sufficiently advanced science is indistinguishable from magic.”

All of which is to say that KV is not surprised in the least that when you peek under the hood of “Soft AI,” you find a system of statistics run on large datasets. Intelligence, artificial or otherwise, remains firmly in the domain of philosophers and, perhaps, psychologists. As computer scientists, we may have pretensions about the nature of intelligence, but any astute observer can see that there is a lot more work to do before we can have a robot pass us the salt, or tell us why we might or might not want to put it on our slugs before eating them for breakfast.

KV

Related articles on queue.acm.org

Scaling in Games and Virtual Worlds

January 02, 2009

<http://queue.acm.org/detail.cfm?id=1483105>

A Conversation with Arthur Whitney

<http://queue.acm.org/detail.cfm?id=1531242>

Information Extraction

Andrew McCallum

<http://queue.acm.org/detail.cfm?id=1105679>

The Network Protocol Battle

Kode Vicious

<http://queue.acm.org/detail.cfm?id=2090149>

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and co-chair 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.

Calendar of Events

April 2–6

ASIA CCS '17: ACM Asia Conference on Computer and Communications Security, Abu Dhabi, United Arab Emirates,
Sponsor: ACM/SIG,
Contact: Ramesh Karri,
Email: rkarri@nyu.edu

April 3–4

SOSR '17: Symposium on SDN Research, Santa Clara, CA,
Contact: Dave Levin,
Email: dml@cs.umd.edu

April 3–7

SAC 2017: Symposium on Applied Computing, Marrakech, Morocco,
Contact: Sung Shin,
Email: sung.shin@sdsstate.edu

April 8–12

ASPLOS '17: Architectural Support for Programming Languages and Operating Systems, Xi'an, China,
Contact: Yunji Chen,
Email: eyj@ict.ac.cn

April 18–21

CPS Week '17: Cyber Physical Systems Week 2017, Pittsburgh, PA,
Contact: Radu Marculescu,
Email: radum@cmu.edu

April 20–21

L@S 2017: 4th ACM Conference on Learning @ Scale, Cambridge, MA,
Sponsor: ACM/SIG,
Contact: Claudia Urrea,
Email: calla@mit.edu

April 22–26

ICPE '17: ACM/SPEC International Conference on Performance Engineering, L'Aquila, Italy,
Contact: Walter Binder,
Email: walter.binder@usi.ch

April 23–26

EuroSys '17: 12th EuroSys Conference 2017, Belgrade, Serbia,
Sponsor: ACM/SIG,
Contact: Marko Vukolic,
Email: mvu@zurich.ibm.com

Viewpoint

Wanted: Toolsmiths

Seeking to use software, hardware, and algorithmic ingenuity to create unique domain-independent instruments.

“**A**S WE HONOR the more mathematical, abstract, and ‘scientific’ parts of our subject more, and the practical parts less, we misdirect the young and brilliant minds away from a body of challenging and important problems that are our peculiar domain, depriving these problems of the powerful attacks they deserve.”

—Fredrick P. Brooks, Jr.

Computer Scientist as Toolsmith II³

I have the privilege of working at the Defense Advanced Research Projects Agency (DARPA) and currently serve as the Acting Deputy Director of the Defense Sciences Office (DSO). Our goal at DARPA is to create and prevent technological surprise through investments in science and engineering, and our history and contributions are well documented. The DSO is sometimes called “DARPA’s DARPA,” because we strive to be at the forefront of all of science—on the constant lookout for opportunities to enhance our national security and collective well-being, and our projects are very diverse. One project uses cold atoms to measure time with 10^{-18} precision; another is creating amazing composite materials that can change the way in which we manufacture. We have other programs that aim to reinvent synthetic chemistry, and we even have one program that attempts to reimagine the entire process of scientific discovery itself.

Nearly all of our projects in the Defense Sciences Office are using computing and data to transform our view of fundamental scientific questions. Com-



putation has become the most important tool for reimagining scientific problems since the advent of calculus and codification of the scientific method over 300 years ago. This observation has been made several times over the past decade, from a major issue of *Nature*⁷ in 2006 to the major report by the Computing Community Consortium⁴ in 2016. But, as I look across the people working on all of these and other exciting DSO projects, where are all of the computer scientists? I am actually kind of lonely.

This loneliness has led to a concern that the field of computer science (in particular computer science education) might have a growing problem. In this new wave of interest in computing we may be unintentionally “misdirect[ing] the young and brilliant minds” away from important, use-inspired problems. In Fred Brooks’ Newell Award lecture over 20 years ago, entitled “Computer Scientist as Toolsmith,”² Brooks notes that “a scientist builds in order to study.” As computer scientists we build algorithms and software systems of various kinds in order to study. Yet, in spite

of the expanding set of interdisciplinary opportunities for computer scientists to reimagine with our tools, too few of us venture to build tools in foreign territory and we certainly do not teach the interdisciplinary thinking to our undergraduate computer science students.

I have grown concerned that our educational priorities as a discipline are misplaced. We focus on training students to become productive software developers, which is driven by the current availability of lucrative jobs. Hence, when they complete their education, most students’ experience of computing is limited to what they have encountered in the classroom or their personal life. Employers hand them existing problems or they work on things they know (such as games or social media). The hard and complex problems of scientific, engineering, societal, or national security importance are rarely used to inspire students, let alone to seriously shape curricula, at the undergraduate level.

We need a renewed emphasis on the study of computing phenomena in the physical world and stronger call to service to address issues of societal need. Consider that Hilbert’s Tenth Problem inspired Turing’s theoretical specification of a computing machine. The needs of ballistics experts at the Aberdeen Proving Ground and Manhattan Project scientists wrestling with neutron propagation and explosive “lenses” motivated the need for ENIAC, MANIAC and calculating machines. Computer graphics emerged from the Semi-Automatic Ground Environment (SAGE), a system designed for Cold War air defense. Some readers might recall

that the protocols that became the basis for the World Wide Web were developed specifically to help physicists at CERN share scientific data. Many mainstream computer science subject areas today were born from computing scientists working with domain scientists to build the tools needed to solve specific problems within a non-computing domain.

I love Brooks' vision of the computer scientist as the toolsmith, in which we use software, hardware, and algorithmic ingenuity to create unique instruments. We should make no restriction on what domain those instruments are for. Academic computer science should embrace this broader charter as toolsmith regardless of the domain of inquiry. But what constitutes a legitimate computer science problem, as opposed to "merely an application" for a given discipline? Creating novel algorithms driven by specific needs of material scientists is as much a computer science problem as a new machine learning technique—provided the algorithms or data structures are truly novel from a computer science view and not "merely" an application of an existing idea to the materials problem. This is a use-inspired^{6,8} vision that goes beyond what might today be called "scientific computing" to include the engineering, systems, and national-security related challenges in which computing is vital.

There is a vast opportunity for innovation at the intersection of computing and various disciplines, where we might transform longstanding domain-specific problems using algorithmic thinking. Students and educators do not need to look far to find potential topics for disruption. A report like the National Academies' *Grand Challenges for Engineering in the 21st Century*³ provides 14 civilization-changing challenge problems, the solution to each of which will certainly involve computing. For example, the grand challenge of "Advancing Health Informatics" as a case study. The medical domain has been strangely intertwined with computing over the last half-century mostly due to intrepid health and life scientists who sought out computing as a solution to their challenges. Seminal work in AI planning and knowledge representation came out of medical schools; medical imag-

Academic computer science should embrace this broader charter as toolsmith regardless of the domain of inquiry.

ing challenges created fertile ground for machine vision, among many other issues. Now we are seeing the digitization of the entire healthcare industry, thanks to the push for electronic health records and various healthcare delivery mechanisms. This is not merely an IT deployment challenge, but one of creating an ecosystem of humans and machines to deliver healthcare—and computer science will play a central role as the tool builders that enable us to reimagine this system. Will the innovations come from computing sciences or elsewhere?

At DARPA we have several programs under way that reimagine longstanding problems using computation, including projects in nuclear security (SIGMA^a), engineering design (TRADES^b), and applied mathematics (CASCADE^c)—to name just three, as there are others—all with central computer science challenges. But the issue remains that there are few computer science departments in the country that would look at these topics as central to the discipline of computing. As a discipline we are exceptionally good at teaching software development and theory, but not as good on the use of computational imagination to address such wicked problems.

Perhaps we should work toward a broader view of computing, one that can follow from computing intertwined with use-inspired problems being reimaged as computational ones. To this end, I suggest three rubrics that can be used to extract basic computer

a <http://www.darpa.mil/program/sigma>

b <http://www.darpa.mil/program/transformational-design>

c <http://www.darpa.mil/program/complex-adaptive-system-composition-and-design-environment>

science questions from interdisciplinary problems or those that some might consider to be mere applications.

► Representation and reasoning:

The opening quotation in the famous algorithms text book of Aho, Hopcroft, and Ullman¹ states that "Computer Science is a science of abstraction—creating the right model for a problem and devising the appropriate mechanizable techniques to solve it." A standard exercise in many an 'Introduction to AI' class is the eight queens problem. Represent the problem in the naive way and the solution can take hours; a clever representation results in an instant solution for vastly larger problems. For any domain, representation of the problem and the structures used to solve it is the fundamental issue. Across the landscape of those who use software and computing, many are suffering with outmoded or inappropriate representations. Developing new representations, however, requires that computing scientists embed with the domain scientists and transform these disciplines' issues into digital abstractions.

For those looking for such challenges, consider materials science and engineering, highlighted in recent years by the Materials Genome Initiative (MGI) in the United States. MGI promises to transform materials and manufacturing science from the methods of handbooks and tables to one of direct computational design from functional requirements. Key to achieving this vision is the development of new data structures for representing materials across multiple scales (literally from the scale of atoms to the scale of products); representations of uncertain and sparse data (that is, data from physical tests and simulations is only a tiny fraction of the universe of possible materials); and new modeling paradigms that let designers reason about new materials possibilities. Developing effective data structures will require computer scientists working with materials scientists to understand their problems and how to best represent them.

► Innovation by Algorithm:

We praise and cite those who can produce a better algorithm for a known problem. How about those who can define new problems and, thus, new algorithms? Venturing into a domain, capturing and

bringing back “wild” problems suitable for algorithmic study is not adequately awarded. Hence, the computing problems driven by scientific domains are often developed by the domain scientists, not the computer scientists. This gives rise to an obvious question: Are we sure these are the best algorithms for the tasks at hand? If our answer is yes, then we have a problem: either all domain-driven problems are trivial (very unlikely) or expert algorithms can be developed by nearly anyone without formal computer science training, and our discipline is extraneous. Clearly, there is a big opportunity for domain-driven algorithmic thinking and a need for the broader computer science community to embrace scientific problems and engineering opportunities. Should we fail to do so, we contribute to a balkanization of computing, in which computing is reimagined within each scientific community by non-experts in computing sciences.

A great example of algorithmic innovation comes from DARPA’s MAKE-IT,^d a program that is reimagining the process of synthesizing chemical reaction pathways into a search algorithm. The key insight of MAKE-IT is that chemical reactions can be viewed as nodes in a directed graph: given certain precursor compounds, a reaction produces certain chemical products. New kinds of search algorithms will enable chemists to find new and better methods of producing important compounds.

► **The human-machine symbiosis:**⁵ Computing over the decades has been focused on understanding computing machinery, building it, controlling it and understanding the consequences. A computer, or piece of software, is principally a device that enables a person to do a job, usually an existing job, perhaps in a better manner than before. As electronic spreadsheets have largely replaced accountants’ physical ones, the human-centric task has evolved but not fundamentally changed. Computing machinery is now poised to go beyond just assisting with human problems—it enables us to fundamentally rethink them. This is seen across science and engineering domains in a particularly pointed way: progress has become intimately tied to computation and data.

^d <http://www.darpa.mil/program/make-it>

What other disciplines can we disrupt with computational tools that augment our intelligence?

As this scientific revolution unfolds we are witnessing a shift from merely computational methods (for example, computer as calculator) to those in which humans and machines are partners. Over a decade ago, the Sloan Digital Sky Survey enabled astronomers to pose questions that were previously impossible to answer. Currently, DARPA is working to advance machine reading under our Big Mechanism^e program with the goal of having computers that can help scientists harvest vast collections of experimental results and produce a shared human-machine understanding of certain cancer pathways. An emerging challenge is to consider how problems can be broken up and distributed across human-machine teams in order to exploit the power of silicon-based machines while optimizing the insight and creativity of the carbon-based ones. What other disciplines can we disrupt with computational tools that augment our intelligence?

The computer scientist is the tool-smith that enables domain scientists and engineers to reinvent and reimagine their disciplines as algorithmic or information-centric. However, if computing really is going to be the new framework for scientific discovery and engineering innovation, we—the computer scientists—need to support this kind of interdisciplinary study among students as well as evangelize the unconverted. Sometimes we might find those outside computing with a “not invented here” bias—we need to meet them halfway or more than halfway. Sometimes we may find disciplines are not as much converted to computing, as they are becoming assimilated under the assault of ideas from computing.

^e <http://www.darpa.mil/program/big-mechanism>

Scientific disciplines are refactoring key problems around systems for vast data and computation, and computer scientists should be working deeply with these domain scientists and engineers to bring about revolutions.

Many of the most important problems facing the U.S. and the world today represent a call to service similar to that which began during the Second World War, when many of the best and brightest minds (mostly physicists) in the country put commercial or academic interests on hold and dedicated themselves to innovations in service of national need. Students—pulled by the lure of the Apollo program or the needs of the Cold War—pursued careers of national service in science and engineering in order to solve big problems.

I would love our best and brightest computer science students to feel that pull now, and have the opportunity to be captivated by these major interdisciplinary challenges. But with the current din of opportunities, we must work harder to instill the passion for such service. Computing is no longer a new discipline and is well into adolescence—it needs to make some decisions about what it really wants to be when it grows up. Rather than just disrupting industries and creating the next great mobile app, I long for an expanded view of what constitutes legitimate computer science inquiry and for increased scientific citizenship. ■

References

1. Aho, A.V., Hopcroft, J.E., and Ullman, J.D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1976.
2. Brooks, F.P., Jr. The computer scientist as toolsmith II. *Commun. ACM* 39, 3 (Mar. 1996), 61–68.
3. Grand Challenges For Engineering, The National Academy of Engineering, 2008.
4. Vonavar, V., Hill, M., and Yelick, K. Accelerating science: A computing research agenda: A white paper prepared for the computing community consortium committee of the computing research association. Technical report, CRA/CCC, 2016.
5. Licklider, J.C. R. Man-computer symbiosis. *Institute for Radio Engineers, Transactions on Human Factors in Electronics*, HFE-1:4–11 (Mar. 1960).
6. Shneiderman B. *The New ABCs of Research: Achieving Breakthrough Collaborations*. Oxford University Press, 2016.
7. Steering the future of computing. *Nature* 440, 7083, (Mar. 2006).
8. Stokes, D.E. *Pasteur’s Quadrant: Basic Science and Technological Innovation*. Brookings Institution Press, 1997.

William Regli (regli@darpa.mil) is Acting Deputy Director, Defense Sciences Office, Defense Advanced Research Projects Agency.

Viewpoint

What It Means to Be an Entrepreneur Today

In his keynote address before the fifth edition of the Tech Open Air conference in Berlin in 2016, Kickstarter's cofounder and CEO Yancey Strickler suggests the city's tech community faces "a very rare opportunity."

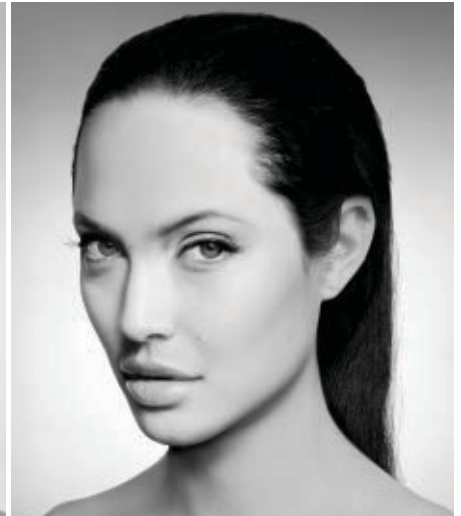
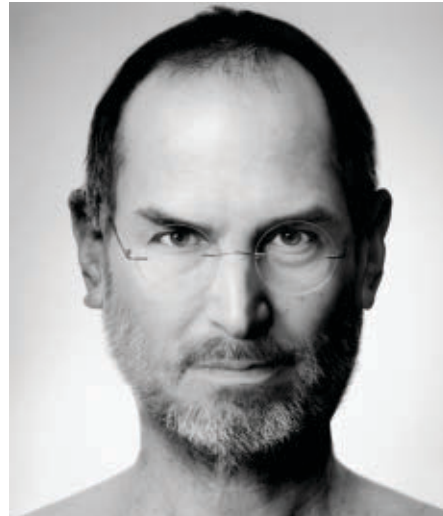
I WANT to talk about some of the values behind Kickstarter and what I think it means to be an entrepreneur.

I wanted to start by talking about a series of works by a Chinese artist named Zhang Wei. These are obviously photographs of very famous, iconic people, except they are actually composite photographs of Chinese people whom Zhang takes portraits of; his thesis is that in this mass-media age, we are defined more by the media's heroes than we even are by ourselves, so each of these photos is a composite of 1,000 different faces, Chinese faces, and it just shows how that is a lot of what defines us (see the accompanying image on this page).

Now if we think about footballers or pop stars or teenagers, we think, "Sure, everyone has their idols that shape the way they think about things," but this is also true in technology. It is important to be aware of the ways we are influenced.

If you look at tech media and the way business is covered today, it is quite violent. It is extremely aggressive: Out for blood. Go to war. Own the world. Companies are judged not by how many jobs they create, but by how many jobs they end; those are the ones that are celebrated. It is very divisive.

At the core of this, of course, is money. By investing money in these companies, people try to make more money for themselves, and ultimately that is the end goal for all these things: infinite



Composite photo portraits of Steve Jobs and Angelina Jolie created by the Chinese artist Zhang Wei.

growth for the desire of infinite riches. Of course, when all of society works this way, the results are horrific. It is inequality. It is an incredibly imbalanced society, and it is one that is increasingly led by a money monoculture of people trying to optimize their money to make more money, and the rest of the world is just a portfolio for them to operate on. You see this across everything.

In music, Ticketmaster monopolizes the concert industry, a diverse ecosystem of labels is disappearing, and most pop songs are written by a handful of bald Scandinavian men. In Hollywood, most of what we see now is sequels or remakes of existing IP, all presented in IMAX and 3D. Every magazine cover

has Taylor Swift on it. And all of these things happen because these are the safe bets. For someone looking to make money on their money, they want safe bets. This is the sort of culture that you get from that, and it is not just the cultural landscape that is shaped; it is even the physical landscape.

I live in New York City. Currently, in New York, in just Manhattan, there are 1,800 physical bank locations. That is up 60% from 10 years ago. You would think the ATM was never invented. The challenge with all these banks appearing on street corners—or chain restaurants, clothing stores, cellphone stores, whatever—is that what had been there before were lo-

**Who Owns
the Social Web?**

Contest Theory

**Toward a Ban on Lethal
Autonomous Weapons**

**Exploiting Vector
Instructions
with Generalized
Stream Fusion**

**DeepDive:
Declarative Knowledge
Base Construction**

**Responsible Research
and Innovation
in the Digital Age**

**Making Money
From Math**

**MongoDB's
JavaScript Fuzzer**

Research for Practice

Plus the latest about bionics in competition, specialized chips for machine-learning, and curing cancer with data.

cal businesses, small shops run by New Yorkers for their community. But commercial real estate has become the new easy way to make money, and so the rents get jacked up. The person that has been there for 30 years gets pushed out, and a Bank of America goes into its place. This is how a city dies. This is how a culture dies, and this is increasingly happening.

Battling Moneyed Imperialism

This is the globalization that is appearing around the world. It is a moneyed imperialism that is changing local cultures. It is really hard to talk about, and it is really important to talk about at the same time. Berlin is feeling this. It seems like any city that has any kind of dynamic culture or where young people live starts to experience this, and it is unclear what to do about it. This is true for entrepreneurs, it is true for cities. This is the big battle of our age. So what can you do about it? I do not have a solution, but I have three places where I think it starts.

The first is, don't sell out.

Now I am 37 years old. I grew up with the Kurt Cobain, "corporate rock sucks," "selling out is the worst thing you could ever do" school of thought, and "selling out" means you have a great idea, and then you use it to personally enrich yourself, and who cares about what happens to it afterward?

The idea of selling out being bad—that has gone out of style. Now, selling out is cool—that is to be celebrated. "Oh, they exited for how much money? Wow, they're super-awesome." It is very shortsighted. It creates a very bottom-line thinking that is not in the broader interest of society, that is just in the self-interest of those entrepreneurs, of those people who hold all that equity.

"Don't sell out" also means you don't sell out your values or your culture to hit a short-term gain; you have a clear set of values that you are operating with.

Taking a big VC (venture capital) funding round is another type of selling out because once you take a big check, you are not able to guide your mission on your own terms; you have to follow somebody else's. To not sell out, it is important that you are sustainable as a business, so you have a source of revenue that allows you to continue to operate and grow. This is an incred-

ibly liberating thing. It lets you call the shots and do what you want to do.

For Kickstarter, we have operated in the black since our 14th month of business; June 2010, we were in the black for the first time, and we have stayed there. We operate within our means. Every dollar we spend, we are quite thoughtful of because this is not a big VC check; this is money that was earned through people having success with our service, so we think very carefully about how we spend that. If you are able to have your own source of revenue, you are able to sustain yourself. You do not have to sell out; you can continue to operate in whatever way you like.

That leads to the second point, which is that you should be idealistic.

When you start a company or any sort of project, you are compelled by some deep, burning thing in your heart; there is something you really want to change. There is something you really want to be different. Those things come from very idealistic places, but the challenge as something exists and is out there in the world is that every day, there are infinite opportunities to compromise on that vision. It could be that a competitor has done something that seems like it might be successful; it is not something you ever thought you would do, but "oh wait, do we have to do this now?" Is this the new term of engagement?

You have to be very careful about your beliefs. You do not want to sink into the morass of industry standards, because industry standards are lowering every single day by everyone trying to chase growth, chase revenue, and stay alive. It allows other people to do the same. You have to lock in your values and the ideals behind your project from the very beginning, at that nascent stage where it is all romance, and you are just dreaming how big, how powerful can this thing be. Think about the things that are important to you at that early moment, and lock them in. Make sure you always stay true to those things, that you act with integrity.

Protecting New Ideas

We were very clear about this for Kickstarter from the very beginning; I and my co-founders Perry Chen and Charles Adler vowed from the beginning that we never wanted to sell Kickstarter, we never wanted to try to IPO.

The idea of selling out being bad—that has gone out of style. Now, selling out is cool—that is celebrated.

We thought about this as a sort of public trust. It made sense to us that in society, there should be a place reserved for new ideas, and that is a place that should be protected. It is not something that should be sullied by overt commercialization or advertising, or any of those sorts of corrupting forces; it should be kept as a place for ideas as pure as it can be.

We have always operated with that mentality. Last year, we took an additional step to legally formalize that, becoming a public benefit corporation. It is a very new thing in the U.S. This legally binds you to have a positive influence on society; you draft a charter to say on what basis that will happen. For us, it was about supporting artistic and creative works, especially those in less-commercial areas. It includes a pledge to not employ legal but esoteric tax avoidance strategies. It includes a pledge to engage with issues facing artists and creators beyond our walls. It includes pledging to donate 5% of our after-tax profits every year to organizations fighting to end systemic inequality, and organizations fighting to provide arts and music education in New York City.

These are the things that are important to us, and we have legally bound the company to follow those principles for as long as it exists. This is how those ideals that were so present in the creation of Kickstarter and are so present in the 120 of us who work there can be maintained throughout Kickstarter's entire life.

The third point is to be generous.

To be perfectly honest, acting out of integrity and values is a harder road in a lot of ways. It means you are not going to have that big check because that VC who asked you whether you would

do anything if we give you this \$40 million, right? You don't do those things. Instead, you are operating within your means. Every choice you have to make is a real choice. It is a slower road; you are not going for hockey-stick growth. You probably won't be huge overnight; it is a longer haul.

In more important ways, it is easier, because in those decisions where there is a moral question about what to do and balancing the needs of the business with what you believe to be right, there is more clarity. There is more freedom to act with conviction.

Being generous is important because it takes a lot to choose this path. It takes a lot of strength, a lot of fortitude, a lot of conviction. We have to support people who choose to do that. We have to support other artists, other creators. You have to support them directly.

People are willing to do this. Look at the statistics for Kickstarter to date; in seven years, \$2.5 billion, 100,000 successfully funded projects, 11-million-plus backers. These are people supporting projects just because they think they are cool, because it is interesting, they like the idea of a world where that exists. This is not for financial self-interest. This is not the money monoculture. This is people just wanting to support someone else who is doing something cool.

There are lots of ways to do that. If you have been listening to someone's album on Spotify or SoundCloud, go to Bandcamp and buy their record. Make sure you give them money. Support independent businesses rather than chains. Choose to spend your money and put your energy toward people, toward artists, toward companies that are behaving in this way. If gravitational forces start to shift in that direction, it will shift culture. It really will.

Kickstarter's internal handbook, when new employees start, includes a lot of our philosophy and thinking about the world, and the final page states: "We champion and celebrate the creation of art and culture."

Our mission is around creative projects. The way we are trying to accomplish this is through encouraging art and creativity and a richer, more diverse culture that is less controlled by money, that is more controlled by individual voices and by audiences. There

are lots of ways to do that; there are lots of ways to challenge the status quo.

A Rare Opportunity

This is the reason I was really excited to come to Berlin. I think this is a very interesting moment in the life of this community. After what happened in the U.K. with Brexit, it seems likely that the gravitational force of technology will shift squarely to Berlin, and that is going to mean a lot of changes. It is probably going to mean a lot more money coming into this community. It is going to mean more people coming here to start things.

At a moment like this, there is a very rare opportunity for this community to define what it means to be a company from Berlin.

If you think about American companies, Silicon Valley companies, there is a lot of focus on hyper-growth, on disruption, things along those lines; a lot of power being amassed on platforms, and users having very little power. I would suggest that for Berlin, there is an opportunity to balance that, to build a code around building products or services or exploring creative projects that empower the individual, that are about decentralizing the Web, that are about giving everyone more power over their data, over what their experience is online, that are about paying creators and supporting people pursuing open source or pursuing projects that are not simply looking to make as much money as possible.

The Internet needs a force like this. It needs a counterbalance. It needs someone looking out for the rights of the individuals, because we are moving toward the future at a very fast rate. Just a few projects here can define a new way of thinking and a new future for the Web.

This is a moment where groups of people can make choices, can draft language, can think about what it means for this city to be producing schools of thought that are shaping the world in a more positive and diverse direction. This is the moment where you can define that, and I encourage you to take advantage of this moment. ■

Yancey Strickler (lena@kickstarter.com) helped to launch the Kickstarter crowdfunding platform in 2009; today, he is its CEO.

Copyright held by author.

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

The use of silicon PUF circuits.

BY MENG-DAY (MANDEL) YU AND SRINIVAS DEVADAS

Pervasive, Dynamic Authentication of Physical Items

AUTHENTICATION OF PHYSICAL items is an age-old problem.³ Common approaches include the use of bar codes, QR codes, holograms, and RFID tags. Traditional RFID tags and bar codes use a *public identifier* as a means of authenticating. A public identifier, however, is *static*: it is the same each time when queried and can be easily copied by an adversary. Holograms can also be viewed as public identifiers: a knowledgeable verifier knows all the attributes to inspect visually. It is difficult to make hologram-based authentication pervasive; a casual verifier does not know all the attributes to look for. Further, to achieve pervasive authentication, it is useful for the authentication modality to be easy to integrate with modern electronic devices (for example, mobile smartphones) and to be easy for non-experts to use.

Identification is not the same as *authentication*.

A public identifier alone cannot distinguish a genuine product from a counterfeit copy, since a public identifier

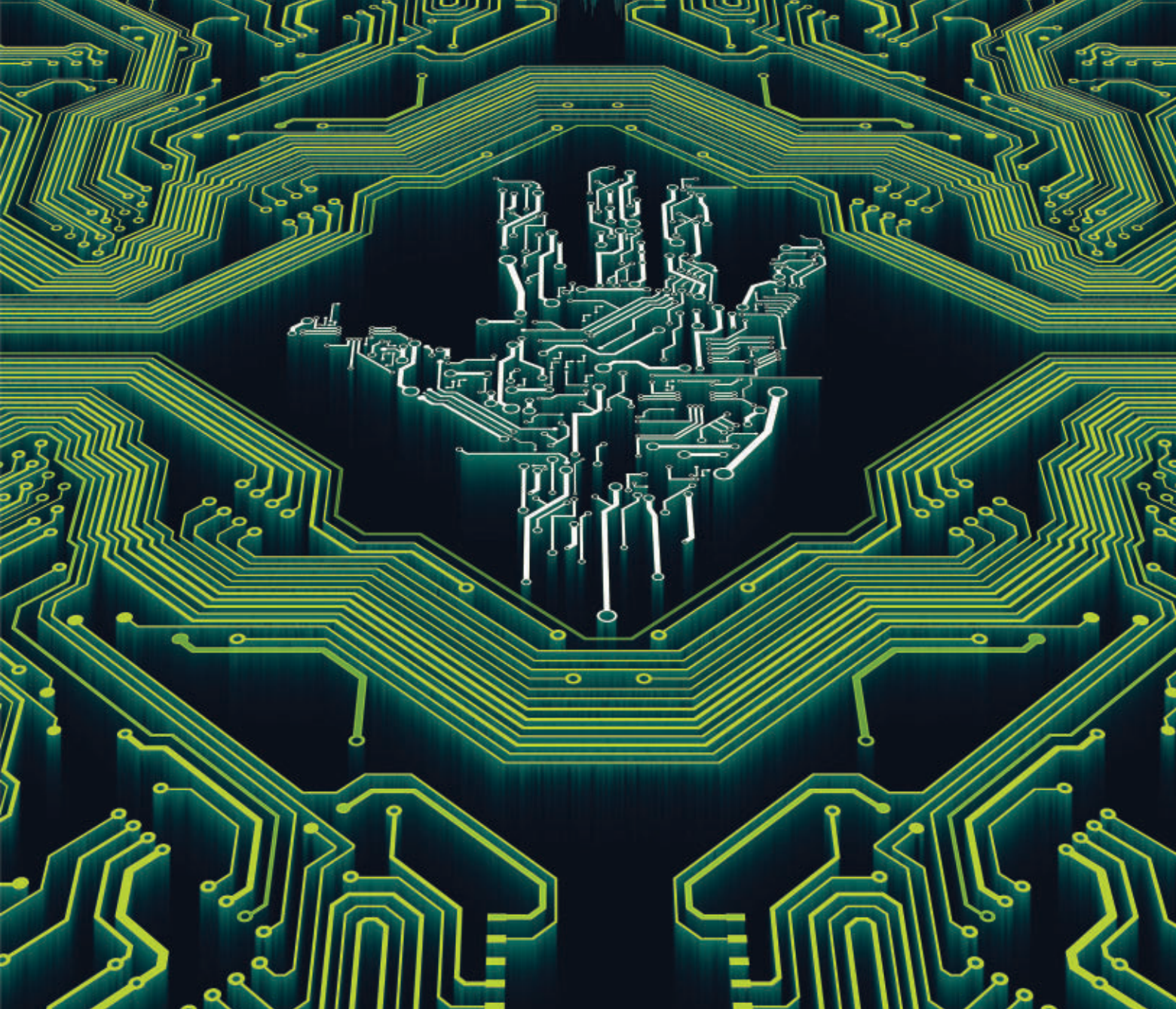
is static and can be openly queried. An adversary can “get ahead” of a legitimate authentication event by querying a genuine product ahead of time, and subsequently replaying the response or making a copy of the identifier.

Attack vectors associated with the inability to distinguish the genuine from a copy are numerous. Consider these two cases:

Physical item counterfeiting. Imagine an authentication system where an authentication server detects the presence of a counterfeit item based on scans associated with its public identifier; any available geolocation and timestamp information is associated with the public identifier upon a scan, and then stored on the authentication server. A counterfeiter can produce products with bar codes or RFID tags that are programmed with a previously seen identifier of a genuine product. If the server is presented with a scan of both a genuine and a counterfeit product, it cannot distinguish one from the other, and can do no better than marking both as suspected counterfeits.

False scan injection. It may be possible, depending on the system design, for an adversary to disrupt the authentication decision ability of the server without ever building a physical counterfeit product. Continuing from the previous example, let’s suppose that an adversary is able to electronically submit a scan to the authentication server, with a geolocation that has been spoofed; the scan is purely electronic and does not come from a physical product. The scan contains a public identifier obtained from a genuine product that was sitting in a store; alternatively, a list of product identifiers might have been pilfered from a distribution center. If a genuine product is scanned later, the server may regard the genuine product as a suspected counterfeit since that product identifier has apparently (from the perspective of the server) been in a different geolocation.

Indeed, the ability to distinguish a genuine from a copy is very useful,



if not central, to any effective anti-counterfeiting scheme. The magnetic-stripe-based credit cards that have been widely used in the U.S. suffer from not having this capability (they are being replaced with chip cards for this reason). Risk management analytics trigger the issuance of a new credit card number and a new credit card because the genuine one cannot be distinguished from a copy or clone. A consumer might be in possession of a genuine credit card, but they are forced to replace it because the system cannot distinguish it from a clone.

Broadly speaking, while the use of public identifiers can establish the supply-chain provenance trail, assuming all parties in the supply chain are honest, such a scheme does not pre-

vent *substitution attacks* where genuine products packaged with genuine public identifiers are replaced with counterfeit products packaged with maliciously replicated public identifiers. This can occur at supply-chain checkpoints or while products are in transit. It is desirable for an authentication system to be able to distinguish genuine from clone; address man-in-the-middle, replay, and substitution attacks; and allow multiple parties to cross-audit each other without the need to assume *all* the upstream supply-chain parties have behaved properly.

Dynamic Authentication

One of the main problems with using a public identifier to authenticate is that it is static and subject to replay attacks

when there is a man-in-the-middle adversary between the device and the authentication verification server. In the cryptographic realm, using a cryptographic primitive such as a keyed block cipher or a keyed hash function solves this problem. An authentication verification server generates a random number that can be used as a challenge, and the device's response is a function of the incoming challenge from the server and a secret key that is stored securely on the device. The authentication verification server also needs the secret key in order to verify that the incoming response from the device is correct; the response can be the cipher text of the block cipher or the digest of the keyed hash function. A public identifier (for example, a serial

number) can be used to enable the authentication verification server to look up the correct key for the device being queried. Here, the public identifier is being used for its proper purpose, to identify, and not as the primary means to authenticate. The response cannot be simply replayed to the server by a man-in-the-middle adversary because the server uses a different unpredictable challenge each time.

Cryptographic implementations of a challenge/response protocol require two items on the device:

Keyed cryptographic module. The device needs a cryptographic primitive such as a block cipher or hash function that uses a secret key.

Obfuscated secret key. The device needs a secret key that is securely stored, using, for example, secure non-volatile memory that is obfuscated and not publicly readable. Nonvolatile memory technologies are known to be subject to reverse-engineering attacks, where the secret-key bits that are stored can be recovered.⁹ Layout obfuscation is viewed as important in making key recovery more difficult.

Today, many products do not have dynamic authentication because cryptographic approaches may be too expensive or unusable in a passive circuit setting where energy to power the cryptographic circuit is harvested from an external RF field source (for example, a dedicated RFID reader or an NFC, or near-field communication-enabled, smartphone). A lower-complexity and inexpensive implementation of a challenge/response protocol would allow authentication to become more pervasive, especially if it could be integrated with a modern mobile smartphone in a manner that is easy to use.

There is an alternative method of implementing a challenge/response protocol with integrated measurement capability. The approach requires neither a keyed cryptographic module nor an obfuscated secret key on the silicon device. The idea emerged at MIT more than 10 years ago.⁶ This article discusses what the research community has learned since then in terms of using silicon PUFs (physical unclonable functions) for challenge/response authentication, how PUFs have been deployed to combat counterfeiting, and what some of the open problems are.

Silicon Physical Unclonable Function

Silicon PUF circuits generate output response bits based on a silicon device's *manufacturing variation*. The variation is difficult to control or reproduce since it is within the tolerances of the semiconductor fabrication equipment.⁶ The devices are manufactured identically from the same mask, and there is no secret-key programming to make each device respond differently even to the same challenge. When the same challenge is applied to different devices, each device outputs a different response. When the same challenge is applied repeatedly to the same device, the PUF outputs a response that is unique to the manufacturing instance of the PUF circuit, though some of the response bits may flip from query to query. This is because the response is produced based on a physical (versus a purely algorithmic) evaluation, which is subject to physical evaluation noise that depends on temperature, voltage, and other environmental effects.

PUFs have two broad classes of applications:¹⁴

Authentication. In the authentication use case, the silicon device is deemed authentic if the response from an authentication query is close enough in Hamming distance to a reference response obtained during a provisioning process. This is similar to the false-positive and false-negative behavior found in human biometric systems, where noisy mismatching bits can be “forgiven” using a threshold-based comparison. To prevent replay attacks, challenges are not reused. Early research at MIT showed that identically manufactured circuitry could produce unique challenge/response pairs on different silicon instances of the same circuit, and it was argued that for any given device, the response is difficult to predict when subject to a random challenge.

Key generation. If, instead of a threshold-based authentication, the PUF is to serve as a secret-key generator, only a fixed number of response bits need to be generated from the PUF. These bits can serve as symmetric key bits and can be used in a secure processor.¹⁵ Since cryptographic keys are required to be bit exact, the basic PUF circuit needs to be enhanced with error-correction logic, which

increases implementation complexity. There are security considerations in the exposure of error-correcting bits that need to be addressed in any PUF error-correction scheme.⁶ These considerations have been addressed in the past ten years, leading to commercial products that use PUFs to provide key generation capability enabled through manufacturing variation. For example, a PUF was integrated into an ARM-based SoC (system-on-a-chip) from Xilinx¹⁹ for secure firmware load during the device boot process.

The focus here is on the authentication use case. The goal is to bring authentication to applications where conventional cryptographic approaches are too expensive and cumbersome, and enable pervasive dynamic challenge/response authentication of physical items.

Differential Measurements

To make silicon PUFs viable, one main obstacle to overcome is preventing changes in environmental conditions (for example, temperature, voltage) from overwhelming minuscule manufacturing-variation-induced measurements. One of the key insights in early PUF research⁶ was to use differential measurements, which was later proven to be highly effective in silicon. Before then, the characterization of manufacturing variation required expensive semiconductor test equipment and a lot of evaluation time, and it was not obvious how to do so very quickly in an *in-circuit* fashion (without expensive external equipment) and in a manner that is robust to environmental changes. To the extent that temperature, voltage, and other environmental effects impact the differential measurements equally, their effects cancel out, thereby allowing the minute manufacturing-variation-induced effects to be manifested in the PUF response measurements. This is a general principle that was successfully employed in many subsequent silicon PUF circuits. A survey of different PUF approaches as they relate to authentication was published in 2015.⁴

The first custom silicon PUF implementation from MIT was the arbiter PUF,⁷ shown in the dash-lined box in Figure 1. This is referred to as a “basic” arbiter PUF building block to distinguish it from more complex constructs to be

discussed later. The PUF output for each basic arbiter PUF is derived from a differential race condition formed by successive delay stages. Each stage consists of a crossbar switch that can be formed using 2:1 multiplexers. A challenge bit for each of the $n = 64$ stages determines whether the parallel path or the cross path is active. Collectively, n challenge bits determine which path is chosen to create the transition at the top input to the arbiter latch, and similarly for the bottom input. The arbiter latch is formed using a pair of NAND gates that is cross-coupled (this is denoted by the rectangular boxes marked A in Figure 1). The difference comparison between the two delay paths, configured by the challenge bits, determines whether the basic arbiter PUF produces a 1 or a 0 output bit. The layout of the design has to be symmetrically matched so that random manufacturing variation affects the response.

To obtain a multibit challenge, a seed challenge is applied to a challenge expansion circuit—for example, an LFSR (linear-feedback shift register), which is not shown in the figure. The LFSR is used to produce the subchallenge bits $\langle c \rangle$. The subchallenge bits are used to generate a response bit. Using multiple subchallenges and concatenating the resulting response bits together forms a multibit response.

Because of the linear and additive nature of the response evaluation, it was recognized early on that learning attacks can be employed to mathematically model a basic arbiter PUF.⁷ Various techniques, including creating multiple instances of the basic PUF and bitwise-XORing their output bits,¹⁴ served as countermeasures to make learning attacks more difficult. Figure 1 depicts four basic 64-stage PUF instances whose output can be XORed together in both a parallel and a serial fashion. An XOR PUF is formed by instantiating multiple copies and XORing the output bits.

Model Building to Aid Authentication

For many commercial applications, more than just a few challenge/response pairs are needed. With the original PUF authentication scheme, the number of challenge/response pairs grows linearly with the number of supported authentication events. If the

number of supported events is relatively large—say, 1,000 authentications or more—this would result in many challenge/response pairs needing to be collected as part of the provisioning process and then stored on the server; both the provisioning time and the server storage would grow linearly with the number of authentications supported.

A major development in PUF research was using the ease of model building of the basic arbiter PUF (prior to the XOR countermeasure) to create an *authentication verification model* on the server side. This authentication verification model essentially serves as a symmetric counterpart to the physical PUF circuit on the device. Therefore, instead of collecting a number of challenge/response pairs that are linear in the number of authentication events and requiring a linear amount of storage on the server side, there is now a constant-size storage per PUF device on the server side regardless of the number of authentication events.

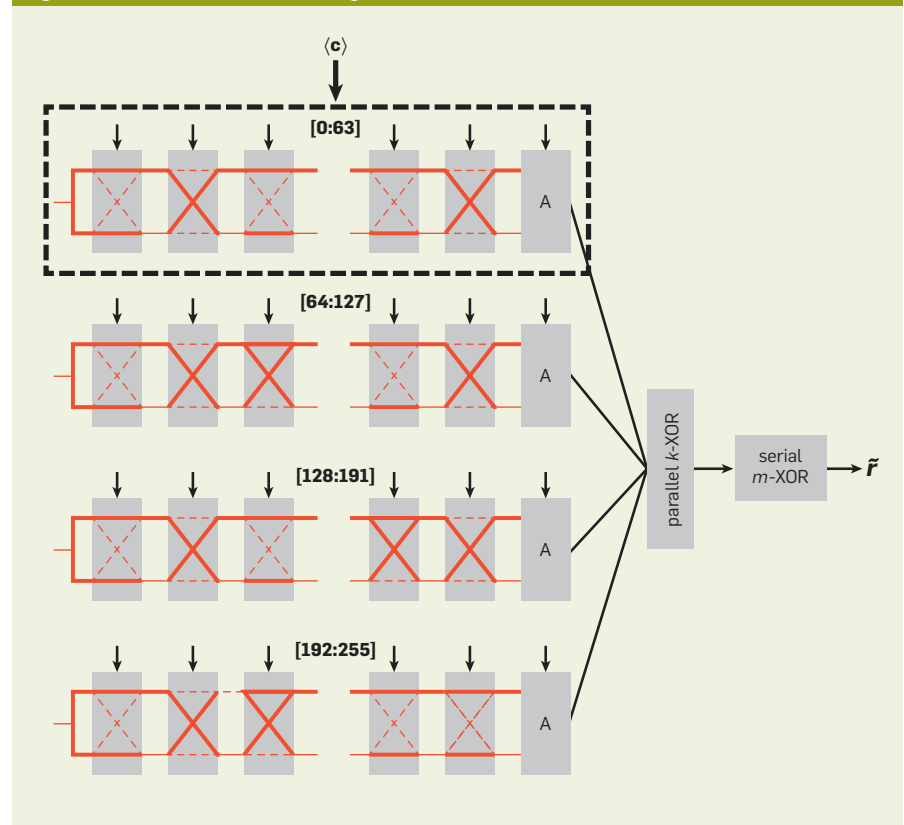
While the basic arbiter PUF can be learned with relative ease, the XORing produces output bits that are more difficult to learn. One can take advantage of this by making the easier-to-learn

variant available during the provisioning process (that is, bypassing the XORs). Here, the pre-XOR response bits for each basic arbiter PUF are obtained, and a state-of-the-art machine-learning algorithm can be used to derive the delay values on the server side. Afterward, the XORs are no longer bypassed, increasing the machine-learning difficulty for the adversary. To get the benefit of constant provisioning time and the constant storage requirement, the provisioning functionality needs to be disabled after the device leaves the manufacturing facility. For example, the publicly readable serial number of the device, once programmed, can disable the extraction of the PUF response bits prior to the XOR countermeasure. Reprogramming of the serial number is also disabled.

Machine Learning Attacks

When lightweight PUF-based authentication is performed using a threshold-based comparison as described previously, neither a cryptographic algorithm nor an obfuscated key is required on the silicon PUF device. Unfortunately, without a cryptographic algorithm and an obfuscated secret or

Figure 1. Basic arbiter PUF building block shown in the dotted line.



private key, it is difficult to derive an exponential number of challenge/response pairs from a linearly sized PUF circuit. Arbitrary logical or arithmetic post-processing cannot be applied to the silicon manufacturing variation since the physical PUF evaluation noise would be amplified. Therefore, popular PUF authentication circuits are evaluated in a mostly linear fashion, with limited nonlinear mixing (for example, using XORs as described earlier), and are therefore prone to modeling attacks.

Machine-learning attacks have been applied successfully on a variety of PUF constructs using computer-simulated PUF models¹¹ and, later, for silicon PUF implementations.¹² These attacks include the popular XOR construction of the arbiter PUF and serve as a benchmark for a PUF's machine-learning attack attributes as well as a catalyst for the development of countermeasures.

Another breakthrough attack was presented in 2015, where PUF response "reliability" information, which can be viewed as noise side-channel information, is used essentially to bypass the nonlinear mixing effects of the XORs, making even a PUF with a very high (for example, 20-plus) number of XORs relatively easy to learn (for example, using a few hundred thousand response bits).¹

Probabilistic Authentication

To thwart the attacks described in the previous section, machine-learning attack countermeasures were developed, taking advantage of the model-building concept. With it, challenges can now be determined *at runtime*. Recall that the authentication verification model that is stored on the server side can be used to synthesize *any* challenge/response pairs; the server is no longer restricted to the challenge/response pairs collected during the provisioning process. The device can now produce part of the challenge at runtime, so neither the server nor the device alone can fully determine the exact subchallenges used. This also makes the authentication process probabilistic: even if a (malicious) server repeatedly issues the same challenge C_s to the device, any output response R would be a function of both C_s and C_d . Here, C_d represents the device-generated part

of the challenge, which is passed back to the server. Instead of $R = \text{PUF}_{id}(C_s)$, now it is $R = \text{PUF}_{id}(C_s || C_d)$. As a result, the PUF cannot be trivially distinguished from random by repeating the same challenge C_s , analogous to what happens when probabilistic encryption is used in the cryptographic realm.

The use of device-generated challenges^{8,21} can be viewed as a countermeasure to prevent repeated challenges, which addresses the reliability-based machine-learning attacks that take advantage of noise-side-channel information.¹ This also addresses the noise-filtering approach using majority voting employed to attack silicon PUFs.¹²

Theoretical Learning Difficulty

Recently published research⁵ establishes the theoretical difficulty of PUF learning—in particular, for the popular XOR PUF construct. These recent results used the celebrated PAC (probably-approximately correct) framework,¹⁶ which links learning with complexity theory, and applied that framework to determine which kinds of PUFs are polynomially learnable and which would require an exponential attack resource (for example, attack runtime, number of response bits) with respect to the circuit size. The authors of this research⁵ not only declared certain XOR PUF constructs to be exponentially difficult to learn under the PAC framework, but also questioned whether such PUFs can be realized in practice. A new protocol-level countermeasure²⁰ allows exponentially difficult-to-learn PUFs based on the PAC results⁵ to be instantiated in practice, with silicon results to demonstrate practical feasibility. The main idea is to run the authentication protocol in *both* directions; only after the PUF device has authenticated the authentication verification server does the PUF device release new response bits to a potential man-in-the-middle adversary. The device is effectively locked down, with the exposure of *new* response bits implicitly controlled by the server at the protocol level.

To address noise-side-channel attacks, a device-side challenge^{8,21} can be added. The challenge/response behavior of the device is locked down at the protocol level, and the adversary is faced with machine learning using

limited data; meanwhile, the theoretical learning results from the previously mentioned research⁵ imply that both an exponential number of response bits and an exponential attack time are required. The response-data exposure can be throttled dynamically by the server as knowledge of new attacks emerges, even against an adaptive chosen-challenge adversary with uninterrupted interface access to the device, a result of the use of mutual authentication.

So Where Are We?

While a lot of strides have been made in terms of silicon PUF constructs and an understanding of their behavior in terms of practical and theoretical learnability when used in a challenge/response context, some questions remain. While the recent theoretical learning results declared that certain PUF constructs are exponentially difficult to PAC-learn, there is still a reliance on heuristic results for what the exponential learning-difficulty curve looks like for the "best attacks" available. Although there have been several years of machine-learning attacks on PUFs by multiple research groups thus far, there is still work to be done in this area to affirm a practical and safe heuristic limit in terms of number of response bits that can be exposed for different XOR PUF constructs that ensures security. Fortunately, the lockdown approach²⁰ allows the server to manage the response-bits exposure at the protocol level, thereby allowing a degree of dynamic adjustment to emerging attack results.

Additionally, side-channel attacks coupled with machine learning represent a relatively new research area. While the lockdown protocol²⁰ addressed various side-channel attacks that have been published, including noise-side-channel attacks, noise-filtering attacks, and backside photonics imaging attacks, new attacks may emerge that could bring forth research into new countermeasures and new PUF constructs.

PUF NFC IC and Tags

For a silicon PUF to be useful for authentication, it must be integrated into a form that can be easily authenticated. With the recent emergence of NFC-enabled smartphones, custom RFID/

NFC readers are no longer necessary for many use cases. An Android smartphone with a downloaded app would put the power of authentication in the hands of the consumer, or allow a store owner, distributor, or others in the supply chain to perform authentication of products. A product-authentication kiosk (similar to the barcode-scanning kiosks found in many major U.S. retailers) can also be used to provide PUF NFC authentication results as well as pulling information from a cloud server about a particular product instance's origin, source of manufacturing, freshness/expiration, provenance, and other information.

Major progress has been made in terms of PUF packaging and form factor. The first silicon PUF circuit was a relatively large research lab prototype and required wired connections to a computer for authentication, as shown in Figure 2.

After a decade of iteration and refinement, PUF NFC tags appeared in commercial products. Figure 3 shows a PUF NFC tag on a Canon camera package sold in Asia and an Android off-the-shelf NFC device that can be used to authenticate the tag.

Figure 4 is a close-up of a PUF-NFC IC encapsulated in a tag inlay. The area is taken up mostly by the antenna, and the actual IC area is extremely small (shown by the arrow). The antenna size affects the read range. The tag shown has a read range of about five centimeters. A small read range is useful for applications where privacy is an issue or for item-level tagging applications where it is desirable to know that a particular item is being interrogated using an NFC scan, which can be done with a modern NFC-enabled smartphone.

The lightweight nature of the PUF-NFC implementation also brings dynamic authentication capabilities to new product types—for example, secure paper, as shown in Figure 5. This is a useful means of tracking the processing of official documents (for example, when submitted by a private citizen to a government office for processing) and of authenticating them. An NFC scan made by a government employee authenticates the document; the authentication verification server can also record the geolocation and timestamp associated with the authen-

ticated document. This allows a citizen or a government audit agency to more easily track which processing step the document is undergoing as well as the whereabouts of the document.

When applied to an ID card, the PUF NFC approach not only allows a public employee to authenticate the identity of a private citizen, but also allows a private citizen to make sure a person who claims to be a public employee is not a fraud (for example, a public employee visiting a private citizen's house to perform inspection and possible repairs). An NFC scan with a smartphone would authenticate the employee's ID card, and an image of the card could be accessed on the homeowner's smartphone to make sure the picture and other vital information on the card have not been altered. A work order associated with the task that the visiting public employee is authorized to perform can also be displayed. Pervasive authentication by both the public employee and the private citizen would promote better public-sector accountability.

Previously, RFID scans required dedicated readers, which may be feasible to distribute to public employees or in other enterprise settings, but would be

Figure 2. MIT silicon PUF prototype, 2002.



Figure 3. Commercial deployment, 2014.



cumbersome if not cost-prohibitive to distribute to private citizens. A modern NFC-enabled smartphone, when used with a PUF NFC tag, democratizes authentication, putting the power of authentication in the hands of a private

Figure 4. PUF NFC tag.

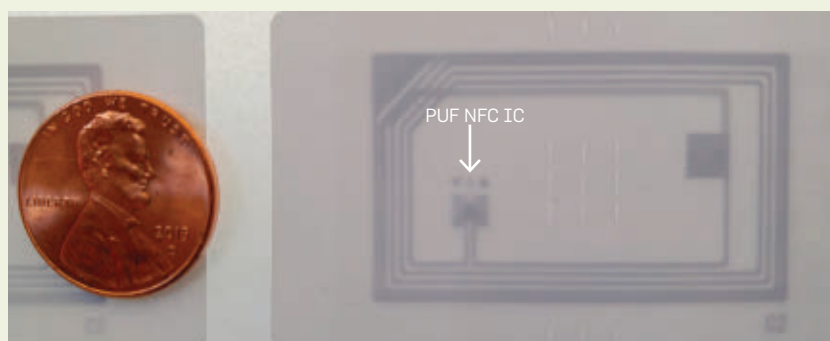
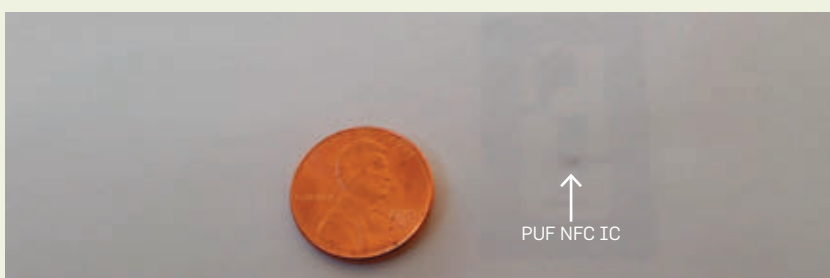


Figure 5. PUF embedded in secure paper.



individual without the burden of specialized reader hardware distribution.


Identification, Authentication, Authorization

In 2004, Bruce Schneier wrote about the importance of distinguishing three inter-related security services: identification, authentication, and authorization.¹³ While we have discussed PUFs mostly in the context of item-level authentication, they can also be used to provide each of the three security services.


Identification. As described by Schneier, an identifier needs to distinguish one member of a population from another member. Schneier also stated that conventional human biometric measurements such as fingerprint scans or iris scans cannot be used for identification; a separate identifier is needed so the biometric reading can be matched against a single reference biometric template vs. all the templates for the population. This is because, for a human biometric reading, if a match is performed across all templates in a population, the collision probability is too high (for example, on the order of 1 in 10,000 or 1 in 100,000^{10,18}). This means that out of a reasonably sized population larger than a small city, there is a high probability that two biometric readings would be regarded as coming from the same individual if a separate identifier were not used. Human biometrics can be used for authentication *if a person has already been identified through other means.*

When a silicon PUF is used, the collision probability can be made well below those for human biometrics—for example, it can be made below 1 in 1 trillion *without* the use of a separate public identifier. A silicon PUF implementation can scale the uniqueness information content better than a human biometric scheme, allowing the former to be used for identification.

Although the NFC PUF IC implementation described earlier uses a serial number to identify the device, if a challenge/response PUF authentication occurs, it is feasible to use the PUF to *identify* the device. A preselected challenge, possibly hardwired into the chip, can have the corresponding preselected response designated as an identifier. When different devices are queried, each device outputs a unique preselected response that is possibly noisy. The



A modern NFC-enabled smartphone, when used with a PUF NFC tag, democratizes authentication, putting the power of authentication in the hands of a private individual without the burden of specialized reader hardware distribution.



server performs a fuzzy match across a reference set of preselected responses (collected during provisioning) for the entire population to determine which device is being accessed. Then the server can fetch the corresponding authentication verification model for that device, to be used for the authentication phase. The nonvolatile storage bits on the chip that would otherwise be used to store the serial number to identify the device can be eliminated.

In certain use cases, where the RFID/NFC is used for identification only, and no on-chip data storage is needed to store ancillary data associated with a tagged product, it may be possible to eliminate all nonvolatile storage from an RFID/NFC device by using a PUF to provide the identification. Eliminating on-chip nonvolatile storage is a potential source for savings in terms of silicon area and manufacturing cost.

Authentication. Item-level authentication is an obvious use case for the challenge/response silicon PUF. This is a case of *server-to-device entity authentication*. As mentioned previously, it is also possible to run the entity authentication protocol in the reverse direction, for *device-to-server entity authentication*. In the lockdown protocol scenario,²⁰ running the entity authentication in both directions is used to limit response-bits exposure.

It is possible to extend the aforementioned mutual *entity* authentication functionality to perform *data* authentication and, in particular, to authenticate a relatively small number of data bytes. This protocol extension provides *server-to-device data authentication* as well as *device-to-server data authentication*. For example, a read/write interface can be implemented between the server and the device, so only authenticated read/write commands from the server are acted upon by the device. If the server read/write commands are modified in transit, the device could detect the bit modifications.

This can be achieved by incorporating the data bytes as a part of the challenge. The server first receives the serial number (id) from the device as well as a device-side challenge C_d . Then it sends to the device a server-side challenge C_s , command bytes B , and a response R , where $R = \text{PUF}_{id}(C_s \parallel C_d \parallel B)$ emulated using the server-side model. The

device validates the response; since the challenge now incorporates the command bytes B , the read/write command is also authenticated. The device authenticates the server and the incoming command before executing the latter. These might be commands that allow certain configuration bits to be written into the device's on-chip memory, or certain data to be read. The device can send back data in a similar manner. The server can then authenticate both the source of the data (entity authentication) and that the data from the device hasn't been modified in transit (data authentication).

Authorization. In many applications, the verifier is a public employee who obtains access to a private person's database entry from a cloud server in order to perform a more comprehensive authentication of an individual. The sensitive information may include security questions or other personal information that can be verbally validated. A private person's PUF-NFC ID card can be used to limit database access by a public employee so that such an employee cannot arbitrarily pilfer sensitive private data. The employee is *authorized* to obtain database access to certain sensitive and personal information only when a particular PUF-NFC ID card from a private person is physically present and produces a proper response to a server's challenge.

In many of today's RFID use cases, it is also common to store certain information associated with a tagged product on the RFID device itself, which incurs a silicon area overhead and an increase in manufacturing cost associated with larger on-chip nonvolatile storage. Since a conventional RFID device does not offer dynamic authentication (it emits only a static public identifier), the locally stored information cannot be safely moved into the cloud; this is because another RFID that is programmed with the same serial number would be associated with that data record in the cloud. For example, the data record can be the maintenance trail of an airplane part or the supply-chain provenance trail of a pharmaceutical product. If the RFID/NFC device, however, is used to offer authorization (for reading or both read/write) to access a particular database entry in the cloud, then the data that otherwise would be

stored locally on the RFID device can be more safely moved into the cloud. This minimizes the need for large storage local to the RFID/NFC device. An individual on the ground is authorized to access the cloud data record only when the PUF-NFC device is physically present. This assumes that reader devices are cloud-connected, which is increasingly becoming the trend.

Also, in the age of big data and cloud computing, data is worth more when it is aggregated in the cloud vs. stored separately in each tag. In the former case, analytics can be performed to uncover unauthorized activities or to gather other forms of business intelligence information. The PUF serves to bind the tag to a particular database record in the cloud by providing access authorization in a manner that a static public identifier cannot.

Conclusion

For more than a decade, silicon PUFs have gathered an enormous amount of interest in applications ranging from product authentication to secure processors. There have been commercial deployments and complete integration with authentication servers and consumer-grade, off-the-shelf smartphones to give the power of authentication to the ordinary person.

People know much more about PUFs and how to use them, including vulnerabilities and countermeasures, than they did a few years ago. As the PUF field becomes well established, more attacks and countermeasures are expected to be published to further vet the security properties of PUFs. Such a cycle has also been seen in the cryptographic world—for example, the AES-ECB algorithm was subject to the “Penguin” attack,¹⁷ and the plain RSA algorithm is subject to existential forgery,² both of which can be addressed by using the fundamental primitives in a different fashion. **C**

Related articles on queue.acm.org

A Threat Analysis of RFID Passports

Alan Ramos et al.

<http://queue.acm.org/detail.cfm?id=1626175>

The NSA and Snowden: Securing the All-Seeing Eye

Bob Toxen

<http://queue.acm.org/detail.cfm?id=2612261>

References

1. Becker, G. The gap between promise and reality: On the insecurity of XOR arbiter PUFs. *International Workshop on Cryptographic Hardware and Embedded Systems* (2015), 535–555.
2. Boneh, D., Joux, A. and Nguyen, P. Why textbook elgamal and RSA encryption are insecure. *Advances in Cryptology* (2000), 30–43.
3. Counterfeiting and piracy: stamping it out. *The Economist*. April 23, 2016.
4. Delvaux, J., Peeters, R., Gu, D. and Verbauwhe, I. A survey on entity authentication with strong PUFs. *ACM Computing Surveys* 48, 2 (2015), 26:1–26:42.
5. Ganji, F., Tajik, S. and Seifert, J.-P. Why attackers win: on the learnability of XOR arbiter PUFs. *International Conference on Trust and Trustworthy Computing* (2015), 22–39.
6. Gassend, B., Clarke, D., van Dijk, M. and Devadas, S. Silicon physical random functions. *ACM Conference on Computer and Communication Security* (2002).
7. Lim, D. Extracting secret keys from integrated circuits. Master's thesis, MIT, 2004.
8. Majzoubi, M., Rostami, M., Koushanfar, F., Wallach, D. and Devadas, S. SlenderPUF: A lightweight, robust and secure strong PUF by substrating matching. *IEEE International Workshop on Trustworthy Embedded Devices* (2012).
9. Quadir, S. E., Chen, J., Forte, D., Asadizanjani, N., Shahbazmohamadi, S., Wang, L., Chandy, J. and Tehranipoor, M. A survey on chip-to-system reverse engineering. *ACM Journal on Emerging Technologies in Computing Systems* 13, 1 (2016).
10. Quinn, G. and Grother, P. IREX III: Supplement I: Failure Analysis. NIST Interagency Report 7853 (2012).
11. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S. and Schmidhuber, J. Modeling attacks on physical unclonable functions. *ACM Conference on Computer and Communication Security* (2010).
12. Rührmair, U., Sölter, J., Sehnke, F., Xu, X., Mahmoud, A., Stoyanova, V., Dror, G., Schmidhuber, J., Burleson, W. and Devadas, S. PUF modeling attacks on simulated and silicon data. *IEEE Transactions on Information Forensics and Security* 8, 11 (2013), 1876–1891.
13. Schneider, B. Sensible authentication. *ACM Queue* 1, 10 (2004): 74–78.
14. Suh, G.E. and Devadas, S. Physical unclonable functions for device authentication and secret key generation. *Design Automation Conference* (2007), 9–14.
15. Suh, G.E. AEGIS: A single-chip secure processor. Ph.D. thesis. Electrical Engineering and Computer Science Dept., MIT, 2005.
16. Valiant, L. A theory of the learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.
17. Valsorda, F. The ECB penguin, 2013; <https://blog.filippo.io/the-ecb-penguin/>.
18. Wilson, C., Hicklin, R., Bone, M., Korves, H., Grother, P., Utley, B., Micheals, R., Zoepfl, M., Otto, S. and Watson, C. Fingerprint vendor technology evaluation 2003: summary of results and analysis report. NIST Internal Report 7123 (2004).
19. Xilinx Inc. Xilinx addresses rigorous security demands at 5th Annual Working Group for Broad Range of Applications, 2016; <http://www.prnewswire.com/news-releases/xilinx-addresses-rigorous-security-demands-at-fifth-annual-working-group-for-broad-range-of-applications-300351291.html>.
20. Yu, M., Hiller, M., Delvaux, J., Sowell, R., Devadas, S. and Verbauwhe, I. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems* 2, 3 (2016): 146–159.
21. Yu, M., M'Raihi, D., Verbauwhe, I. and Devadas, S. A noise bifurcation architecture for linear additive physical functions. *IEEE International Symposium on Hardware Oriented Security and Trust* (2014), 124–129.

Meng-Day (Mandel) Yu is the chief scientist at Verayo Inc., a research affiliate for CSAAIL/MIT, and is pursuing a Ph.D. based on a research career with COSIC/KU Leuven. He was manager of R&D engineering at TSI and developed a secure digital baseband radio.

Srinivas Devadas is the Webster professor of electrical engineering and computer science at MIT, where he has been since 1988. He served as associate head of EECS from 2005 to 2011. He is a Fellow of the ACM and IEEE.

Copyright held by owner(s)/authors.
Publication rights licensed to ACM. \$15.00.

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

Understanding the proposed revisions to the C language.

BY ROBERT C. SEACORD

Uninitialized Reads

MOST DEVELOPERS UNDERSTAND that reading uninitialized variables in C is a defect, but some do it anyway—for example, to create entropy. What happens when you read uninitialized objects is unsettled in the current version of the C standard (C11).³ Various proposals have been made to resolve these issues in the planned C2X revision of the standard. Consequently, this is a good time to understand existing behaviors as well as proposed revisions to the standard to influence the evolution of the C language. Given the behavior of uninitialized reads is unsettled in C11, prudence dictates eliminating uninitialized reads from your code.

This article describes *object initialization*, *indeterminate values*, and *trap representations* and then examines sample programs that illustrate the effects of these concepts on program behavior.

Initialization

Understanding how and when an object is initialized is necessary to understand the behavior of reading an uninitialized object.

An object whose identifier is declared with no linkage (a file scope object has internal linkage by default) and without the storage-class specifier `static` has *automatic storage duration*. The initial value of the object is *indeterminate*. If an initialization is specified for the object, it is performed each time the declaration or compound literal is reached in the execution of the block; otherwise, the value becomes indeterminate each time the declaration is reached.

Subsection 6.7.9 paragraph 10 of the C11 Standard⁴ describes how objects having static or thread storage duration are initialized:

If an object that has automatic storage duration is not initialized explicitly, its value is indeterminate. If an object that has static or thread storage duration is not initialized explicitly, then:

- ▶ if it has pointer type, it is initialized to a null pointer;
- ▶ if it has arithmetic type, it is initialized to (positive or unsigned) zero;
- ▶ if it is an aggregate, every member is initialized (recursively) according to these rules, and any padding is initialized to zero bits;
- ▶ if it is a union, the first named member is initialized (recursively) according to these rules, and any padding is initialized to zero bits.

Many of the dynamic allocation functions do not initialize memory. For example, the `malloc` function allocates space for an object whose size is specified by its argument and whose value is indeterminate. For the `realloc` function, any bytes in the new object beyond the size of the old object have indeterminate values.

Indeterminate Values

In all cases, an uninitialized object has an indeterminate value. The C standard states that an *indeterminate value* can be either an unspecified value or

I SEE VARIABLES



UNINITIALIZED

a trap representation. An *unspecified value* is a valid value of the relevant type where the C standard imposes no requirements on which value is chosen in any instance. The phrase “in any instance” is unclear. The word *instance* is defined in English as “a case or occurrence of anything,” but it is unclear from the context what is occurring. The obvious interpretation is that the occurrence is a read.⁹ A *trap representation* is an object representation that need not represent a value of the object type. Note that an unspecified value cannot be a trap representation.

If a stored value of an object has a trap representation and is read by an lvalue expression that does not have character type, the behavior is undefined. Consequently, an automatic variable can be assigned a trap representation without causing undefined behavior, but the value of the variable cannot be read until a proper value is stored in it.

Annex J.2, “Undefined behavior,” summarizes incompletely that behavior is undefined in the following circumstances:

- ▶ A trap representation is read by an lvalue expression that does not have character type.
- ▶ The value of an object with automatic storage duration is used while it is indeterminate.

The second undefined behavior is much more general (at least with respect to objects with automatic storage duration), because indeterminate values include all unspecified values and trap representations. This (incorrectly) implies that reading an indeterminate value from an object that has allocated, static, or thread storage duration is well-defined behavior unless a trap representation is read by an lvalue expression that does not have character type.

According to the current WG14 Convener, David Keaton, reading an

indeterminate value of any storage duration is implicit undefined behavior in C, and the description in Annex J.2 (which is non-normative) is incomplete. This revised definition of the undefined behavior might be stated as “The value of an object is read while it is indeterminate.”

Unfortunately, there is no consensus in the committee or broader community concerning uninitialized reads. Memarian and Sewell conducted a survey among 323 C experts to discover what they believe about the properties that systems software relies on in practice, and what current implementations provide.⁵ The survey gathered the following responses to the question, Is reading an uninitialized variable or struct member (with a current mainstream compiler):

- ▶ undefined behavior? 139 (43%)
- ▶ going to make the result of any expression involving that value unpredictable? 42 (13%)

- ▶ going to give an arbitrary and unstable value (maybe with a different value if you read again)? 21 (6%)
- ▶ going to give an arbitrary but stable value (with the same value if you read again)? 112 (35%)

Trap Representations

Trap representations are not always well understood, even by expert C programmers and compiler writers.⁶ A *trap representation* is an object representation that need not represent a value of the object type. Fetching a trap representation *might* perform a trap but is not required to. Performing a trap in C interrupts execution of the program to the extent that no further operations are performed.

Trap representations were introduced into the C language to help in debugging. Uninitialized objects can be assigned a trap representation so that an uninitialized read would trap and consequently be detected by the programmer during development. Some compiler writers would prefer to eliminate trap representations altogether and simply make any uninitialized read undefined behavior—the theory being, why prevent compiler optimizations because of obviously broken code? The counterargument is, why optimize obviously broken code and not simply issue a fatal diagnostic?

Unsigned integer types. The C standard states that for unsigned integer types other than `unsigned char`, an object representation is divided into *value bits* and *padding bits* (where padding bits are optional). Unsigned integer types use a pure binary representation known as the *value representation*, but the values of any padding bits are unspecified. According to the C standard, some combinations of padding bits might generate trap representations—for example, if one padding bit is a *parity bit*.

A parity bit acts as a check on a set of binary values, calculated in such a way that the number of ones in the set plus the parity bit should always be even (or occasionally, should always be odd). Early computers sometimes required the use of parity RAM, and parity checking could not be disabled. Historically, faulty memory was relatively common, and noticeable parity errors were not uncommon. Since then, errors have become less visible as simple parity RAM has

fallen out of use. Errors are now invisible because they are not detected, or they are corrected invisibly with ECC (error-correcting code) RAM. ECC memory can detect and correct the most common kinds of internal data corruption. Modern RAM is believed, with much justification, to be reliable, and error-detecting RAM has largely fallen out of use for non-critical applications. Parity bits and ECC bits are seen by the memory-processing unit but are invisible to the programmer.

No arithmetic operation on known values can generate a trap representation other than as part of an exceptional condition such as an overflow, and this cannot occur with unsigned types. All other combinations of padding bits are alternative object representations of the value specified by the value bits. Reads of trap representations have undefined behavior. No known current architecture, however, implements trap representations for unsigned integers of any type stored in memory other than `_Bool`. Consequently, trap representations for most unsigned integer types are an obsolete feature of the C standard.

The `_Bool` type is a special case of an unsigned type that has an actual memory-representable trap representation on many architectures. Values of type `_Bool` typically occupy one byte. Values in that byte other than 0 or 1 are trap representations. Consequently, an implementation may assume that a byte read of a `_Bool` object produces a value of 0 or 1, and optimize based on that assumption. GCC (GNU Compiler Collection) is an example of an implementation that behaves in this manner.

Because converting any nonzero value to type `_Bool` results in the value 1, type punning is required to create an object of type `_Bool` that contains a determinate bit pattern that does not represent any value of type `_Bool` (and is consequently a trap representation in the current standard).

Undefined behaviors can occur from deductions from which optimizations may follow. Consider the following code, for example:

```
_Bool a, b, c, d, e;
switch (a | (b << 1) | (c << 2) |
        (d << 3) | (e << 4))
```

Value range propagation may deduce that the switch argument is in

the range 0 to 31 and use that deduction when producing a table jump, so that an arbitrary address is jumped to if one of the values is out of range and, consequently, the switch argument is out of that range. No existing implementations have been shown to omit the range test for the table jump completely. GCC will optimize out the default case and jump to one of the other cases for an out-of-range argument. Omitting the range test, however, is permitted by the C standard and possibly by an implementation that defines `__STDC__ ANALYZABLE__`.

Consider the following code:

```
unsigned char f(
    unsigned char y
) {
    _Bool a; /* uninitialized */
    unsigned char x[2] = { 0, 0 };
    x[a] = 1;
}
```

In this example, it is possible that the write to `x[a]` would result in an out-of-bounds store for an implementation that does not define `__STDC__ ANALYZABLE__`.

Signed integer types. For signed integer types, the bits of the object representation are divided into three groups: value bits, padding bits, and the sign bit. Padding bits are not necessary; `signed char` in particular cannot have padding bits. If the sign bit is zero, it does not affect the resulting value.

The C standard supports three representations for signed integer values: sign and magnitude, one's complement, and two's complement. An implementation is free to choose which representation to use, although two's complement is the most common. The C standard also states that for sign and magnitude and two's complement, the value with sign bit 1 and all value bits zero can be a trap representation or a normal value. For one's complement, a value with sign bit 1 and all value bits 1 can be a trap representation or a normal value. In the case of sign and magnitude and one's complement, if this representation is a normal value, it is called a negative zero. For two's complement variables, this is the minimum (most negative) value for the type.

Most two's complement imple-


mentations treat all representations as normal values. Likewise, most sign magnitude and one's complement implementations treat negative zero as normal values. The C Standards Committee was unable to identify any current implementations that treated these representations as trap values, so this is a potentially unused and obsolete feature of the C standard.

Pointer types. An integer may be converted to any pointer type. The result is implementation-defined, might not be correctly aligned, might not point to an entity of the referenced type, and might be a trap representation. The mapping functions for converting pointers to integers and integers to pointers are intended to be consistent with the addressing structure of the execution environment.


Floating point-types. IEC 60559² requires two kinds of NaNs (not a number): quiet and signaling. The C Standards Committee has adopted only quiet NaNs. It did not adopt signaling NaNs because it is believed that their utility is too limited for the work required to support them.⁷

The IEC 60559 floating-point standard specifies quiet and signaling NaNs, but these terms can be applied to some non-IEC 60559 implementations as well. For example, the VAX reserved operand and the Cray indefinite are signaling NaNs. In IEC 60559 standard arithmetic, operations that trigger a signaling NaN argument generally return a quiet NaN result, provided no trap is taken. Full support for signaling NaNs implies restartable traps, such as the optional traps specified in the IEC 60559 floating-point standard. The C standard supports the primary utility of quiet NaNs “to handle otherwise intractable situations, such as providing a default value for 0.0/0.0,” as stated in IEC 60559.

Other applications of NaNs may prove useful. Available parts of NaNs have been used to encode auxiliary information—for example, about the origin of the NaN. Signaling NaNs might be candidates for filling uninitialized storage, and their available parts could distinguish uninitialized floating objects. IEC 60559 signaling NaNs and trap handlers potentially provide hooks for maintaining diagnostic information or for implementing special arithmetic.



Understanding how and when an object is initialized is necessary to understand the behavior of reading an uninitialized object.



C support for signaling NaNs, or for auxiliary information that could be encoded in NaNs, is problematic, however. Trap handling varies widely among implementations. Implementation mechanisms may trigger, or fail to trigger, signaling NaNs in mysterious ways. The IEC 60559 floating-point standard recommends that NaNs propagate, but it does not require this, and not all implementations do this. Additionally, the floating-point standard fails to specify the contents of NaNs through format conversion. Making signaling NaNs predictable imposes optimization restrictions that exceed the anticipated benefits. For these reasons, the C standard neither defines the behavior of signaling NaNs, nor specifies the interpretation of NaN significands.

The x86 Extended Precision Format is an 80-bit format first implemented in the Intel 8087 math coprocessor and is supported by all processors based on the x86 design that incorporate a floating-point unit. Pseudo-infinity, pseudo-zero, pseudo-NaN, unnormal, and pseudo-denormal are all trap representations.

Itanium CPUs have a NaT (not a thing) flag for each integer register. The NaT flag is used to control speculative execution and may linger in registers that are not properly initialized before use. An 8-bit value may have as many as 257 different values: 0–255 and a NaT value. C99, however, explicitly forbids a NaT value for an unsigned `char`. The NaT flag is not a trap representation in C, because a trap representation is an object representation and an object is a region of data storage in the execution environment and not a register flag.⁸

Instead of classifying the Itanium NaT flag as a trap representation, the following language was added to C11 subsection 6.3.2.1 paragraph 2 to account for the possibility of a NaT flag:

If the lvalue designates an object of automatic storage duration that could have been declared with the register storage class (never had its address taken), and that object is uninitialized (not declared with an initializer and no assignment to it has been performed prior to use), the behavior is undefined.

This sentence was added to C11 to support the Itanium NaT flag to give compiler developers the latitude to treat applicable uninitialized reads as

undefined behavior on all implementations. This undefined behavior applies even to direct reads of objects of type `unsigned char`. The `unsigned char` type normally has a special status in the standard in that values stored in non-bit-field objects may be copied into an object of type `unsigned char [n]` (for example, by `memcpy`), where `n` is the size of an object of that type.

Sample Programs

The preceding review of trap representations makes it clear the `unsigned char` type is the most interesting case. Consider the following code:

```
unsigned char f(
    unsigned char y
) {
    unsigned char x[1]; /*unit */
    if (x[0] > 10)
        return y/x[0];
    else
        return 10;
}
```

The `unsigned char` array `x` has automatic storage duration and is consequently uninitialized. Because it is declared as an array, the address of `x` is taken, meaning that the read is defined behavior. While the compiler could avoid taking the address, it cannot change the semantics of the code from unspecified value to undefined behavior. Consequently, the compiler is not allowed to translate this code into instructions that might perform a trap. Objects of `unsigned char` type are guaranteed not to have trap values. The read in this example is defined because it is from an object of type `unsigned char` and known to be backed up by memory. It is unclear, however, which value is read and if this value is stable. From this perspective, it could be argued that this behavior is implicitly undefined. Minimally, the standard is unclear and possibly contradictory.

Defect Report #451¹¹ deals with the instability of uninitialized automatic variables. The proposed committee response to this defect report states that any operation performed on indeterminate values will have an indeterminate value as a result. Library functions will exhibit undefined behavior when used on indeterminate values. It is unclear, however, whether `y/x[0]` can

result in a trap. Based on the proposed committee response to Defect Report #451, for all types that do not have trap representations, an uninitialized value can appear to change its value, allowing a conforming implementation to print two different values.

Consider the following code:

```
void f(void) {
    unsigned char x[1]; /*uninit */
    x[0] ^= x[0];
    printf("%d\n", x[0]);
    printf("%d\n", x[0]);
    return;
}
```

In this example, the `unsigned char` array `x` is intentionally uninitialized but cannot contain a trap representation because it has a character type. Consequently, the value is both indeterminate and an unspecified value. The bitwise exclusive OR operation, which would produce a zero on an initialized value, will produce an indeterminate result, which may or may not be zero. An optimizing compiler has the license to remove this code because it has undefined behavior. The two `printf` calls exhibit undefined behavior and, consequently, might do anything, including printing two different values for `x[0]`.

Uninitialized memory has been used as a source of entropy to seed random number generators in OpenSSL, DragonFly BSD, OpenBSD, and elsewhere.¹⁰ If accessing an indeterminate value is undefined behavior, however, compilers may optimize out these expressions, resulting in predictable values.¹

Conclusion

The behavior associated with uninitialized reads is an unsettled issue that the C Standards Committee needs to address in the next revision of the standard (C2X). One simple solution would be to eliminate trap representations altogether and simply state that reads of indeterminate values are undefined behavior. This would greatly simplify the standard (which itself is of value) and provide compiler developers with all the latitude they want to optimize code. The diametrically opposed solution is to define fully concrete semantics for uninitialized reads in which such a read is guaranteed to give the actual contents of memory.

Most likely, some middle ground will be identified that allows compiler optimizations but doesn't eliminate all guarantees for the programmer. One possibility is the introduction of a *wobly value* that would allow uninitialized objects to change values without requiring this to be undefined behavior.

Trap representations are an oddity, because they were introduced to help diagnose uninitialized reads but are now viewed with suspicion by the safety and security communities, which are wary that the undefined behavior associated with reading a trap value is being imparted to reads of indeterminate values. □

Related articles on queue.acm.org

Passing a Language through the Eye of a Needle

Roberto Ierusalimschy et al.

<http://queue.acm.org/detail.cfm?id=1983083>

The Challenge of Cross-Language Interoperability

David Chisnall

<http://queue.acm.org/detail.cfm?id=2543971>

References

- Debian Security Advisory, DSA-1571-1 OpenSSL—Predictable random number generator, 2008; <http://www.debian.org/security/2008/dsa-1571>.
- IEC, Binary floating-point arithmetic for microprocessor systems (60559:1989).
- ISO/IEC, Programming languages—C, 3rd ed. (ISO/IEC 9899:2011). Geneva, Switzerland.
- Krebbbers, R., Wiedijk, F. N1793: Stability of indeterminate values in C11; <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1793.pdf>.
- Memarian, K. and Sewell, P. Clarifying the C memory object model, 2016 (revised version of WG14 N2012). University of Cambridge; <http://www.cl.cam.ac.uk/~pes20/cerberus/notes64-wg14.html#clarifying-the-c-memory-object-model-uninitialised-values>.
- Memarian, K., Sewell, P. What is C in practice? 2015 (updated 2016). (Cerberus survey v2): Analysis of responses (2014)—with comments; <https://www.cl.cam.ac.uk/~pes20/cerberus/notes50-survey-discussion.html>.
- Open Standards. Optional support for signaling NaNs, 2003; <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1011.htm>.
- Peterson, R. Defect report #338, C99 seems to exclude indeterminate value from being an uninitialized register. Open Standards, 2007; http://www.open-std.org/jtc1/sc22/wg14/www/docs/dr_338.htm.
- Seacord, R.C. Clarification of unspecified value. Open Standards, 2016; <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2042.pdf>.
- Wang, X. More randomness or less; <http://queue.org/blog/2012/06/25/more-randomness-or-less/>.
- Wiedijk, F. and Krebbbers, R. Defect report #451. Instability of uninitialized automatic variables. Open Standards, 2013; http://www.open-std.org/jtc1/sc22/wg14/www/docs/dr_451.htm.

Robert C. Seacord is a Principal Security Consultant with NCC Group, where he works with software developers and software development organizations to eliminate vulnerabilities resulting from coding errors before they are deployed.

Copyright held by owner/author.
Publication rights licensed to ACM. \$15.00



How do you step up from mere contributor to real change-maker?

BY KATE MATSUDAIRA

Does Anybody Listen to You?

I KNOW IF you read this column, you care about your career. You're smart, passionate, and driven to do better.

And the people you work with probably know that about you, too.

But does anybody care? When you raise your hand to share an idea, does anybody listen? How likely is it an idea of yours would get implemented?

I get email messages all the time from readers who just cannot seem to get the people in charge to listen to their ideas. I totally get their frustration. It's really confusing when that happens, especially when you are smart, passionate, and driven—and you know that other people know that about you, too.

So where is the disconnect?

How do you step up from being a star contributor to a serious innovator and change-maker in your organization? How can you get people—those who make decisions in your company—to listen to you?

Read on to find out.

An idea on its own is not worth much.

Just because you think you know a better way to do something, even if you are right, no one is required to care.

Making great things happen at work is about more than just being smart. Good ideas succeed or fail depending on your ability to communicate them correctly to the people who have the power to make them happen.

When you are navigating an organization, it pays to know whom to talk to and how to reach them. Here is a simple guide to sending your ideas up the chain and actually making them stick. It takes three elements: the right people, the right timing, and the right way.

1. The Right People

For most decisions, there will be a number of stakeholders who must buy in to your idea: your manager, any other department leads who would be involved, maybe even your executive team.

Before you do anything else, you should identify exactly who needs to sign off on your idea. Who are the key stakeholders? Why do they matter? Who is least likely to be on board? Who is most likely?

Think about how each of these people functions in the organization and what your relationship to them is. Your relationship with each of these people matters and will influence how receptive each is to your ideas. Do not underestimate your power to influence anyone in your organization—even if you are relatively low ranking in the company—through good relationships.

To build those relationships and influence, however, you have to start working on them *before* you ever need to pitch an idea to anyone.

If you don't yet have relationships with the biggest influencers and decision makers in your organization, that should be your first order of business. Ask people to coffee, find ways to add value to them, and get to know them.

Then when you are ready to pitch your idea, move in an order that makes sense; don't be random or just start where it's easiest.

It usually makes much more sense to approach people one on one at first, rather than holding a big meeting and making an announcement. This way you can get feedback on

your idea, answer specific questions, and slowly build momentum so that by the time everyone hears your idea, it is a foregone conclusion that it will be adopted.

2. The Right Timing

Have you ever had somebody come to you with a minor request when you are buried under important work with a tight deadline? How excited were you to jump to their assistance, when you were already busy with something else? Probably not very.

Now multiply that by 10, and that is how your manager or executive team will respond to an idea that is presented at a time that doesn't make any sense. A good idea presented at the wrong time is likely to fail. Focus is one of the keys to business success, and people are not excited about adopting ideas that look like distractions.

So if your team is focused on hiring, for example, and that is your top priority this quarter, any ideas for improvement in other areas will probably be disregarded.

Once you have presented an idea and had it disregarded by your leadership, it can be much more difficult to get anyone to listen to it in the future. They will be thinking, "Didn't I already say no to this? Why are we talking about this again?"

Make sure your idea is worth the effort right now. The change must be measurable and likely to result in a great improvement for the organization.

And it goes without saying that raising any new idea and investing in its success should happen only when all of your regular work is done, and done well. No one will be receptive to your ideas if you are not already shining in the areas you are expected to work on.

3. The Right Way

At certain companies, great ideas succeed when they are presented as narratives. At other companies, slides are king. Other leaders love data and numbers.

How you tell your story and pitch your idea matters.

While you may think, "A good idea is a good idea, right? Any smart person will be able to see that," this is not 100% true. A person who relies on data to make good decisions will not find


a hypothetical story about a customer convincing. Someone who wants to see slides might not connect with an idea they just hear in a meeting.

Speak to people in their language; that is the only way to be sure they hear you.

Try to imagine every question each person will ask you (these will be different for every person; the finance department cares about different issues than the software engineers), and make sure you have an answer ready before they ask the question.

If you are not sure what style your leadership likes, think back to some recent meetings you have attended. How have the leaders expressed ideas themselves? That will give you a good guide to what they prefer and how they think.

Think also about other ideas that have been adopted recently. How did the person who proposed the idea convince the people in charge and the influencers? If you don't know what the person did, ask. Odds are, people who get their ideas passed are influencers, too, and it is good to have a working relationship with them.

How do things get done where you work? Who doesn't listen to you at work? Whom do they listen to? Hopefully this article has given you some direction on getting your ideas out there. Now it is time to go make them happen. 

Related articles on queue.acm.org

The Paradox of Autonomy and Recognition

Kate Matsudaira

<http://queue.acm.org/detail.cfm?id=2893471>

A Conversation with Tim Marland

<http://queue.acm.org/detail.cfm?id=1066063>

Culture Surprises in Remote Software Development Teams

Judith S. Olson, Gary M. Olson

<http://queue.acm.org/detail.cfm?id=966804>

Kate Matsudaira (katemats.com) is the founder of her own company, Popforms. Previously she worked at Microsoft and Amazon as well as startups like Decide, Moz, and Delve Networks.

MobiCom 2017

The 23rd Annual International Conference
on Mobile Computing and Networking

Snowbird, Utah, USA
October 16–20

ACM MobiCom is the premier international conference dedicated to addressing challenges in the areas of mobile computing and wireless and mobile networking. The 23rd MobiCom conference will be held at the Snowbird resort, close to Salt Lake City, Utah, USA.

The MobiCom conference series serves as a highly selective forum addressing networks, systems, algorithms and applications that support mobile computing and mobile and wireless networking. The conference program will feature regular scientific papers, as well as experience, challenge and verification papers. In addition MobiCom'17 will feature keynote speeches, workshops, research demonstrations and poster sessions. The conference will also host the 5th annual competition for novel and innovative mobile apps. Finally, MobiCom'17 is pleased to host the second ACM SIGMOBILE Student Career Evening (MobiJob).

See conference website for updated information and other deadlines.

<https://www.sigmobile.org/mobicom/2017/>



Association for Computing Machinery
Advancing Computing as a Science & Profession



SIG on Mobility of Systems, Users,
Data, and Computing

Microsecond-scale I/O means tension between performance and productivity that will need new latency-mitigating ideas, including in hardware.

BY LUIZ BARROSO, MIKE MARTY, DAVID PATTERSON, AND PARTHASARATHY RANGANATHAN

Attack of the Killer Microseconds

THE COMPUTER SYSTEMS we use today make it easy for programmers to mitigate event latencies in the nanosecond and millisecond time scales (such as DRAM accesses at tens or hundreds of nanoseconds and disk I/Os at a few milliseconds) but significantly lack support for microsecond (μs)-scale events. This oversight is quickly becoming a serious problem for programming warehouse-scale computers, where efficient handling of microsecond-scale events is becoming paramount for a new breed of low-latency I/O devices ranging from datacenter networking to emerging memories (see the first sidebar “Is the Microsecond Getting Enough Respect?”).

Processor designers have developed multiple techniques to facilitate a deep memory hierarchy that works at the nanosecond scale by providing a simple synchronous programming interface to the memory system. A load operation will logically

block a thread’s execution, with the program appearing to resume after the load completes. A host of complex microarchitectural techniques make high performance possible while supporting this intuitive programming model. Techniques include prefetching, out-of-order execution, and branch prediction. Since nanosecond-scale devices are so fast, low-level interactions are performed primarily by hardware.

At the other end of the latency-mitigating spectrum, computer scientists have worked on a number of techniques—typically software based—to deal with the millisecond time scale. Operating system context switching is a notable example. For instance, when a `read()` system call to a disk is made, the operating system kicks off the low-level I/O operation but also performs a software context switch to a different thread to make use of the processor during the disk operation. The original thread resumes execution sometime after the I/O completes. The long overhead of making a disk access (milliseconds) easily outweighs the cost of two context switches (microseconds). Millisecond-scale devices are slow enough that the cost of these software-based mechanisms can be amortized (see Table 1).

These synchronous models for interacting with nanosecond- and millisecond-scale devices are easier than the alternative of asynchronous models. In an asynchronous programming model, the program sends a request to a device and continue processing other work

» key insights

- **A new breed of low-latency I/O devices, ranging from faster datacenter networking to emerging non-volatile memories and accelerators, motivates greater interest in microsecond-scale latencies.**
- **Existing system optimizations targeting nanosecond- and millisecond-scale events are inadequate for events in the microsecond range.**
- **New techniques are needed to enable simple programs to achieve high performance when microsecond-scale latencies are involved, including new microarchitecture support.**



until the request has finished. To detect when the request has finished, the program must either periodically poll the status of the request or use an interrupt mechanism. Our experience at Google leads us to strongly prefer a synchronous programming model. Synchronous code is a lot simpler, hence easier to write, tune, and debug (see the second sidebar “Synchronous/Asynchronous Programming Models”).

The benefits of synchronous programming models are further amplified at scale, when a typical end-to-end application can span multiple small closely interacting systems, often written across several languages—C, C++, Java, JavaScript, Go, and the like—where the languages all have their own differing and ever-changing idioms for asynchronous programming.^{2,11} Having simple and consistent APIs and idioms across

the codebase significantly improves software-development productivity. As one illustrative example of the overall benefits of a synchronous programming model in Google, a rewrite of the Colossus⁶ client library to use a synchronous I/O model with lightweight threads gained a significant improvement in performance while making the code more compact and easier to understand. More important, however, was that shifting the burden of managing asynchronous events away from the programmer to the operating system or the thread library makes the code significantly simpler. This transfer is very important in warehouse-scale environments where the codebase is touched by thousands of developers, with significant software releases multiple times per week.

Recent trends point to a new breed of low-latency I/O devices that will

work in neither the millisecond nor the nanosecond time scales. Consider datacenter networking. The transmission time for a full-size packet at 40Gbps is less than one microsecond (300ns). At the speed of light, the time to traverse the length of a typical datacenter (say, 200 meters to 300 meters) is approximately one microsecond. Fast datacenter networks are thus likely to have latencies of the order of microseconds. Likewise, raw flash device latency is on the order of tens of microseconds. The latencies of emerging new non-volatile memory technologies (such as the Intel-Micron Xpoint 3D memory⁹ and Moneta³) are expected to be at the lower end of the microsecond regime as well. In-memory systems (such as the Stanford RAMCloud¹⁴ project) have also estimated latencies on the order of microseconds. Fine-grain GPU offloads (and other accelerators) have similar microsecond-scale latencies.

Not only are microsecond-scale hardware devices becoming available, we also see a growing need to exploit lower latency storage and communication. One key reason is the demise of Dennard scaling and the slowing of Moore’s Law in 2003 that ended the rapid improvement in microprocessor performance; today’s processors are approximately 20 times slower than if they had continued to double in performance every 18 months.⁸ In response, to enhance online services, cloud companies have increased the number of computers they use in a customer query. For instance, single-user search query already turns into thousands of remote procedure calls (RPCs), a number that will increase in the future.

Techniques optimized for nanosecond or millisecond time scales do not scale well for this microsecond regime. Superscalar out-of-order execution, branch prediction, prefetching, simultaneous multithreading, and other techniques for nanosecond time scales do not scale well to the microsecond regime; system designers do not have enough instruction-level parallelism or hardware-managed thread contexts to hide the longer latencies. Likewise, software techniques to tolerate millisecond-scale latencies (such as software-directed context switching) scale poorly down to microseconds; the overheads in these techniques often equal or exceed the latency of the I/O device

Table 1. Events and their latencies showing emergence of a new breed of microsecond events.

nanosecond events	microsecond events	millisecond events
register file: 1ns–5ns	datacenter networking: O(1μs)	disk: O(10ms)
cache accesses: 4ns–30ns	new NVM memories: O(1μs)	low-end flash: O(1ms)
memory access: 100ns	high-end flash: O(10μs)	wide-area networking: O(10ms)
	GPU/accelerator: O(10μs)	

Figure 1. Cumulative software overheads, all in the range of microseconds can degrade performance a few orders of magnitude.

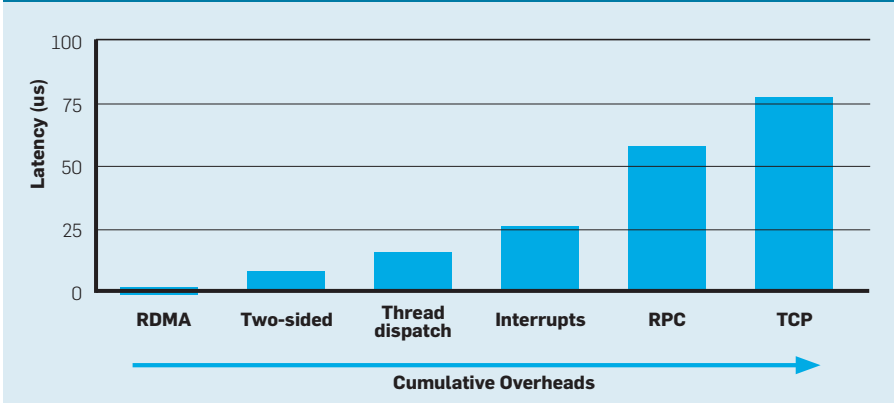


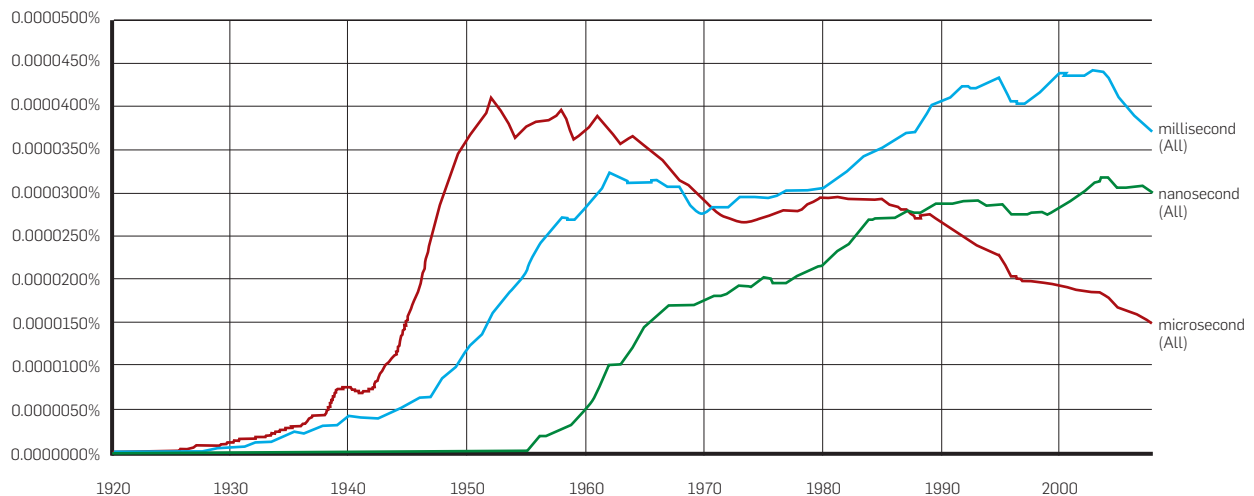
Table 2. Service times measuring number of instructions between I/O events for a production-quality-tuned web search workload. The flash and DRAM data are measured for real production use; the bolded data for new memory/storage tiers is based on detailed trace analysis.

Task: data-intensive workload	Service time (task length between I/Os)
Flash	225K instructions = O(100μs)
Fast flash	~20K instructions = O(10μs)
New NVM memory	~2K instructions = O(1μs)
DRAM	500 instructions = O(100ns–1μs)

Is the Microsecond Getting Enough Respect?

A simple comparison, as in the figure here, of the occurrence of the terms “millisecond,” “microsecond,” and “nanosecond” in Google’s n-gram viewer (a tool that charts frequencies of words in a large corpus of books printed from 1800 to 2012, <https://books.google.com/ngrams>) points to the lack of adequate attention to the microsecond-level time scale. Microprocessors moved out of the microsecond scale toward nanoseconds, while networking and storage latencies have remained in the milliseconds. With the rise of a new breed of I/O devices in the datacenter, it is time for system designers to refocus on how to achieve high performance and ease of programming at the microsecond-scale.

n-gram viewer of ms, μ s, and ns.



itself. As we will see, it is quite easy to take fast hardware and throw away its performance with software designed for millisecond-scale devices.

How to Waste a Fast Datacenter Network

To better understand how optimizations can target the microsecond regime, consider a high-performance network. Figure 1 is an illustrative example of how a $2\mu\text{s}$ fabric can, through a cumulative set of software overheads, turn into a nearly $100\mu\text{s}$ datacenter fabric. Each measurement reflects the median round-trip latency (from the application), with no queuing delays or unloaded latencies.

A very basic remote direct memory access (RDMA) operation in a fast datacenter network takes approximately $2\mu\text{s}$. An RDMA operation offloads the mechanisms of operation handling and transport reliability to a specialized hardware device. Making it a “two-sided” primitive (involving remote software rather than

just remote hardware) adds several more microseconds. Dispatching overhead from a network thread to an operation thread (on a different processor) further increases latency due to processor-wake-up and kernel-scheduler activity. Using interrupt-based notification rather than spin polling adds many more microseconds. Adding a feature-filled RPC stack incurs significant software overhead in excess of tens of microseconds. Finally, using a full-fledged TCP/IP stack rather than the RDMA-based transport adds to the final overhead that exceeds $75\mu\text{s}$ in this particular experiment.

In addition, there are other more unpredictable, and more non-intuitive, sources of overhead. For example, when an RPC reaches a server where the core is in a sleep state, additional latencies—often tens to hundreds of microseconds—might be incurred to come out of that sleep state (and potentially warm up processor caches). Likewise, various mechanisms (such as interprocessor interrupts, data copies, context switches,

and core hops) all add overheads, again in the microsecond range. We have also measured standard Google debugging features degrading latency by up to tens of microseconds. Finally, queuing overheads—in the host, application, and network fabric—can all incur additional latencies, often on the order of tens to hundreds of microseconds. Some of these sources of overhead have a more severe effect on tail latency than on median latency, which can be especially problematic in distributed computations.⁴

Similar observations can be made about overheads for new non-volatile storage. For example, the Moneta project³ at the University of California, San Diego, discusses how the latency of access for a non-volatile memory with baseline raw access latency of a few microseconds can increase by almost a factor of five due to different overheads across the kernel, interrupt handling, and data copying.

System designers need to rethink

the hardware and software stack in the context of the microsecond challenge. System design decisions like operating system-managed threading and interrupt-based notification that were in the noise with millisecond-scale designs now have to be redesigned more carefully, and system optimizations (such as storage I/O schedulers targeted explicitly at millisecond scales) have to be rethought for the microsecond scale.

How to Waste a Fast Datacenter Processor

The other significant negative effect of microsecond-scale events is on processor efficiency. If we measure processor resource efficiency in terms of throughput for a stream

of requests (requests per second), a simple queuing theory model tells us that as service time increases throughput decreases. In the microsecond regime, when service time is composed mostly of “overhead” rather than useful computation, throughput declines precipitously.

Illustrating the effect, Figure 2 shows efficiency (fraction of achieved vs. ideal throughput) on the y-axis, and service time on the x-axis. The different curves show the effect of changing “microsecond overhead” values; that is, amounts of time spent on each overhead event, with the line marked “No overhead” representing a hypothetical ideal system with zero overhead. The model assumes a simple closed queuing model with determin-

istic arrivals and service times, where service times represent the time between overhead events.

As expected, for short service times, overhead of just a single microsecond leads to dramatic reduction in overall throughput efficiency. How likely are such small service times in the real world? Table 2 lists the service times for a production web-search workload measuring the number of instructions between I/O events when the workload is tuned appropriately. As we move to systems that use fast flash or new non-volatile memories, service times in the range of 0.5 μ s to 10 μ s are to be expected. Microsecond overheads can significantly degrade performance in this regime.

At longer service times, sub-microsecond overheads are tolerable and throughput is close to the ideal. Higher overheads in the tens of microseconds, possibly from the software overheads detailed earlier, can lead to degraded performance, so system designers still need to optimize for killer microseconds.

Other overheads go beyond the basic mechanics of accessing microsecond-scale devices. A 2015 paper summarizing a multiyear longitudinal study at Google¹⁰ showed that 20%–25% of fleet-wide processor cycles are spent on low-level overheads we call the “datacenter tax.” Examples include serialization and deserialization of data, memory allocation and de-allocation, network stack costs, compression, and encryption. The datacenter tax adds to the killer microsecond challenge. A logical question is whether system designers can address reduced processor efficiency by offloading some of the overheads to a separate core or accelerator. Unfortunately, at single-digit microsecond I/O latencies, I/O operations tend to be closely coupled with the main work on the processor.

It is this frequent and closely coupled nature of these processor overheads that is even more significant, as in “death by 1,000 cuts.” For example, if microsecond-scale operations are made infrequently, then conservation of processor performance may not be a concern. Application threads could just busy-poll to wait for the microsecond operation to complete. Alternatively, if these operations are not coupled closely to the main computation on the processor, traditional offload

Figure 2. Efficiency degrades significantly at low service times due to microsecond-scale overheads.

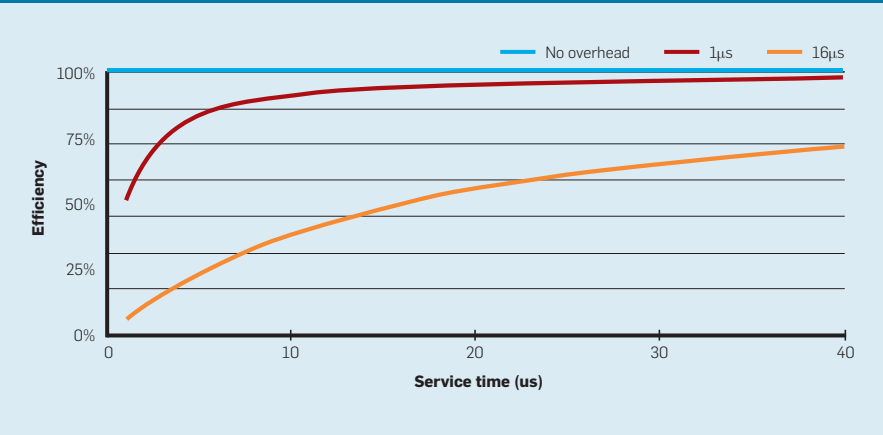


Table 3. High-performance computing and warehouse-scale computing systems compared. Though high-performance computing systems are often optimized for low-latency networking, their designs and techniques are not directly applicable to warehouse-scale computers.

	High-Performance Computing	Warehouse-Scale Computing
Workloads	Supercomputing workloads that often model the physical world; simpler, static data structures.	Large-scale online data-intensive workloads; operate on big data and complex dynamic data structures; response latency critical.
Programming environment	Code touched by a fewer programmers; slower-changing workloads. Hardware concurrency visible to programmers at compile time.	Codebase touched by thousands of developers; significant software releases 100 times per year. Automatic scale-out of queries per second.
System constraints	Focus on highest performance; recent emphasis on performance per Watt. Stranding of resources (such as underutilized processors) acceptable.	Focus on highest performance per dollar. Significant effort to avoid stranding of resources (such as processor, memory, and power).
Reliability/security	Reliability in hardware; often no-long-lived mutable data; no encryption.	Commodity hardware; reliability across the stack. Encryption/authentication requirements.

Synchronous/Asynchronous Programming Models

Synchronous APIs maintain the model of sequential execution, and a thread's state can be stored on the stack without explicit management from the application. This leads to code that is shorter, easier-to-understand, more maintainable, and potentially even more efficient. In contrast, with an asynchronous model, whenever an operation is triggered, the code must typically be split up and moved into different functions. The control flow becomes obfuscated, and it becomes the application's responsibility to manage the state between asynchronous event-handling functions that run to completion.

Consider a simple example where a function returns the number of words that appears in a given document identifier where the implementation may store the cached documents on a microsecond-scale device. The example represented in the first code portion makes two synchronous accesses to microsecond-scale devices: the first retrieves an index location for the document, and the second uses that location to retrieve the actual document content for counting words. With a synchronous programming model, it is not only straightforward but the model also can completely abstract away device accesses altogether by providing a natural, synchronous `CountWords` function with familiar syntax.

```
// Returns the number of words found in the document specified by 'doc_id'.
int CountWords(const string& doc_id) {
    Index index;
    bool status = ReadDocumentIndex(doc_id, &index);
    if (!status) return -1;
    string doc;
    status = ReadDocument(index.location, &doc);
    if (!status) return -1;
    return CountWordsInString(doc);
}
```

The C++11 example in the second code portion shows an asynchronous model where a “callback” (commonly known as a “continuation”) is used. When the asynchronous operation completes, the event-handling loop invokes the callback to continue the computation. Since two asynchronous operations are made, two callbacks are needed. Moreover the `CountWords` API now must be asynchronous itself. And unlike the synchronous example, state between asynchronous operations must be explicitly managed and tracked on the heap rather than on a thread's stack.

```
// Heap-allocated state tracked between asynchronous operations.
struct AsyncCountWordsState {
    bool status;
    std::function<void(int)> done_callback;
    Index index;
    string doc;
};

// Invokes the 'done' callback, passing the number of words found in the
// document specified by 'doc_id'.
void AsyncCountWords(const string& doc_id, std::function<void(int)> done) {
    // Kick off the first asynchronous operation, and invoke the
    // ReadDocumentIndexDone when it finishes. State between asynchronous
    // operations is tracked in a heap-allocated 'state' object.
    auto state = new AsyncCountWordsState();
    state->done_callback = done;
    AsyncReadDocumentIndex(doc_id, &state->status, &state->index,
        std::bind(&ReadDocumentIndexDone, state));
}

// First callback function.
void ReadDocumentIndexDone(AsyncCountWordsState* state) {
    if (!state->status) {
        state->done_callback(-1);
        delete state;
    } else {
        // Kick off the second asynchronous operation, and invoke the
        // ReadDocumentDone function when it finishes. The 'state' object
        // is passed to the second callback for final cleanup.
        AsyncReadDocument(state->index.location, &state->status,
            &state->doc, std::bind(&ReadDocumentDone, state));
    }
}

// Second callback function.
void ReadDocumentDone(AsyncCountWordsState* state) {
    if (!state->status) {
        state->done_callback(-1);
    } else {
        state->done_callback(CountWordsInString(state->doc));
    }
    delete state;
}
```

Callbacks make the flow of control explicit rather than just invoking a function and waiting for it to complete. While languages and libraries can make it somewhat easier to use callbacks and continuations (such as support for `async/await`, lambdas, tasks, and futures), the result remains code that is arguably messier and more difficult to understand than a simple synchronous function-calling model.

In this asynchronous example, wrapping the code inside a synchronous `CountWords()` function—in order to abstract away the presence of an underlying asynchronous microsecond-scale device—requires support for a `wait` primitive. The existing approaches for waiting on a condition invoke operating system support for thread scheduling, thereby incurring significant overhead when wait times are at the microsecond scale.

engines can be used. However, given fine-grain and closely coupled overheads, new hardware optimizations are needed at the processor micro-architectural level to rethink the implementation and scheduling of such core functions that comprise future microsecond-scale I/O events.

Solution Directions

This evidence indicates several major classes of opportunities ahead in the upcoming “era of the killer microsecond.” First, and more near-term, it is relatively easy to squander all the benefits from microsecond devices by progressively adding suboptimal sup-

porting software not tuned for such small latencies. Computer scientists thus need to design “microsecond-aware” systems stacks. They also need to build on related work from the past five years in this area (such as Caulfield et al.,³ Nanavati et al.,¹² and Ousterhout et al.¹⁴) to continue redesigning traditional low-level system optimizations—reduced lock contention and synchronization, lower-overhead interrupt handling, efficient resource utilization during spin-polling, improved job scheduling, and hardware offloading *but for the microsecond regime*.

The high-performance computing industry has long dealt with low-

latency networking. Nevertheless, the techniques used in supercomputers are not directly applicable to warehouse-scale computers (see Table 3). For one, high-performance systems have slower-changing workloads, and fewer programmers need to touch the code. Code simplicity and greatest programmer productivity are thus not as critical as they are in deployments like Amazon or Google where key software products are released multiple times per week. High-performance workloads also tend to have simpler and static data structures that lend themselves to simpler, faster networking. Second, the emphasis is primarily on performance

(vs. performance-per-total-cost-of-ownership in large-scale Web deployments). Consequently, they can keep processors highly underutilized when, say, blocking for MPI-style rendezvous messages. In contrast, a key emphasis in warehouse-scale computing systems is the need to optimize for low latencies while achieving greater utilizations.

As discussed, traditional processor optimizations to hide latency run out of instruction-level pipeline parallelism to tolerate microsecond latencies. System designers need new hardware optimizations to extend the use of synchronous blocking mechanisms and thread-level parallelism to the microsecond range.

Context switching can help, albeit at the cost of increased power and latency. Prior approaches for fast context switching (such as Denelcor HEP¹⁵ and Tera MTA computers¹) traded off single-threaded performance, giving up on latency advantages from locality and private high-level caches and, consequently, have limited appeal in a broader warehouse-scale computing environment where programmers want to tolerate microsecond events with low overhead and ease of programmability. Some languages and runtimes (such as Go and Erlang) feature lightweight threads^{5,7} to reduce memory and context-switch overheads associated with operating system threads. But these systems fall back to heavier-weight mechanisms when dealing with I/O. For example, the Grappa platform¹³ builds an efficient task scheduler and communication layer for small messages but trades off a more restricted programming environment and less-efficient performance and also optimizes for throughput. New hardware ideas are needed to enable context switching across a large number of threads (tens to hundreds per processor, though finding the sweet spot is an open question) at extremely fast latencies (tens of nanoseconds).

Hardware innovation is also needed to help orchestrate communication with pending I/O, efficient queue management and task scheduling/dispatch, and better processor state (such as cache) management across several contexts. Ideally, future schedulers will have rich support for I/O (such as being able to park a thread based on the readiness of multiple I/O operations). For instance,

facilities similar to the x86 `monitor/mwait` instructions^a could allow a thread to yield until an I/O operation completes with very low overhead. Meanwhile, the hardware could seamlessly schedule a different thread. To reduce overhead further, a potential hardware implementation could cache the thread's context in either the existing L1/L2/L3 hierarchy or a special-purpose context cache. Improved resource isolation and quality-of-service control in hardware will also help. Hardware support to build new instrumentation to track microsecond overheads will also be useful.

Finally, techniques to enable microsecond-scale devices should not necessarily seek to keep processor pipelines busy. One promising solution might instead be to enable a processor to stop consuming power while a microsecond-scale access is outstanding and shift that power to other cores not blocked on accesses.

Conclusion

System designers can no longer ignore efficient support for microsecond-scale I/O, as the most useful new warehouse-scale computing technologies start running at that time scale. Today's hardware and system software make an inadequate platform, particularly given support for synchronous programming models is deemed critical for software productivity. Novel microsecond-optimized system stacks are needed, reexamining questions around appropriate layering and abstraction, control and data plane separation, and hardware/software boundaries. Such optimized designs at the microsecond scale, and corresponding faster I/O, can in turn enable a virtuous cycle of new applications and programming models that leverage low-latency communication, dramatically increasing the effective computing capabilities of warehouse-scale computers.

Acknowledgments

We would like to thank Al Borchers, Robert Cypher, Lawrence Greenfield, Mark Hill, Urs Hölzle, Christos Kozyrakis, Noah Levine, Milo Martin, Jeff Mogul, John Ousterhout, Amin Vahdat, Sean Quinlan, and Tom Wenisch

a The x86 `monitor/mwait` instructions allow privileged software to wait on a single memory word.

for detailed comments that improved this article. We also thank the teams at Google that build, manage, and maintain the systems that contributed to the insights we have explored here. □

References

1. Alverson, R. et al. The Tera computer system. In *Proceedings of the Fourth International Conference on Supercomputing* (Amsterdam, The Netherlands, June 11–15). ACM Press, New York, 1990, 1–6.
2. Boost C++ Libraries. Boost asio library; http://www.boost.org/doc/libs/1_59_0/doc/html/boost_asio.html
3. Caulfield, A. et al. Moneta: A high-performance storage array architecture for next-generation, non-volatile memories. In *Proceedings of the 2010 IEEE/ACM International Symposium on Microarchitecture* (Atlanta, GA, Dec. 4–8). IEEE Computer Society Press, 2010.
4. Dean, J. and Barroso, L.A. The tail at scale. *Commun. ACM* 56, 2 (Feb. 2013), 74–80.
5. Erlang. *Erlang User's Guide Version 8.0. Processes*; http://erlang.org/doc/efficiency_guide/processes.html
6. Fikes, F. Storage architecture and challenges. In *Proceedings of the 2010 Google Faculty Summit* (Mountain View, CA, July 29, 2010); <http://www.systutorials.com/3306/storage-architecture-and-challenges/>
7. Golang.org. Effective Go. Goroutines; https://golang.org/doc/effective_go.html#goroutines
8. Hennessy, J. and Patterson, D. *Computer Architecture: A Quantitative Approach, Sixth Edition*. Elsevier, Cambridge, MA, 2017.
9. Intel Newsroom. Intel and Micron produce breakthrough memory technology, July 28, 2015; http://newsroom.intel.com/community/intel_newsroom/blog/2015/07/28/intel-and-micron-produce-breakthrough-memory-technology
10. Kanev, S. et al. Profiling a warehouse-scale computer. In *Proceedings of the 42nd International Symposium on Computer Architecture* (Portland, OR, June 13–17). ACM Press, New York, 2015.
11. Microsoft. *Asynchronous Programming with Async and Await (C# and Visual Basic)*; <https://msdn.microsoft.com/en-us/library/hh191443.aspx>
12. Nanavati, M. et al. Non-volatile storage: Implications of the datacenter's shifting center. *Commun. ACM* 50, 1 (Jan. 2016), 58–63.
13. Nelson, J. et al. Latency-tolerant software distributed shared memory. In *Proceedings of the USENIX Annual Technical Conference* (Santa Clara, CA, July 8–10). Usenix Association, Berkeley, CA, 2015.
14. Ousterhout, J. et al. The RAMCloud storage system. *ACM Transactions on Computer Systems* 33, 3 (Sept. 2015), 7:1–7:55.
15. Smith, B. A pipelined shared-resource MIMD computer. Chapter in *Advanced Computer Architecture*. D.P. Agrawal, Ed. IEEE Computer Society Press, Los Alamitos, CA, 1986, 39–41.
16. Wikipedia.org. Google n-gram viewer; https://en.wikipedia.org/wiki/Google_Ngram_Viewer

Luiz André Barroso (luiz@google.com) is a Google Fellow and Vice President of Engineering at Google Inc., Mountain View, CA.

Michael R. Marty (mikemarty@google.com) is a senior staff software engineer and manager at Google Inc., Madison, WI.

David Patterson (davidpatterson@gmail.com) is an emeritus professor at the University of California, Berkeley, and a distinguished engineer at Google Inc., Mountain View, CA.

Parthasarathy Ranganathan (partha.ranganathan@google.com) is a principal engineer at Google Inc., Mountain View, CA.

Copyright held by the authors.



Watch the authors discuss their work in this exclusive *Communications* video. <http://cacm.acm.org/videos/the-attack-of-the-killer-microseconds>

This framework for developing pre-service teachers' knowledge does not necessarily depend on computers or other educational technology.

BY AMAN YADAV, CHRIS STEPHENSON, AND HAI HONG

Computational Thinking for Teacher Education

ENTHUSIASM HAS GROWN in recent years for computer science education in many countries, including Australia, the U.S, and the U.K.^{14,15} For example, in 2012, the Royal Society in the U.K. said, “Every child should have the opportunity to learn concepts and principles from computing, including computer

science and information technology, from the beginning of primary education onward, and by age 14 should be able to choose to study toward a recognized qualification in these areas.”²⁶ And in 2016, the College Board in the U.S. launched a new computer science curriculum for high schools called “Computer Science Principles”⁶ focusing on exposing students to computational thinking and practices to help them understand how computing influences the world. Within the computer science education community, computational thinking is a familiar term, but among K–12 teachers, administrators, and teacher educators there is confusion about what it entails.

Computational thinking is often mistakenly equated with using computer technology.^{11,29} In order to address this misrepresentation, the scope of this article includes a definition of computa-

» key insights

- **Few teacher-education programs focus on training pre-service teachers to incorporate computational thinking into K–12 classrooms.**
- **Redesign of courses on educational technology and methods is critical to developing pre-service teacher competencies in computational thinking.**
- **Education and computer science faculty should work collaboratively, using their complementary expertise in computing and teacher development.**

tional thinking and the core constructs that would make it relevant for key stakeholders from K–12 education and teacher-training programs.

Denning suggested¹³ that the idea of computational thinking has been present since the 1950s and 1960s “as algorithmic thinking,” referring specifically to using an ordered and precise sequence of steps to solve problems and (when appropriate) a computer to automate that process. Today, the term “computational thinking” is defined by Wing²⁸ as “solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science.” Computational thinking also involves “using abstraction and decomposition when attacking a large complex task or designing a large complex system.”²⁸ A report on computational thinking by the National Council for Research suggested it is a cognitive skill the average person is expected to possess. For example, the cognitive aspects of computational thinking involve the use of heuristics, a problem-solving approach that involves applying a general rule of thumb or strategy that may lead to a solution.²⁸ This heuristic process involves searching for strategies that generally produce the right solution but do not always guarantee a solution to the problem. For example, “asking for directions in an unfamiliar place” from a local usually leads one to the right place, but one could also end up at a wrong place, depending on one’s understanding of local geography.¹⁷ Heuristic processes can be contrasted

with algorithms that guarantee success if followed correctly.¹⁷

Computational thinking has been suggested as an analytical thinking skill that draws on concepts from computer science but is a fundamental skill used by, and useful for, all people.²⁸ Some have argued that computational thinking is a practice that is central to all sciences, not just computer science.^{13,28} Bundy,⁴ for example, noted that computational thinking concepts have been used in other disciplines through problem-solving processes and the ability to think computationally is essential to every discipline. These powerful ideas and processes have begun to have significant influence in multiple fields, including biology, journalism, finance, and archaeology,²² making it important to include computational thinking as a priority for K–12 education. Wing²⁸ said, “To reading, writing, and arithmetic, we should add computational thinking to every child’s analytical ability.” In summary, computational thinking is a set of problem-solving thought processes derived from computer science but applicable in any domain.

Embedding computational thinking in K–12 teaching and learning requires teacher educators to prepare teachers to support students’ understanding of computational thinking concepts and their application to the disciplinary knowledge of each subject area. Specifically, teacher educators need to provide teachers with the content, pedagogy, and instructional strategies needed to incorporate computational

thinking into their curricula and practice in meaningful ways, enabling their students to use its core concepts and dispositions to solve discipline-specific and interdisciplinary problems. It is important to acknowledge that the current lack of an agreed-upon, exclusive definition of the elements of computational thinking makes it a challenge to develop clear pathways for pre-service teachers to be educated teachers—computationally.³⁰ Nevertheless, it is both important and possible to begin taking steps in this direction.

Here, we argue that, given the cross-disciplinary nature of computational thinking and the need to address educational reforms—Next Generation Science Standards and Common Core—it is beneficial to prepare teachers to incorporate computational thinking concepts and practices into K–12 classrooms. While most current efforts to embed computational thinking focus on in-service professional development, we posit that pre-service teacher education is an opportune time to provide future teachers with the knowledge and understanding they require to successfully integrate computational thinking into their curricula and practice. The following sections discuss the relevance of computational thinking constructs in K–12 education. We also discuss how to embed computational thinking into classrooms by using it as a methodology for teaching programming. In addition, we provide examples of how teachers in various disciplines can use computational thinking to address and enhance




Edtech start-up pavilion at International Society for Technology in Education conference.


existing learning outcomes. Finally, we discuss ways to implement computational thinking into pre-service teacher training, including how teacher educators and computer science educators can collaborate to develop pathways to help pre-service teachers become computationally educated.

What Is Computational Thinking?

How do we define computational thinking and use the definition as a framework to embed it in K–12 classrooms? Wing’s seminal column²⁸ offered a promising definition of computational thinking: “... breaking down a difficult problem into more familiar ones that we can solve (problem decomposition), using a set of rules to find solutions (algorithms), and using abstractions to generalize those solutions to similar problems.” Finally, automation is the ultimate step in computational thinking that can be implemented through computing tools. These concepts cut across disciplines and could be embedded across subjects in elementary and secondary schools. Based on this definition, a steering committee formed by the Computer Science Teachers Association (CSTA <https://www.csteachers.org/>) and the International Society for Technology in Education (ISTE <https://www.iste.org/>) presented a computational thinking framework for K–12 schools in 2011 with nine core computational thinking concepts and capabilities, including data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation. They were also discussed in 2015 in the Computing at School (CAS) framework and guide for teachers to enable teachers in the U.K. to incorporate computational thinking into their teaching work.¹⁰ CSTA/ISTE and CAS also provide pedagogical approaches to embed these capabilities across the curriculum in elementary and secondary classes. For example, CSTA/ISTE describes how the nine core computational thinking concepts and capabilities could be practiced in science classrooms by collecting and analyzing data from experiments (data collection and data analysis) and summarizing that data (data representation).



Computational thinking is often mistakenly equated with using computer technology.



Algorithms are central to both computer science and computational thinking. Algorithms underlie the most basic tasks everyone engages in, from following a simple cooking recipe to providing complicated driving directions. Because there is a general misconception that algorithms are used only to solve mathematical problems and are not applicable in other disciplines,²⁹ it is important to introduce students to algorithms by first using examples from their daily lives. For example, in early grades, teachers could highlight the steps involved in brushing teeth, while in later grades, students could engage in following steps during a lab experiment. Understanding algorithms as a set of precise steps provides the basis for understanding how to develop an algorithm that can be implemented in a computing program. Students can be exposed to the computational thinking concept of abstraction by creating models of physics entities (such as a model of the solar system).³ Abstraction helps students learn to strip away complexity in order to reduce an artifact to its essence and still be able to know what the artifact is. In another example, Barr and Stephenson³ suggested students can learn about parallelization by simultaneously running experiments with different parameters. A number of leading research, educational, and funding organizations have argued for the need to introduce K–12 students to these core constructs and practices.

Computational Thinking in K–12 Education

The National Research Council (NRC)²² highlighted the importance of exposing students to computational thinking notions early in their school years and helping them to understand when and how to apply these essential skills.^{3,22} The NRC report²² on the scope and nature of computational thinking highlighted the need for students to learn the related strategies from knowledgeable educators who model these strategies and guide their students to use them independently. Similarly, Barr and Stephenson³ argued that, given that students will go into a workforce heavily influenced by computing, it is important for them to begin to work with computational thinking ideas and

tools in grades K–12. Specifically, they discussed³ the need to highlight “algorithmic problem solving practices and applications of computing across disciplines, and help integrate the application of computational methods and tools across diverse areas of learning.”

Recent educational reform movements (such as the Next Generation Science Standards and the Common Core) have also focused on computational thinking as a key skill for K–12 students. For example, the Next Generation Science Standards (NGSS) have identified computational thinking as key scientific and engineering practices that must be understood and applied in learning about the sciences.²⁰ Computational theories, information technologies, and algorithms played a key role in science and engineering in the 20th century; hence, NGSS suggested allowing students to explore datasets using computational and mathematical tools. One example of embedding computational thinking in science classrooms is Project GUTS (Growing Up Thinking Scientifically), which highlights what computational thinking looks like for students using three domains: modeling and simulation, robotics, and game design.¹⁸ Using the NetLogo computational environment, Project GUTS focuses on abstraction, automation, and analysis through a use-modify-create learning progression to allow students to use the tools, as well as modify and create them, thus deepening their acquisition of computational thinking concepts in the context of science learning. Similarly, the Scalable Game Design project engages students in computational thinking concepts through game and simulation design in science classes.²³

While such efforts involve embedded computational thinking in elementary and secondary subject areas, the College Board, with support from the National Science Foundation, has led the effort to introduce students to computational thinking constructs through a standalone advanced-placement course called Computer Science Principles designed to go beyond programming and focus on computational thinking practices to “help students coordinate and make sense of knowledge to accomplish a goal or task.”⁶ The course includes six computational thinking

practices: connecting computing, creating computational artifacts, abstracting, analyzing problems and artifacts, communicating, and collaborating. They are designed to allow students to develop a deep understanding of computational content and how computing is changing our world.⁶ The course is also structured around seven big ideas: creativity, abstraction, data and information, algorithms, programming, the Internet, and global impact. These big ideas from computer science overlap with the computational thinking concepts detailed in the CSTA/ISTE framework outlined earlier, but the Computer Science Principles course also adds programming as a fundamental concept. Programming is the next step in the computational thinking framework, allowing students to develop software and create computational artifacts like visualizations.⁶

Programming allows students to develop and execute algorithms while providing opportunities for them to show creative expression, create new knowledge, and solve problems.⁶ While programming is one of big ideas of Computer Science Principles, the goal is to go beyond learning one particular type of programming language to how computing tools can be used to solve problems through an iterative process.⁶ The Computer Science Principles framework is being used by a number of leading educational organizations across the U.S. to instantiate different versions of the Computer Science Principles course. For example, Project Lead The Way, a nonprofit organization that provides a transformative learning experience for K–12 students and teachers across the U.S. through pathways in computer science, engineering, and biomedical science has rolled out its version of the Computer Science Principles course to more than 400 schools. Another organization, Code.org, is rolling out its own advanced placement computer science curriculum to public school districts across the U.S.

While embedding computational thinking in STEM subject areas or through standalone courses is an important effort, the trans-disciplinary nature of computational thinking competencies provides an opportunity to integrate computational thinking

ideas into all K–12 subject areas. The goal of computational thinking, said Hemmendinger,¹⁶ is “to teach them [students] how to think like an economist, a physicist, an artist, and to understand how to use computation to solve their problems, to create, and to discover new questions that can fruitfully be explored.” Research on embedding computational thinking in K–12 is also starting to emerge and has suggested that students exposed to computational thinking show significant improvement in their problem-solving and critical-thinking skills.⁵ A 2015 study by Calao et al.⁵ reported that integrating computational thinking in a sixth-grade mathematics class significantly improved students’ understanding of mathematics processes when compared to a control group that did not learn computational thinking in their math class.

Although computational thinking is deeply connected to the activity of programming, it is not essential to teach programming as part of a pre-service computational thinking course. Such courses should focus on computational thinking within the context of the teachers’ content areas. Those interested in programming should have access to standalone courses that focus more specifically on programming and computer science. Despite the current lack of clarity as to the definition of computational thinking, Wing’s ideas provide a good starting point for conceptualizing it for teacher educators. Similarly, the computational thinking concepts and practices outlined in the CSTA/ISTE framework exemplify how these concepts can be used across the curriculum and to prepare pre-service teachers. The rest of this article focuses on how to develop pre-service teacher computational thinking competencies not related to programming to allow them to teach computational thinking ideas in their classrooms.

To achieve these goals, we need to prepare new teachers who are able to incorporate computational thinking skills into their discipline and teaching practice so they can guide their students to use computational thinking strategies.²² The following section discusses how the education community can prepare teachers to embed computational thinking in their curricula and

practice and offers recommendations to prepare the next generation of computationally literate teachers.

Computational Thinking and Teacher Education

As discussed previously, researchers have argued that computational thinking needs to be on par with reading, writing, and arithmetic.^{3,28} Recent efforts to train teachers to embed computational thinking have focused on in-service teacher professional development,¹⁸ but there is limited understanding of how to engage pre-service teachers from other content areas in computer science and computational thinking.²⁹ This complication is compounded by the fact that few teacher-preparation institutions offer programs specifically for computer science teachers. Furthermore, certification and licensure of computer science teachers is deeply flawed, as detailed in the “Bugs in the System” report by the Computer Science Teacher Association.⁸ It is vital that teacher education programs address the lack of teacher training around computer science ideas, given the burgeoning grassroots movement and impetus from governments to expand computer science learning opportunities in elementary and secondary classrooms, including the Computer Science For All initiative launched January 2016 in the U.S.

So how do teacher educators develop mechanisms to expose pre-service teachers to computational thinking constructs and understanding within the context of their subject areas? How do we develop pre-service teachers’ knowledgebase so they can provide relevant, engaging, and meaningful computational thinking experiences for their students? Darling-Hammond and Bransford¹² proposed a framework that could be adapted to prepare teachers to incorporate computational thinking, articulating knowledge, skills, and dispositions that teachers should acquire and suggesting that teachers need knowledge of learners, as well as of subject matter and curriculum goals.

Teacher educators need to first develop pre-service teachers’ knowledge and skills on how to think computationally and then how to teach their students to think computationally. It is thus impera-



Project Lead the Way session at San José State University, San Jose, CA.

tive for pre-service teachers to understand computational thinking in the context of the subject area they will be teaching. This requires them to have deep understanding of their own discipline and knowledge of how computational thinking concepts relate to what students are learning in the classroom.²² Moreover, the NRC report on the pedagogical aspects of computational thinking argued that teaching this content could put teachers in new and unfamiliar roles in classrooms where students collaborate to solve complex problems. It is thus important that teacher educators “build on what teachers know and feel comfortable doing.”²¹

Developing pre-service teachers’ competencies to embed computational thinking in their future classrooms requires that they are taught to think computationally, as well as how to teach their students to think computationally, especially in the context of specific subject areas. Teacher-training programs are a natural place to introduce teachers to computational thinking and how to incorporate it in their content. A 2014 study by Yadav et al.²⁹ examined the influence of a one-week computational thinking module on pre-service teachers’ understanding and attitudes toward embedding computational thinking in their future classrooms. Pre-service teachers enrolled in a required introductory educational psychology course were divided into two groups. One (the con-

trol group) did not experience the computational thinking module, while the second (the experimental group) specifically learned about computational thinking ideas by working through the module. The authors found the majority of pre-service teachers in the control group viewed computational thinking as integration of technology in the classroom, whereas the majority of participants in the experimental group developed their understanding of computational thinking as a problem-solving approach by using algorithms/heuristics. Results also suggested pre-service teachers in the experimental group were better able to articulate how to integrate computational thinking in K–12 classrooms as compared to the control group. The results from the study indicate the potential to integrate computational thinking for pre-service teachers within existing teacher-education courses. The authors used examples from daily life, as well as discipline-specific examples, to highlight computational thinking to pre-service teachers. For example, they used an example of giving directions from point A to point B to highlight what an algorithm is (a step-by-step route), the efficiency of algorithms (how to provide the best way to get from A to B), abstraction (how to effectively give any direction), and automation (how to design a system like Google Maps). In another example, Yadav et al.²⁹ showcased the idea of parallel processing by discuss-

ing the quickest way for two friends to buy movie tickets when three lines are available; see the Computational Thinking Modules at http://cs4edu.cs.purdue.edu/comp_think for other examples of how computational thinking constructs were highlighted for pre-service teachers. However, the study by Yadav et al.²⁹ was conducted in a general teacher-education course for pre-service teachers from all content areas, next steps should involve embedding computational thinking concepts into courses for teachers of specific subject areas.

Given the strict sequence of courses for teacher education students, teacher educators need to expose pre-service teachers to computational thinking ideas and competencies through existing coursework. One opportunity that offers a natural fit is to introduce computational thinking within existing educational-technology courses in teacher-education programs. These

courses are typically disconnected from the teaching theories and methods pre-service teachers learn in other education courses, focusing instead on technology (such as Web 2.0 tools to teach).¹⁹ Rather than focus on using educational technology tools, educational-technology courses should be revised to provide pre-service teachers with opportunities to think computationally and experience computational thinking as a generic set of skills and competencies that do not necessarily depend on computers or other educational technology.

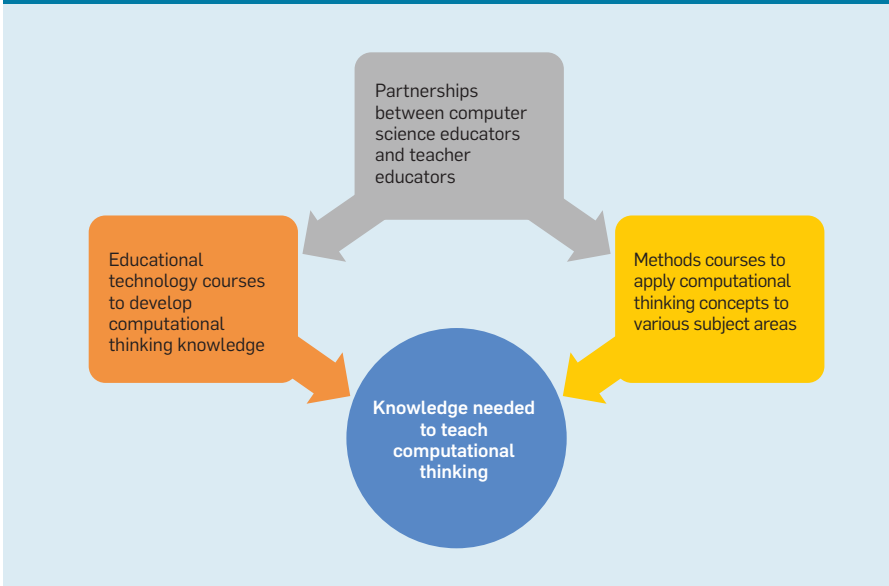
Redesigning introduction-to-educational-technology courses around learning core computational thinking concepts and capabilities is also an opportunity for computer science and education faculty to work together. Taylor²⁵ wrote that many early courses focus on simple programming intended to help students “learn something both

about how computers work and how his or her own thinking works.” About a decade ago, however, educational-technology courses moved away from this view and began to focus instead on use of pre-designed software tools in the classroom. The recent burgeoning movement around computational thinking is an opportunity to reset and redesign educational technology courses, making them both more relevant and more rigorous.

Given the importance of exposing pre-service teachers to computational thinking in the context of their discipline, educational technology courses could be customized for groups of pre-service teachers based on their subject areas and tied to their day-to-day classroom activities. The Technological Pedagogical Content Knowledge (TPACK) framework¹⁹ is a useful model for integrating computational thinking where the related ideas are closely knit within the subject matter and pedagogical approaches pre-service teachers will teach in their future classrooms. TPACK extends Shulman’s idea²⁴ of pedagogical content knowledge by including knowledge teachers need to teach effectively with technology.¹⁹ TPACK suggested teachers learn about effective technology integration within the context of subject matter and pedagogy; similarly, teachers need to develop computational thinking knowledge within the context of their content knowledge and pedagogical knowledge.

Methods courses in teacher-education programs also provide an opportunity to help pre-service teachers incorporate computational thinking in the context of their future subject areas. Methods courses enable them to acquire new ways to think about teaching and learning in one particular subject area and provide opportunities for “developing pedagogical ways of doing, acting, and being as a teacher.”¹ A methods course weaves “together knowledge about subject matter with knowledge about children and how they learn, about the teacher’s role, and about classroom life and its role in student learning.”¹ Within this context, a methods course could also be a place where pre-service teachers explore computational thinking ideas within the context of their specific subject-

Developing pre-service teacher computational thinking.



Recommendations for computational thinking in teacher preparation.

Curriculum. Develop a pre-service teacher education curriculum to prepare teachers to embed computational thinking in their classrooms.

Core ideas. Introduce pre-service teachers to core ideas of computational thinking by redesigning educational technology courses.

Methods courses. Use elementary and secondary methods courses to develop pre-service teachers’ understanding of computational thinking in the context of the discipline.

Collaboration. Computer science educators and teacher educators collaborate on developing computational thinking curricula that goes beyond programming.

Teacher education. Use existing resources and curriculum standards to assimilate computational thinking into pre-service teacher education.

area specializations. For example, in a Methods of Teaching English course, prospective teachers could learn to embed algorithms into a writing activity by asking students to write a detailed recipe—a step-by-step series of instructions—for a favorite food. Similarly, pre-service teachers in a social studies methods course could learn to incorporate data analysis and pattern recognition by having students collect and analyze population statistics and use it to identify and represent trends.³ Data-analysis tools could be as simple as Piktochart, which allows students to represent data and information through infographics, to more advanced tools like Google Charts, which allow students to dynamically represent data using customizable and interactive charts. Pre-service teachers in a science-methods course could be exposed to computational thinking through computational models for demonstrating scientific ideas and phenomena to their future students.²⁷ The computational models could also be used to test hypotheses, as well as solutions to problems.²³ As discussed earlier in this article, students could use tools like NetLogo and Scalable Game Design to develop computational models as they engage in simulation and game design.

The general concepts of computational thinking acquired in the educational-technology course and the discipline-specific computational thinking practices acquired in the methods courses would help pre-service teachers connect computational thinking to content they will cover in their future classrooms. In this way, the constructs of pedagogical content knowledge²⁴ and technological pedagogical content knowledge¹⁹ provide support for developing pre-service teachers' computational thinking knowledge. Specifically, educational-technology courses would serve as a foundation for developing content knowledge for computational thinking. This knowledge would allow teachers to explore core computational thinking ideas, why those ideas are central, and how computational thinking constructs are similar to or differ from other parallel concepts (such as mathematical thinking).²⁴ As pre-service teachers take teaching-methods courses, they

would learn to integrate computational thinking into the context of particular subject areas. This would allow them to learn how to represent and formulate computational thinking in the subject and make it comprehensible to students.²⁴ By engaging pre-service teachers in computational thinking ideas in the context of teaching their content area, teacher educators could better ensure it becomes part of their own and their students' vocabulary and problem-solving tool set.

The Computational Thinking Progression Chart³ provides a starting framework around which teacher educators could begin to shape pre-service teacher experiences in elementary and secondary teacher-education programs. Within elementary education, incorporating computational thinking exercises into literacy learning offers a straightforward transition for teacher candidates. For example, pre-service teachers would be able to explore how to include abstraction into the analysis of themes within prose or poems using textual details or in summarizing text.⁷ They could also build a lesson plan that incorporates data analysis and data representation by having students identify words that depict feelings and comparing how they are represented across different versions of the same story. Similarly, pre-service teachers could embed computational thinking into lesson plans for language arts at the secondary level by allowing students to collect and integrate data/information from multiple sources to visually represent common themes. Elementary- and secondary-level pre-service science teachers could include data collection, analysis, and representation into any activity in which students gather data and identify and represent patterns in that data. Finally, social studies pre-service teachers could explore how to use large datasets (such as census data) to enable students to explore and identify patterns and discuss the implications of the increasing access to large amounts of personal data.

While existing teacher-education courses provide opportunities to introduce pre-service teachers to computational thinking, some programs might consider developing standalone courses and/or certificate programs that allow pre-service teachers to discover

the scope of computational thinking concepts and capabilities, as well as engage with computational tools that nurture development of computational thinking competencies. These courses would ideally be developed by education faculty and computer science faculty collaborating to identify appropriate learning outcomes and available resources.

Because integrating computational thinking into any curriculum involves exposing teachers and students to concepts and practices used by computer scientists, it is important for teacher educators to work closely with computer science faculty. Similarly, education faculty have a nuanced understanding of K–12 curriculum and educational policies that are key to ensuring current computational thinking efforts are successful. A 2016 report by the Computing Research Association⁹ highlighted the need for computer science faculty to establish interdisciplinary connections with colleagues from other disciplines (such as teacher education, educational psychology, and learning sciences). These collaborations included co-developing and co-teaching courses that prepare teachers to teach computational thinking; see Yadav and Korb³¹ for what such a course might look like. Furthermore, faculty could have joint appointments in education and computer science that would enable them to jointly develop programs and collaborate on research around teaching computational thinking.⁸ This would enable computer scientists and teacher educators to collaborate on developing both plugged and unplugged activities to expose pre-service teachers to computational thinking and its implementation. The accompanying figure showcases the interconnectedness of our recommendation for developing pre-service teacher competencies as computer science and teacher educators collaborate to develop computational thinking understanding through education-technology courses. Pre-service teachers then learn how to use that knowledge to teach children to think computationally in the context of a particular subject area through methods courses.

Additionally, many available resources could be incorporated into an educational-technology or subject-spe-

cific methods course that could help pre-service teachers connect computational thinking to their daily lives and to classroom contexts. For example, pre-service teachers could carry out “CS Unplugged” activities (<http://csunplugged.org/>), many of which teach computational thinking skills without needing a computer and are easily adapted to other subjects. Pre-service teachers could also use Scratch—a programming environment that allows students to create programs by dragging and dropping blocks representing core constructs—to create simple programs and animations.

Recognizing the need for teachers to address computational thinking in their curricula and practice, several organizations, including the CSTA, ISTE, and the National Science Teachers Association, are also developing and sharing tools and resources for current and future teachers. Google’s Exploring Computational Thinking website (<http://g.co/exploringCT>) provides more than 130 lesson plans and sample programs aligned with international education standards; a collection of videos demonstrating how computational thinking concepts are used in real-world problem solving; and a “Computational Thinking for Educators” online course (<http://g.co/computationalthinking>). Since 2014, the Computer Science Education Research Group at the University of Adelaide in Australia has been partnering with Google to create introductory courses for implementing Australia’s Digital Technologies Curriculum and teaching computer science and computational thinking at primary and secondary levels, explicitly tied to the Australian curriculum (<https://csdigitaltech.appspot.com>). These resources provide a starting point for teacher educators to incorporate computational thinking ideas and relate them to specific subject area pre-service teachers will go on to teach in their future classrooms.

Conclusion

The 21st century is heavily influenced by computing, making it imperative that teacher educators incorporate computational thinking into elementary and secondary education. This means they must prepare teachers for computational thinking,² empow-

ering them to teach students these higher-order-thinking skills. Teacher-education programs are the opportune time to engage teachers early in their preparation to formulate ways to integrate computational thinking into their practice. Educational-technology and methods courses in elementary and secondary teacher preparation programs are ideal places for teacher educators to discuss computational thinking. The accompanying table summarizes our recommendations for teacher educators to embed computational thinking into teacher-education programs.

In summary, we have emphasized the importance of embedding computational thinking curricula in teacher education and provided recommendations for how teacher educators might be able to do it. For this effort to succeed, however, computer science and education faculty must work collaboratively, as both groups bring complementary expertise in computing and teacher development. C

References

1. Ball, D.L. Breaking with experience in learning to teach mathematics: The role of a pre-service methods course. *For the Learning of Mathematics* 10, 2 (June 1990), 10–16.
2. Barr, D. et al. Computational thinking: A digital age. *Learning & Leading with Technology* (Mar./Apr. 2011), 20–23.
3. Barr, V. and Stephenson, C. Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads* 2, 1 (Mar. 2011), 48–54.
4. Bundy, A. Computational thinking is pervasive. *Journal of Scientific and Practical Computing* 1, 2 (2007), 67–69.
5. Calao, L.A. et al. Developing mathematical thinking with Scratch: An experiment with 6th grade students. In *Proceedings of the Design for Teaching and Learning in a Networked World 10th European Conference on Technology Enhanced Learning* (Toledo, Spain, Sept. 15–18). Springer International Publishing, 2015, 17–27.
6. College Board. *AP Computer Science Principles*, 2014; <https://advancesinap.collegeboard.org/stem/computer-science-principles>
7. Common Core State Standards Initiative. *Common Core State Standards for English Language Arts & Literacy in History/Social Studies, Science, and Technical Subjects*, 2010; http://www.corestandards.org/wp-content/uploads/ELA_Standards.pdf
8. Computer Science Teachers Association. *Bugs in the System: Computer Science Teacher Certification in the U.S.*, 2013; https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/CSTA_BugsInTheSystem.pdf
9. Cooper, S. et al. *The Importance of Computing Education Research*. White paper, Computing Community Consortium, Jan. 14, 2016, 1–12; <http://cra.org/ccc/wp-content/uploads/sites/2/2015/01/CSEdResearchWhitePaper2016.pdf>
10. Csizmadia, A. et al. Computational thinking: A guide for teachers. *Computing at School Community*, 2015, 1–18; <https://community.computingatschool.org.uk/resources/2324>
11. Czerkaski, B.C. and Lyman, E.W. Exploring issues about computational thinking in higher education. *TechTrends* 59, 2 (Mar. 2015), 57–65.
12. Darling-Hammond, L. and Bransford, J., Eds. *Preparing Teachers for a Changing World: What Teachers Should*

- Learn and Be Able to Do. Jossey-Bass, San Francisco, CA, 2005.
13. Denning, P.J. The profession of IT beyond computational thinking. *Commun. ACM* 52, 6 (June 2009), 28–30.
14. Franklin, D. Putting the computer science in computing education research. *Commun. ACM* 58, 2 (Feb. 2015), 34–36.
15. Grover, S. and Pea, R.D. Computational thinking in K–12: A review of the state of the field. *Educational Researcher* 42, 1 (Jan. 2013), 38–43.
16. Hemmendinger, D. A plea for modesty. *ACM Inroads* 1, 2 (June 2010), 4–7.
17. Kahney, H. *Problem Solving: Current Issues*. Open University Press, Buckingham, U.K., 1993.
18. Lee, I. et al. Computational thinking for youth in practice. *ACM Inroads* 2, 1 (Mar. 2011), 32–37.
19. Mishra, P. and Koehler, M.J. Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record* 108, 6 (June 2006), 1017–1054.
20. National Research Council. *A Framework for K–12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*. The National Academies Press, Washington, D.C., 2012.
21. National Research Council. *Report of a Workshop of Pedagogical Aspects of Computational Thinking*. The National Academies Press, Washington, D.C., 2011.
22. National Research Council. *Report of a Workshop on The Scope and Nature of Computational Thinking*. The National Academies Press, Washington, D.C., 2010.
23. Reppenning, A. et al. Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education* 15, 2 (May 2015), 1–31.
24. Shulman, L.S. Those who understand: Knowledge growth in teaching. *Educational Researcher* 15, 2 (Feb. 1986), 4–14.
25. Taylor, R.P. Introduction. In *The Computer in School: Tutor, Tool, Tutee*, R.P. Taylor, Ed. Teachers College Press, New York, 1980, 1–10.
26. The Royal Society. *Shut down or restart? The way forward for computing in UK schools*. The Royal Society, London, U.K., Jan. 2012; <https://royalsociety.org/-/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf>
27. Weintrop, D. et al. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology* 25, 1 (Feb. 2016), 127–147.
28. Wing, J.M. Computational thinking. *Commun. ACM* 49, 3 (Mar. 2006), 33–35.
29. Yadav, A. et al. Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education* 14, 1 (Mar. 2014), 1–16.
30. Yadav, A. et al. Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, Mar. 9–12). ACM Press, New York, 2011, 465–470.
31. Yadav, A. and Korb, J.T. Learning to teach computer science. *Commun. ACM* 55, 11 (Nov. 2012), 31.

Aman Yadav (ayadav@msu.edu) is an associate professor in the College of Education and director of the Masters of Arts in Educational Technology program at Michigan State University, East Lansing, MI.

Chris Stephenson (stephenson@google.com) is the head of Computer Science Education Strategy at Google, Mountain View, CA.

Hai Hong (haihong@google.com) leads the K–12 Education U.S. Outreach team at Google, Mountain View, CA.

Copyright held by the authors.



Watch the authors discuss their work in this exclusive *Communications* video. <http://cacm.acm.org/videos/computational-thinking-for-teacher-education>

CELEBRATING 50 YEARS OF COMPUTING'S GREATEST ACHIEVEMENTS

Since its inauguration in 1966, the ACM A. M. Turing Award has recognized major contributions of lasting importance in computing.

Through the years, it has become the most prestigious technical award in the field, often referred to as the "Nobel Prize of computing."

ACM will celebrate 50 years of the Turing Award and the visionaries who have received it with a conference on June 23 - 24, 2017 at the Westin St. Francis in San Francisco. ACM Turing laureates will join other ACM award recipients and experts in moderated panel discussions exploring how computing has evolved and where the field is headed. Topics include:

- **Advances in Deep Neural Networks**
- **Restoring Personal Privacy without Compromising National Security**
- **Moore's Law Is Really Dead: What's Next?**
- **Quantum Computing: Far Away? Around the Corner? Or Maybe Both at the Same Time?**
- **Challenges in Ethics and Computing**
- **Preserving Our Past for the Future**
- **Augmented Reality: From Gaming to Cognitive Aids and Beyond**

We hope you can join us in San Francisco, or via our live web stream, to look ahead to the future of technology and innovation, and to help inspire the next generation of computer scientists to invent and dream.

For more information and to reserve your spot, visit
www.acm.org/turing-award-50

Program Committee

Craig Partridge
Program Chair

Fahad Dogar
Deputy Program Chair

Karin Breitman

Vint Cerf

Jeff Dean

Joan Feigenbaum

Wendy Hall

Joseph Konstan

David Patterson



CELEBRATING 50 YEARS
OF COMPUTING'S GREATEST ACHIEVEMENTS

ATHMAN BOUGUETTAYA
The University of Sydney

MUNINDAR SINGH
North Carolina State University

MICHAEL HUHNS
University of South Carolina

QUAN Z. SHENG
Macquarie University

HAI DONG
Royal Melbourne Institute of Technology

QI YU
Rochester Institute of Technology

AZADEH GHARI NEIAT
Royal Melbourne Institute of Technology

SAJIB MISTRY
Royal Melbourne Institute of Technology

BOUALEM BENATALLAH
University of New South Wales

BRAHIM MEDJAHED
University of Michigan, Dearborn

MOURAD OUZZANI
Qatar Computing Research Institute

FABIO CASATI
University of Trento

XUMIN LIU
Rochester Institute of Technology

HONGBING WANG
Southeast University

DIMITRIOS GEORGAKOPOULOS
Swinburne University of Technology

LIANG CHEN
Sun Yat-Sen University

SURYA NEPAL
CSIRO

ZAKI MALIK
Texas A&M University

ABDELKARIM ERRADI
Qatar University

YAN WANG
Macquarie University

BRIAN BLAKE
University of Miami

SCHAHRAM DUSTDAR
TU Wein

FRANK LEYMANN
University of Stuttgart

MICHAEL PAPAZOGLU
Tilburg University

Mapping out the challenges and strategies for the widespread adoption of service computing.

A Service Computing Manifesto: The Next 10 Years

CURRENT IT ADVANCES can be likened in their effect and impact to the deep economic and societal transformations that were brought about by the industrial revolution. Recent years have seen the expansion of services in all sectors of the economy, building on the expansion of the Internet.

We define service computing (alternatively termed service-oriented computing) as the discipline that seeks to develop computational abstractions, architectures, techniques, and tools to support services broadly. A service orientation seeks to transform physical, hardware and software assets into a paradigm in which users and assets establish on-demand interactions, binding resources and operations, providing an abstraction layer that



The design of service systems should build upon a formal model of services that enables efficient access to a large service space with diverse functionalities.

Service composition research should extend to non-WSDL-described services or services described by plain text.

Future research in service composition should target large-scale Web and cloud service composition, and trustworthiness of crowdsourcing contributors.

shifts the focus from infrastructure and operations to services.

But, as we argue here, service computing has not fully reached its potential. Technological advances provide an increasing opportunity for service computing. To avoid the mistakes of the past, we propose a manifesto^a—a community declaration of objectives and approaches—as a way to establish common ground among researchers in the field and to guide its future development.³

^a This manifesto is the culmination of a workshop that was organized at RMIT University (Melbourne, Australia) on Dec 8–9, 2014. A number of experts in service computing from around the world attended. The document was also circulated among other key leaders for further input, giving rise to this manifesto.

This manifesto first takes stock of the current state of service computing and then maps out a strategy for leveraging emerging concepts and technologies to deliver on the full potential of the service paradigm. It identifies the major obstacles that hinder the development and potential realization of service computing in the real world; proposes research directions; and draws a roadmap to enable the service computing field to redefine itself and become one of the powerful engines for social and economic activities.

Evolution of IT: Services as the next layer of the computing value chain. Computing has progressed dramatically over the past few decades in delivering automated solutions for an increasing number of areas. In what

» key insights

- **Service computing is a key paradigm that offers cross-disciplinary computational abstractions, architectures, and technologies to support business services.**
- **Service computing has not yet realized its potential, because it has fallen short in addressing the challenges facing business services that go beyond technical aspects, especially in incorporating human concerns; incorporating recent technological advances; and addressing the effect of confusing standards.**
- **A reboot of service computing is essential for it to play its crucial role in the era of cloud computing, big data, the Internet of Things (IoT), and social and mobile computing. It will require rethinking the service life cycle to better incorporate data and interaction semantics and elements of crowdsourcing reputation and trust to provide personalized, explicit value to stakeholders.**

follows, we provide a historical perspective on the evolution of computing. In the early days of computing, the challenge was to represent information in a machine-readable format that consisted of bits and bytes, called data.

Over time, there was keen interest in complementing data with meaning, thus transforming it to information. With further advances in computing came the idea of adding reasoning to information, thus giving rise to knowl-

edge.⁶ The need for higher levels of abstraction has recently led to the notion of adding action to knowledge, resulting in services. Therefore, the abstract definition of a service comes to fulfill the need to act and deliver on knowledge, that is, to provide a way for knowledge to be useful. Accordingly, services are de facto presently considered as the highest level in the computing value chain^b (see Figure 1).

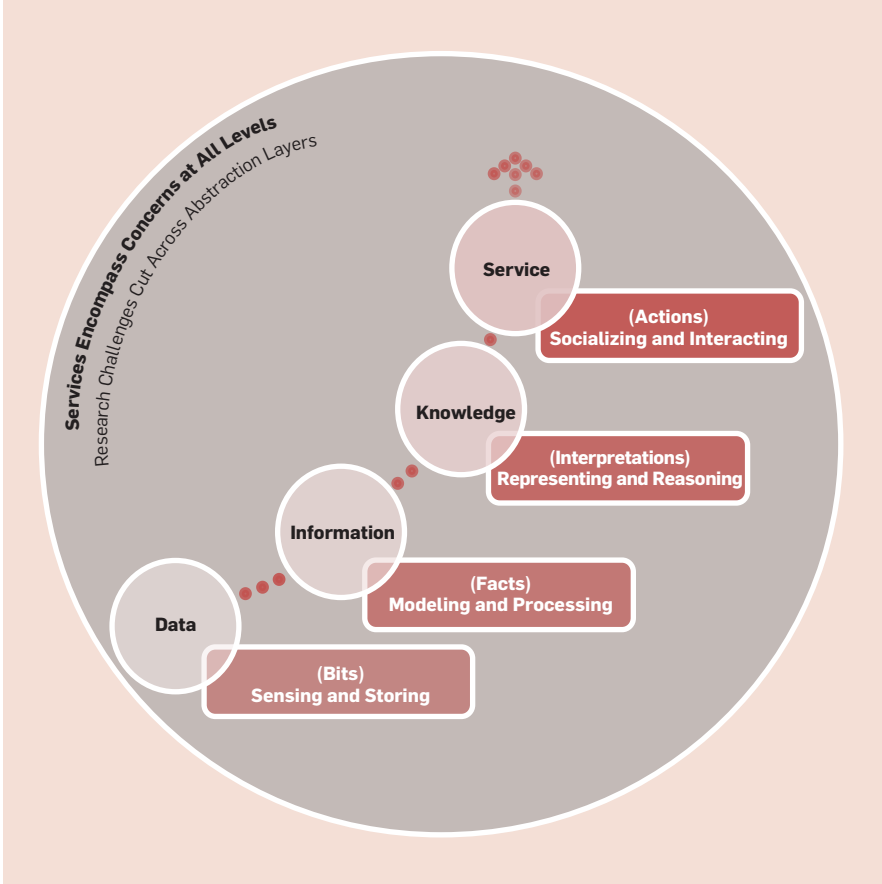
Services are ubiquitous in today's social and economic environment. Examples of such services include healthcare, financial management, human resources, and tourism planning. What distinguishes services from other computing paradigms is their ability to work in a competitive environment where the key parameter to distinguish between similar services is their quality. Knowledge in itself is not sufficient, but needs to be acted upon to bring about benefits. In the case of services, it is the ability to use quality of service (QoS) as a key discriminant to choose between services that provide the "action" on knowledge about services.^{7,19,24}

As economies undergo significant structural changes, digital strategies and innovation must provide industries with tools to create a competitive edge and build more value into their services.⁴ Advances in online service technologies are transforming the Internet into a global workplace, a social forum, a means to manage one's individual affairs and promote collaboration, and a business platform for service delivery. Additionally, organizations are competitively compelled to provide service interfaces to their online services, allowing third-party developers to write auxiliary or satellite "apps" that add new uses to the original service, enrich its features and accessibility, and enhance its agility.

Web service technologies have been developed somewhat independently from the notion of service. They have been at the center of intense research in the past 15 years. In the enterprise market, sales of ready-made software or hardware products are rapidly being replaced by the provisioning of

^b Framing the service paradigm as the top level in the computing value chain does not in any way lessen the importance of issues pertaining to each level in the chain.

Figure 1. Abstractions along the computing value chain.



Representative service computing venues.

Venue	Main Topics in the Last Three Years (in alphabetical order)
IEEE TSC	Business Process Management for Data Services Cloud Service Management Cloud Based Social Computing QoS-Aware Service Composition/Selection/Recommendation for Web/Cloud Services
IJWSR	Service Composition/Selection/Recommendation
ICSOC	Business Processes for Services Cloud Management QoS Management Service Composition
ICWS	QoS-Aware Service Selection/Composition/Recommendation QoS Management Services for Social Networks Service Privacy Service Trust Service Workflows
SCC	Business Processes for Services Cloud Services QoS Management Service Selection/Composition/Recommendation

customized IT Web services, aimed at individual problem solving. Service computing takes a broader view and has emerged as a cross-disciplinary research field that studies the science and technology underlying the popularity of the IT service industry. The ultimate goal of service computing is to bridge the gap between IT and business services to enable IT services to run business services more effectively and efficiently. Web services have so far been the key technology for delivering on service computing.^{4,15,16}

Service computing aims to support the creation and delivery of services, which involve computing devices and software components distributed on the Web and provisioned (and often also controlled) by diverse organizations. Automated composition seeks to mash up these resources to provide customized IT services based on users' goals and preferences.²⁴ To achieve this goal, standardization bodies, such as the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS), have led specification and standardization efforts for implementing service systems. In academia, service computing has attracted intensive attention. A number of journals and conferences were founded to assist research development in this area. The accompanying table lists some major venues and their predominant research topics.^c A state-of-the-art article on service computing can also be found in Yue et al.²⁴

Nevertheless, the state of the art in service computing remains largely in the realm of research and its potential for wide deployment has largely been unfulfilled. One key impediment has been the confusion created by myriad of often-competing Web service standards, attempting to standardize every single aspect of the service life cycle. Moreover, existing Web service standards and technologies do not provide adequate support for the computing needs of key emerging areas, which include mobile computing, cloud

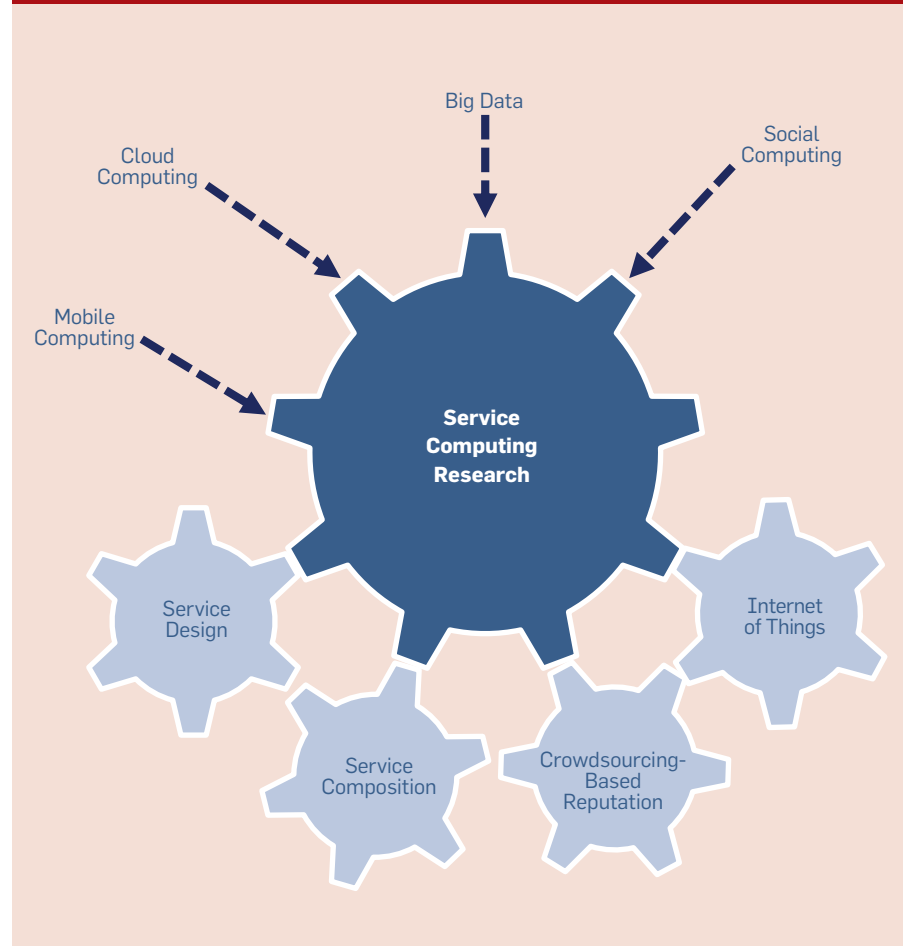
computing, big data, and social computing. These are the four key technologies of the Third Platform named by IDC that are currently influencing the landscape of global business.^d The two impediments (that is, confusion around Web service standards and lack of support for emerging computing grand challenges) have hampered wider and quicker adoption of service computing.

We call for revisiting the practice of superimposing the service paradigm with the technology of delivery (that is, Web services). Although Web services may remain relevant to developing various applications, we call for a focus on their underlying service needs. In particular, we call for the development of a new service paradigm that encompasses the four technologies identified by IDC, along with the introduction of new technologies, rather than emphasizing the expansion of existing Web service standards and technologies.

d <http://www.idc.com/prodserv/3rd-platform/>

An aim of service computing is to harness the power and simplicity of the service paradigm with its functional and non-functional components to build modular software applications and provide a higher level of abstraction for selecting and composing services, thus elevating them to first class object status. A Service Oriented Architecture (SOA) is a technology-independent framework for defining, registering and invoking services.¹⁵ Service computing, however, is broader than SOA and includes the upper level of business process modeling, management, and analysis to the lower level of service data management and analytics. SOA has become a core concept of service computing and provides the fundamental technologies for realizing service computing. The emergence of cloud computing, as a new service delivery model, has inspired researchers and practitioners to explore the use of service computing in the cloud. A large body of research has been devoted to how service-computing concepts are leveraged to facilitate the

Figure 2. Service computing roadmap: In factors and research directions.



c These topics are selected based on the criterion that there were at least five related articles for each topic from 2013 to the venue date. This survey covers all articles for journals and only regular research papers for conferences.

development of cloud applications. As already mentioned, service computing can benefit from and contribute to new directions inspired by the emergence of mobile computing, cloud computing, big data, and social computing, as exemplified in Figure 2. This manifesto outlines the directions.


Since the emergence of the concept of service computing, a number of position and survey articles have been published.^{7,9,16,24} The present effort differs from previous attempts in two main ways. First, the driving factor in this work is to decouple service computing from the technologies to implement service-oriented systems that take full advantage of the promises and expectations of service computing. Second, this work emphasizes the contribution and in of service computing on emerging trends in computing.

The structure of this manifesto starts with analyzing the obstacles to service-computing implementations, followed by examining the major challenges in emerging areas of service computing. We then present a research roadmap to address the identified challenges and draw conclusions and provide our assessments of the prospects for success.


Challenges in Service Computing Research

According to our survey on the articles published in the aforementioned journals and conferences (see the table), current service computing research focuses mostly on seven problem areas: architecture, specification languages, protocols, frameworks, life cycle, quality of service, and the establishment of trust and reputation across the boundaries of autonomous enterprises. The problems are especially apparent in the aforementioned emerging service domains.

An often-overlooked strategic challenge in service computing is analyzing why service computing has not reached its full potential in the real world, and what needs to be done to change that. A barrier that hinders service computing research from being transformed into an effective solution for industry challenges has been the concomitant lack of an easy methodology to transform complex data processing problems into routine services and an easy



What distinguishes services from other computing paradigms is their ability to work in a competitive environment where the key parameter to distinguish between similar services is their quality.



methodology to solve complex service interactions via data-centric architectures, such as Hadoop. A significant challenge is enabling the seamless cooperation of multiple organizations working on different platforms to satisfy consumers' requirements.

This would deliver on the much-needed service composition. Barriers to automated interorganizational service composition include disparities in competency, trust, accountability, functional and non-functional goals (including security and privacy) at the organizational level, and in interaction protocols and representations at the technical level.

We identify four emerging research challenges in service computing: Service design, service composition, crowdsourcing-based reputation, and the Internet of Things (IoT). First, we introduce service design, which is a fundamental but yet unsolved research problem in service computing. Second, we elaborate on challenges of service composition in the context of large-scale Web and cloud service systems, big data, and social networks. Next, crowdsourcing is a cost-effective means for IoT deployment, through collecting sensing data from pervasive sensing devices, for example, mobile phones.¹⁴ We focus on crowdsourcing as a key mechanism for reputation computation. Lastly, we discuss how service computing helps realize the IoT vision and challenges (see Figure 2).

Challenges in service design. Service design is about mapping a formal understanding of the nature of services and relationships thereof. This is an important prerequisite to building sound service systems. Service systems have so far been built without an adequate rigorous foundation that would enable reasoning about them. The preferred approach has been to rely on traditional software engineering approaches, which do not typically take into account the fact that service systems inherently bring together autonomous parts. The Web potentially provides a unique and uniform platform to develop sound service systems. However, there has not been any comprehensive theoretical framework to define and analyze complex service systems on the Web. As a result, only

a few complex service systems have been developed.

Challenges in service composition. Partly fueled by the recent widespread successes of big data, the composition of large numbers of services into a coherent system is an emerging research area. We assess the scale involved as follows. In 2008, a survey discovered 5,077 WSDL-described Web services available on the Internet.¹ In 2016, there are almost 15,000 Web services registered in a public website.^e In addition, we believe that a large proportion of modern Web services are non-WSDL-described, such as those that are cloud-based. The current popularity of cloud computing has inspired the fast growth of cloud-based services and a 2013 survey discovered 6,686 cloud services.¹³ In the era of smartphones, millions of apps are downloadable from cloud-based app stores. It is estimated that there are 1.6 million Android apps and 1.5 million Apple apps as of July 2015.^f Precisely and efficiently searching for services from these large-scale repositories is becoming a critical challenge.^{2,24}

In the setting of big data that might be accessed simultaneously by numerous services, existing service selection, composition, and recommendation approaches that mostly assume a static data environment are inadequate. Composition technologies that address consistency should be explored. In the setting of the IoT, Smart City can be viewed as a typical example of large-scale service composition, where millions of diverse and heterogeneous digital devices and services are integrated dynamically for provisioning multiple real-time functions or user-customized functions. Selecting and composing services from such numerous and ever-changing devices and services to fulfill user requirements in a real-time and context-aware fashion is a difficult mission, which requires urgent attention.²²

In large social networks, such as Facebook and Twitter, in which billions of users register and most users have hundreds of friends or followers on average (on average, 338 friends per

user in Facebook^g in 2010 and 208 followers per user in Twitter^h in 2013), the resultant big data is complex as well as large. Service composition based on social relations poses fundamental serious challenges.

Challenges in crowdsourcing-based reputation. Trust plays an important role in the functioning of a service ecosystem. It is, however, difficult to establish trust when services that offer similar functionalities compete. Reputation is an effective approach in social networks for predicting credibility based on past behavior gleaned from consumers. It is often difficult to ascertain reputation in an open and often anonymous environment. Hence, reputation and crowdsourcing are important approaches for deriving trust.

Computing the reputation of a service entity in a community is realized by collecting all individual users' opinions toward this entity in this community. Crowdsourcing provides an effective means for data collection through collaborations within the community.⁵ Nevertheless, several research challenges remain in the computation of crowdsourcing-based reputation.

Quality of crowdsourcing. The major difference between crowdsourcing and traditional user feedback collection is that crowdsourcing is more likely to use financial rewards and other incentives to motivate participation. However, it is unclear how these factors influence the quality of crowdsourcing. In addition, service users' opinions are typically ambiguous, context-dependent, and based on individual preference. Ambiguity refers to what users express instead what they really think. Users' opinions may also be affected by time, location, and social factors (for example, social relations between service users or between service users and service providers.¹² Some users may have particular preferences on certain service products.¹⁰ There is a strong need to predict the outcome of crowdsourcing reputation given that the reputation is influenced by several dependent factors.

Trustworthiness of crowdsourcing contributors. Some service users' opinions might be unfair and even malicious towards particular service products. Unfairness or maliciousness is context-dependent, affected by the variation in services, time, and locations. A key issue is the selection of crowdsourcing contributors based on their trustworthiness, taking into account the context in which they operate.

Testing platform. Hitherto, there is no standardized testing platform to compare trust and reputation models. There is a strong demand for designing appropriate evaluation metrics to compare trust and reputation models for services.

Challenges in the IoT. The Internet of Things (IoT) is an emerging and promising area that proposes to turn every tangible entity into a node on the Internet. More specifically, the tangible entities ("things") are any Internet-connected sensor, camera, display, smartphone, or other smart communicating devices. The IoT poses two fundamental challenges:¹⁷ communication *with* things, and management *of* things. Service technologies can help address these challenges, through provisioning communication and interoperability means to the IoT, such as REST and service composition models. One challenge is that things are resource-constrained and traditional standards, such as SOAP and BPEL, are too heavyweight to be applicable in the IoT. Furthermore, existing models of service composition cannot be directly used for IoT interoperation, because of architectural differences. In contrast to the single-type Web service component model (such as, services), the IoT component model is heterogeneous and multilayered (for example, devices, data, services, and organizations). Innovative models are required for IoT composition. Specifications of the functionalities of the IoT are a key challenge. Through the diverse, mobile, and context-aware devices, data and services can simultaneously generate diverse and context-aware functions.¹¹ Hence, in the IoT, desired functionality of components is more dynamic and context-aware than in traditional settings, which brings significant complexity in the composition process.

In a nutshell, the IoT involves

e <http://www.programmableweb.com/>

f <http://www.statista.com/statistics/276623/>

number-of-apps-available-in-leading-app-stores/

g <http://www.pewresearch.org/fact-tank/2014/02/03/6-new-facts-about-facebook/>

h <http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/>

incorporating possibly billions of things, and harnessing their data and functionality to provide novel smart services that benefit enterprises, industries, and our society. The fundamental IoT challenges as related to service computing are:

1. Continuously maintaining cyber personalities and contexts for IoT devices. In particular, IoT things need to have Web identities and Web representations (for example, Web proxies) that reflect their physical spaces. They also need to connect and communicate within social, environmental, user-centric, and application contexts, and such contexts need to be maintained and managed.

2. Continually discovering, integrating, and (re)using IoT things and their data. Specifically, the IoT environment is a federated environment where things and their data, cloud services, and IT services (for example, for data analysis and visualization) are often provided by independent providers with diverse interfaces, as well as business, cost, and QoS models. To provide new Internet-scale services, the IoT must (re)use IoT things deployed by others and data collected by others for their own purpose.

Service Computing Research Roadmap

Following the major challenges identified earlier, we now propose a research roadmap for service computing. The roadmap focuses on four emerging research areas, namely service design, service composition, crowdsourcing-based reputation, and Internet of Things.

Service design. Research should focus on laying the foundation for service design drawing from similar research in databases, software engineering, and distributed systems. The design of service systems should build upon a formal model of services that enables efficient access to a large service space with diverse functionalities. The service model can support accessing services much as the relational model supports accessing structured data. As service composition is usually needed to fulfill complex user requests, it is important for the service model to support dependency relationships among different

services and their operations. Similar to the relational model, the service model should enable the design of service query algebra and calculus that allow general users to declaratively query multiple services in a transparent and efficient manner. In addition, service query optimization should go beyond just generating efficient query execution plans. As a large number of service providers may compete to offer similar functionalities, the design of the service model should support the implementation of optimization strategies for finding the “best” Web services or composition thereof with respect to the expected user-supplied quality. In sum, a formal service model that meets the requirements noted here will take center stage to provide efficient and transparent access to computing resources through services, which is a critical step to reach the full potential of service computing.

Recent work²⁵ has shown great promise in designing and using a formal service model to help users access service. A set of algebraic operators, defined based on a formal service model, provides standard ways to manipulate services as first-class objects. These include choosing specific operations from a service as well as quality-based service selection and mashup operators across multiple services to form a composed service. The implementation of these operators enables the generation of service execution plans that can be directly used by users to access services. Due to functional dependency and quality dimensions, a service model is more complex than a relational data model.

In particular, three key features of services are crucial: functionality, behavior, and quality. Functionality is specified by the operations offered by a service; behavior reflects how the service operations can be invoked, and is decided by the dependency constraints between service operations; quality determines the non-functional properties of a service. A viable service query language should allow users to locate and invoke their desired functionalities, select the best service providers that meet their quality requirements, and generate service compositions when functionalities from multiple services are required.

An important approach to design a composition of services is to do so not from the perspective of one party, but from a representation of interactions between two or more autonomous parties. Protocols are specifications of interactions that assiduously disregard the implementations of the services, but describe the interactions: protocols thus promote interoperability among heterogeneous and autonomous services.²³ Newer approaches for protocols accommodate data representations, but additional research is needed to bridge the gap between interaction and data. An important direction is to formalize sociotechnical interactions as a basis for governance of autonomous services.²⁰

Service composition. Future research in service composition should target the following topics:

Large-scale Web and cloud service composition. Service composition research should extend to non-WSDL-described services or services described by plain text. As an example, ProgrammableWeb.com hosts over 10,000 API services and over 6,000 service mashups, with a great majority described in plain text. Web information extraction, natural language processing, data and text mining, collaborative tagging, and information retrieval technologies can be used to extract useful semantics, group relevant services, and detect novel composition patterns. Nevertheless, short service descriptions comprised of limited terms coupled with the diverse naming conventions used by service providers pose novel challenges that demand technological innovations to advance the state of the art in both service computing and all relevant domains. Furthermore, a software system may need to evolve during its life cycle and be able to handle the changes in its running environment and the increasing complexity of its workflows. Self-adaptive software evaluates its behavior and makes adjustment according to the evaluation result to address issues and improve performance.

A cloud-computing environment provides an attractive option for deploying services, because of the potential scalability and accessibility it offers. However, it introduces problems related to:

1. *Maintenance*—The resources are not under the explicit control of the service provider.

2. *Security*—The cloud might not be within the enterprise boundaries of the service provider.

3. *Service-Level Agreements (SLAs)*—Resource allocation is the responsibility of the cloud provider. For example, a service might be unavailable not only due to updates by the service provider, but also due to updates by the cloud provider.

These problems must be addressed as services are moved to clouds and, recently, containers within clouds.

Big data driven service composition. Due to limitations of existing methods, new service selection, composition, and recommendation technologies are key approaches to leveraging state-of-the-art big data research outcomes. One important topic in current big data research is the development of algorithms and models for processing data online. This is fundamentally different from the traditional batch processing approach. Online service composition may provide a promising direction to achieve scalable and adaptive composition solutions to deal with large-scale, highly dynamic, and diverse big data services. Another important consideration is how humans interpret the outcomes of service computing and big data frameworks. As such, it will become increasingly important to relate big data analytic frameworks with the nature of the humans for whom they serve.²¹

Social-network-based service composition. Service selection, recommendation, and composition in large-scale social networks should target combining social network and complex network analysis methods, as well as trust computing techniques. One promising direction is to incorporate social network data that records the interaction of service users with the service data to detect hidden relationship between services and generate potential service compositions. In particular, user activities exhibited through social media services, such as posting experiences, questions and feedbacks about services, could bring novel insight to better understand and leverage both traditional and emerging services. For example, the records of using services



The design of service systems should build upon a formal model of services that enables efficient access to a large service space with diverse functionalities.



to build service mashups posted on public programming forums can be leveraged to recommend interesting services to other users and build new service mashups. As another example, user behavior patterns can be detected from event logs recorded by social media or e-business platforms and used to discover latent knowledge for business process mining. Furthermore, emerging services may bring novel QoS features that go beyond the traditional ones, such as reliability, availability, and response time. Domain-specific quality features that reflect users' interests in choosing and composing services could be extracted through social media services that capture user personal judgment.¹⁰

Crowdsourcing-based reputation.

Future research on crowdsourcing-based reputation of services should target the following directions:

Quality of crowdsourcing. Future studies should focus on how monetary or other interesting factors affect the quality of crowdsourcing data for choosing services. Social studies should be carried out to survey the impact of these interest factors on the reliability of crowdsourcing and the scope of the crowdsourcing contributors. The three factors (ambiguity, context-dependency, and individual preference) that influence the quality of crowdsourcing data should be studied. For the ambiguity issue, future research should focus on how trust assessment questions and scales as well as human computer interactions should be designed to precisely capture users' trust-related perceptions, through the collaboration with cognitive science and human-computer interaction research. The difference between context-dependency and individual preference is that the former relies more on the objective factors, for example, time, location, and social relations, and the latter relies more on the subjective factors, for example, personal experience. These two kinds of factors, for example, social relations and personal preference, may influence each other simultaneously. Future research should target how to model the interrelation between the two groups of factors and how they can be integrated to predict their impact on the quality of crowdsourcing data.

Trustworthiness of crowdsourcing contributors. Future research should focus on studying the context-dependent features of contributors' trustworthiness. The credibility of a crowd member (that is, a human user or another service) determines how much other service consumers may trust its reported ratings regarding the services it has invoked. This allows us to differentiate between service trust and feedback trust. For instance, a service that is not trustworthy as a service provider may be trustworthy as a judge of the behavior of other service providers, and vice versa. Trade-off strategies for selecting users with different cost and trustworthy levels for crowdsourcing should also be explored. Selecting the most appropriate crowdsourcing contributors or crowd workers will require services that interact and combine information from contributors, service providers, and third-party sources, such as job listings and social media.⁸

Internet of Things. In traditional service computing, a service composition focuses on finding an effective combination of component services.¹⁹ Recent work has suggested that a service composition in the IoT needs to find an effective combination of both component services and devices.¹¹ In emerging IoT architectures based on service composition, there is a need to find an effective combination of component services, data services supported by cloud platform services, and devices. Cloud component services are an integral part of the IoT, because they are needed to manage device Web representations, contexts, and related data processing services anywhere and without the need to rely on a computing center.

Complementary to the current work on service discovery and integration, an important and novel direction in IoT research lies in the area of device discovery and integration. Existing results based on a combination of Semantic Sensor Networks (SSN) and OpenIoT¹⁸ provide a device layer architecture and related functionality for IoT device discovery and integration. SSN defines an ontology that is used to describe IoT things and to find devices from the attributes of the data they produce. Nevertheless,

as IoT things are diverse in functionality and access methods, it is infeasible for a single uniform ontology to cover the highly heterogeneous space of things. Information retrieval and text mining techniques can be leveraged to improve the accuracy of device discovery.

Integration of IoT things can be greatly facilitated by discovering the correlations among things. However, correlations among IoT things are usually difficult to detect. Unlike human social networks, where people are well connected, things often have limited explicit interconnections. An interesting direction is multi-hop connections, which leverage human-to-things interactions to correlate IoT things.

Graph-based approaches and machine-learning techniques can help discover hidden interesting relationships among IoT things, and thus suggest interesting and novel integration patterns.

Conclusion

Service computing has a bright future supporting the tremendous advances in emerging areas of computing such as mobile computing, cloud computing, big data, social computing, and beyond. We make the case in this manifesto that the potential of service computing is far greater than what has been achieved so far. We lay down a path forward to take service computing to new heights of innovation. To forge ahead, we make the important statement that, for the service computing paradigm to succeed, it needs to be decoupled from the technology of the day. The challenges may be difficult but the payoffs are great and there is no reason why an ambitious research agenda would not produce enormous benefits for computer science and society.

Acknowledgments

This research was made possible by LP120200305, DP150100149, and DP160103595 grants from Australian Research Council and NPRP 7-481-1-088 and NPRP 9-224-1-049 grants from the Qatar National Research Fund (a member of The Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

1. Al-Masri, E. and Mahmoud, Q.H. Investigating Web services on the World Wide Web. In *Proceedings of the 17th International Conference on World Wide Web* (2008), 795–804.
2. Blake, M.B. and Nowlen, M.E. Knowledge discovery in services (KDS): Aggregating software services to discover enterprise mashups. *IEEE Transactions on Knowledge and Data Engineering* 23, 6 (2011), 889–901.
3. Chesbrough, H. and Spohrer, J. A research manifesto for services science. *Commun. ACM* 49, 7 (July 2006), 35–40.
4. *Commun. ACM* 49, 7 (July 2006). Special Issue: Services Science.
5. Doan, A., Ramakrishnan, R. and Halevy, A. Crowdsourcing systems on the worldwide web. *Commun. ACM* 54, 4 (2011), 86–96.
6. Fricke, M. The knowledge pyramid: A critique of the DIKW hierarchy. *J. Information Science* 35, 2 (2009), 131–142.
7. Huhns, M.N. and Singh, M.P. Service-oriented computing: Key concepts and principles. *IEEE Internet Computing* 9, 1 (2005), 75–81.
8. Jarrett, J., Ferreira da Silva, L., Mello, L., Andere, S., Cruz, G. and Blake, M. Self-generating a labor force for crowdsourcing—Is worker confidence a predictor of quality? In *Proceedings of the 3rd IEEE Workshop on Hot Topics in Web Systems and Technologies*. IEEE, 2015, 85–90.
9. Lemos, A.L., Daniel, F., and Benatallah, B. Web service composition: A survey of techniques and tools. *ACM Computing Surveys* 48, 3 (2015), 1–45.
10. Liu, X., Kale, A., Wasani, J., Ding, C. and Yu, Q. Extracting, ranking, and evaluating quality features of Web services through user review sentiment analysis. In *Proceedings of the 2015 IEEE International Conference on Web Services*. IEEE Computer Society, 153–160.
11. Neiat, A.G., Bouguettaya, A., Sellis, T. and Dong, H. Failure-proof spatio-temporal composition of sensor cloud services. In *Proceedings of 12th International Conference on Service-Oriented Computing*. Springer, 2014, 368–377.
12. Nepal, S., Paris, C. and Bouguettaya, A. Trusting the social Web: Issues and challenges. *World Wide Web Journal* 18 (2015), 1–7.
13. Noor, T., Sheng, Q.Z., Ngu, A. and Dustdar, S. Analysis of Web-scale cloud services. *IEEE Internet Computing* 18, 4 (2014), 55–61.
14. Pal, A. Internet of things: Making the hype a reality. *IT Professional* 17, 3 (2015), 2–4.
15. Papazoglou, M. and Van den Heuvel, W. Service oriented architectures: approaches, technologies and research issues. *The VLDB J.* 16, 3 (2007), 389–415.
16. Papazoglou, M.P. and Georgakopoulos, D. Service-oriented computing. *Commun. ACM* 46, 10 (2003), 24–28.
17. Raggett, D. The Web of things: Challenges and opportunities. *IEEE Computer*, 49, 5 (2015), 26–32.
18. Serrano, M. et al. Defining the stack for service delivery models and interoperability in the internet of things: A practical case with OpenIoT-VDK. *IEEE J. Selected Areas in Commun.* 33, 4 (2015), 676–689.
19. Sheng, Q.Z., Qiao, X., Vasilakos, A.V., Szabo, C., Bourne, S. and Xu, X. Web services composition: A decade's overview. *Information Sciences* 280 (2014), 218–238.
20. Singh, M. Norms as a basis for governing sociotechnical systems. *ACM Trans. Intelligent Systems* 5, 1 (2013), 1–23.
21. Tan, W., Blake, M., Saleh, I. and Dustdar, S. Social-network-sourced big data analytics. *IEEE Internet Computing* 17, 5 (2013), 62–69.
22. Xu, X., Sheng, Q.Z., Zhang, L.-J., Fan, Y. and Dustdar, S. From big data to big service. *IEEE Computer* 48, 7 (2015), 80–83.
23. Yolun, P. and Singh, M. Flexible protocol specification and execution. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems* (2002), 527–534.
24. Yu, Q., Liu, X., Bouguettaya, A. and Medjahed, B. Deploying and managing Web services: Issues, solutions, and directions. *The VLDB Journal* 17, 3 (2008), 537–572.
25. Yu, Q. and Bouguettaya, A. Framework for Web service query algebra and optimization. *ACM Trans. Web* 2, 1 (2008), 1–35.

Correspondence regarding this article should be addressed to **Athman Bouguettaya** (athman.bouguettaya@sydney.edu.au)

research highlights

P. 74

**Technical
Perspective**
**Proving File Systems
Meet Expectations**

By Gernot Heiser

P. 75

Certifying a File System Using Crash Hoare Logic: Correctness in the Presence of Crashes

By Tej Chajed, Haogang Chen, Adam Chlipala, M. Frans Kaashoek,
Nickolai Zeldovich, and Daniel Ziegler

P. 85

**Technical
Perspective**
**Building a Safety Net
for Data Reuse**

By Jonathan Ullman

P. 86

Guilt-Free Data Reuse

By Cynthia Dwork, Vitaly Feldman, Moritz Hardt,
Toniann Pitassi, Omer Reingold, and Aaron Roth

Technical Perspective

Proving File Systems Meet Expectations

By Gernot Heiser

FILE SYSTEMS HAVE BEEN a standard part of computer systems for over 50 years. We routinely (and mostly unthinkingly) use them to store and retrieve persistent data. But how persistent are they really?

This question is far trickier to answer than it may seem at first, for several reasons. Persistent storage media are orders of magnitude slower than non-persistent memories, such as processor registers or random-access memory. They are also organized differently: persistent media are read or written in blocks whose size is on the order of kilobytes but can be up to megabytes. Bridging the speed and granularity gaps requires complex logic for data buffering, implementing efficient lookups, and reordering slow operations to obtain the best performance. Some of this is performed by the operating system, and some of it in the firmware of the storage device, with limited operating-system control.

The file system hides this complexity behind a simple read-write interface, with the promise that data that has been written to a persistent medium can be retrieved correctly at a later time (provided the storage medium itself is not destroyed).

What makes it tricky to fulfill this promise is the fact that computer systems are (*still*) not reliable—they crash. Such a crash may be the result of a physical failure (lack of electrical power), but is more frequently the result of failing software, especially the operating system, or even the file-system implementation itself. Unless great care is taken, such a failure can lead to loss or corruption of supposedly persistent data. For example, a write operation may have updated an internal buffer but not yet reached the persistent medium, resulting in data loss. Reordering may lead to causality violations, where data produced later in the computation is made persistent, while data logically produced before is not. In extreme cases, corrup-

tion of metadata, such as lookup data structures, can result in the loss of data that had been made persistent earlier.

Different file systems go to different lengths in trying to prevent such loss, from making no guarantees at all, to guaranteeing the integrity of metadata, to guaranteeing persistence and integrity of all data written before an explicit or implicit synchronization barrier. But what do those “guarantees” really mean, when they are implemented in complex software that is almost certainly buggy?

In particular, what does it mean that a file system is “crash safe?”

Experience shows that these are serious questions, with even “mature” file systems failing to live up to their promises if crashes happen at particularly inopportune moments. Ideally, one would like a proof that the promise is kept under all circumstances. But you cannot even start on such a proof if you cannot clearly define what crash-safety really means.


The following paper presents a big step toward real-world file systems that are crash-safe in a strict sense. It develops a formalism called crash Hoare logic, which extends the traditional Hoare logic, widely used for reasoning about programs, by a crash condition. This allows the authors to reason about consistency of file systems, when crashes may occur

This is truly remarkable work, combining the development of a new formalism with its application in real systems to prove very complex properties.

at an arbitrary time during execution. It can even deal with crashes happening during recovery, when the file system attempts to reconstruct a consistent state from whatever it finds on the persistent medium after a crash, and in the worst case no progress happens (a situation that could arise due to a crash triggered by a fault in the file system itself).

The authors show how this logic can be applied to actual file systems, including difficulties such as read and write operations that are asynchronous to the execution of the file-system code. They then use this formalism to specify and implement an actual, fully featured file system, and produce a machine-checked and largely automated proof that this file system is actually crash-safe.

This is truly remarkable work, combining the development of a new formalism with its application in real systems to prove very complex properties. It is exactly this kind of properties where formal proofs are most powerful, as anything that is as complex and interconnected as the crash-safety property is notoriously difficult to get right in the implementation. Experience shows it is almost never right—unless formally proved.

Proof automation is very important to making such techniques scale to real-world, high-performance systems. In this paper, the authors have taken a first step toward reducing the manual effort, and their work has already triggered further progress in automating such proof. Other work has demonstrated the feasibility of automating some of the underlying infrastructure, which this work assumes correct without proof. All this means is that complete operating systems, with full proofs of functional correctness, may be closer than we thought only two years ago. 

Gernot Heiser is a Scientia Professor and the John Lions Chair for operating systems at the University of New South Wales.

Copyright held by author.

Certifying a File System Using Crash Hoare Logic: Correctness in the Presence of Crashes

By Tej Chajed, Haogang Chen, Adam Chlipala, M. Frans Kaashoek, Nikolai Zeldovich, and Daniel Ziegler

Abstract

FSCQ is the first file system with a machine-checkable proof that its implementation meets a specification, even in the presence of fail-stop crashes. FSCQ provably avoids bugs that have plagued previous file systems, such as performing disk writes without sufficient barriers or forgetting to zero out directory blocks. If a crash happens at an inopportune time, these bugs can lead to data loss. FSCQ's theorems prove that, under any sequence of crashes followed by reboots, FSCQ will recover its state correctly without losing data.

To state FSCQ's theorems, this paper introduces the Crash Hoare logic (CHL), which extends traditional Hoare logic with a crash condition, a recovery procedure, and logical address spaces for specifying disk states at different abstraction levels. CHL also reduces the proof effort for developers through proof automation. Using CHL, we developed, specified, and proved the correctness of the FSCQ file system. Although FSCQ's design is relatively simple, experiments with FSCQ as a user-level file system show that it is sufficient to run Unix applications with usable performance. FSCQ's specifications and proofs required significantly more work than the implementation, but the work was manageable even for a small team of a few researchers.

1. INTRODUCTION

This paper describes Crash Hoare logic (CHL), which allows developers to write specifications for crash-safe storage systems and also prove them correct. “Correct” means that, if a computer crashes due to a power failure or other fail-stop fault and subsequently reboots, the storage system will recover to a state consistent with its specification (e.g., POSIX¹⁷). For example, after recovery, either all disk writes from a file-system call will be on disk, or none will be. Using CHL we write a simple specification for a subset of POSIX and build the FSCQ *certified* file system, which comes with a machine-checkable proof that its implementation matches the specification.

Proving the correctness of a file system implementation is important, because existing file systems have a long history of bugs both in normal operation and in handling crashes.²⁴ Reasoning about crashes is especially challenging because it is difficult for the file-system developer to consider all possible points where a crash could occur, both while a file-system call is running and during the execution of recovery code. Often, a system may work correctly in many cases, but if a crash happens at a particular point between two specific disk writes, then a problem arises.^{29,39}

Most approaches to building crash-safe file systems fall roughly into three categories (see the SOSP paper³ for a more in-depth discussion of related work): testing, program analysis, and model checking. Although they are effective at finding bugs in practice, none of them can guarantee the absence of crash-safety bugs in actual implementations. This paper focuses precisely on this issue: helping developers build file systems with machine-checkable proofs that they correctly recover from crashes at any point.

Researchers have used theorem provers for certifying real-world systems such as compilers,²³ small kernels,²² kernel extensions,³⁵ and simple remote servers.¹⁵ Some certification projects^{1, 10, 11, 18, 28, 32} have even targeted file systems, as we do, but in each case either the file system was not complete, executable, and ready to run on a real operating system; or its proof did not consider crashes. Reasoning about crash-free executions typically involves considering the states before and after some operation. Reasoning about crashes is more complicated because crashes can expose intermediate states of an operation.

Building an infrastructure for reasoning about file-system crashes poses several challenges. Foremost among those challenges is the need for a specification framework that allows the file-system developer to formalize the system behavior under crashes. Second, it is important that the specification framework allows for proofs to be automated, so that one can make changes to a specification and its implementation without having to redo all of the proofs manually. Third, the specification framework must be able to capture important performance optimizations, such as asynchronous disk writes, so that the implementation of a file system has acceptable performance. Finally, the specification framework must allow modular development: developers should be able to specify and verify each component in isolation and then compose verified components. For instance, once a logging layer has been implemented, file-system developers should be able to prove end-to-end crash safety in the inode layer simply by relying on the fact that logging ensures atomicity; they should not need to consider every possible crash point in the inode code.

CHL addresses these challenges by allowing programmers to specify what invariants hold in case of crashes and

The original version of this paper is entitled “Using Crash Hoare Logic for Certifying the FSCQ File System” and was published in the *Proceedings of the 25th ACM Symposium on Operating Systems Principles (SOSP '15)*.³

by incorporating the notion of a recovery procedure that runs after a crash. CHL supports the construction of modular systems through a notion of logical address spaces. CHL also allows for a high degree of proof automation. Using CHL we specified and proved correct the FSCQ file system, which includes a simple write-ahead log and which uses asynchronous disk writes.

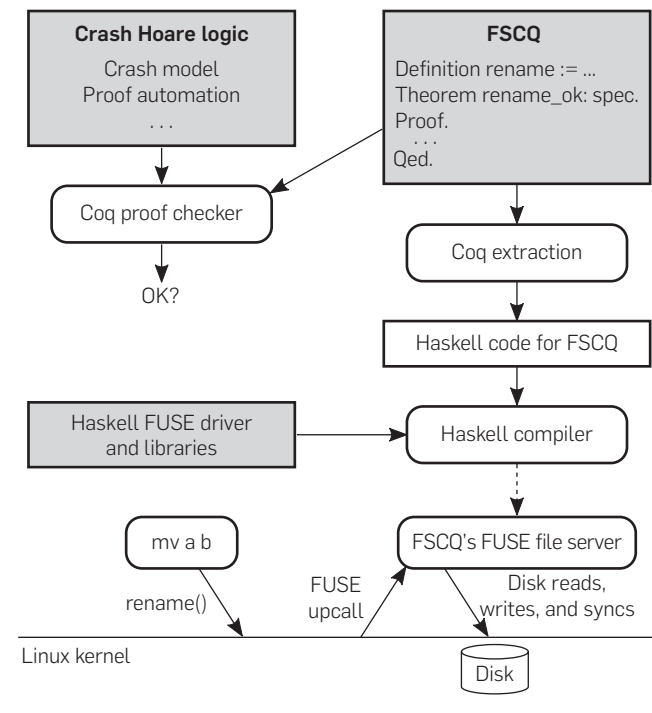
The next section of this article gives an overview of our system architecture, including implementation and proof. Then we introduce CHL, our approach to verifying storage programs that may crash. Afterward, we discuss our prototype file-system implementation FSCQ that we verified with CHL, and we evaluate it in terms of performance, correctness, and other desirable qualities.

2. SYSTEM OVERVIEW

We have implemented the CHL specification framework with the widely used Coq proof assistant,⁸ which provides a single programming language for both proof and implementation. The source code is available at <https://github.com/mit-pdos/fscq>. Figure 1 shows the components involved in the implementation. CHL is a small specification language embedded in Coq that allows a file-system developer to write specifications that include *crash conditions* and *recovery procedures*, and to prove that implementations meet these specifications. We have stated the semantics of CHL and proven it sound in Coq.

We implemented and certified FSCQ using CHL. That is, we wrote specifications for a subset of the POSIX system

Figure 1. Overview of FSCQ’s implementation. Rectangular boxes denote source code; rounded boxes denote processes. Shaded boxes denote source code written by hand. The dashed line denotes the Haskell compiler producing an executable binary for FSCQ’s FUSE file server.



calls using CHL, implemented those calls inside of Coq, and proved that the implementation of each call meets its specification. We devoted substantial effort to building reusable proof automation for CHL. However, writing specifications and proofs still took a significant amount of time, compared to the time spent writing the implementation.

As a standard of completeness for FSCQ, we aimed for the same features as the xv6 file system,⁹ a teaching operating system that implements the Unix v6 file system with write-ahead logging. FSCQ supports fewer features than today’s Unix file systems; for example, it lacks support for multi-processors and deferred durability (i.e., `fsync`). However, it provides the core POSIX file-system calls, including support for large files using indirect blocks, nested directories, and `rename`.

Using Coq’s extraction feature, we do automatic translation of the Coq code for FSCQ into a Haskell program. We run this generated implementation combined with a small (uncertified) Haskell driver as a FUSE¹² user-level file server. This implementation strategy allows us to run unmodified Unix applications but pulls in Haskell, our Haskell driver, and the Haskell FUSE library as trusted components.

3. CRASH HOARE LOGIC

Our goal is to allow developers to certify the correctness of a storage system formally—that is, to prove that it functions correctly during normal operation and that it recovers properly from any possible crashes. As mentioned in the abstract, a file system might forget to zero out the contents of newly allocated directory or indirect blocks, leading to corruption during normal operation, or it might perform disk writes without sufficient barriers, leading to disk contents that might be unrecoverable. Prior work has shown that even mature file systems in the Linux kernel have such bugs during normal operation²⁴ and in crash recovery.³⁸

To prove that an implementation meets its specification, we must have a way for the developer to declare which behaviors are permissible under crashes. To do so, we extend *Hoare logic*,¹⁶ where specifications are of the form $\{P\}$ procedure $\{Q\}$. Here, *procedure* could be a sequence of disk operations (e.g., read and write), interspersed with computation, that manipulates the persistent state on disk, like the implementation of the `rename` system call or a lower-level operation like allocating a disk block. P corresponds to the precondition that should hold before *procedure* is run and Q is the postcondition. To prove that a specification is correct, we must prove that *procedure* establishes Q , assuming P holds before invoking *procedure*. In our `rename` system call example, P might require that the file system be represented by some tree t and Q might promise that the resulting file system is represented by a modified tree t' reflecting the `rename` operation.

Hoare logic is insufficient to reason about crashes, because a crash may cause *procedure* to stop at any point in its execution and may leave the disk in a state where Q does not hold (e.g., in the `rename` example, the new file name has been created already, but the old file name has not yet been removed). Furthermore, if the computer reboots, it often

runs a recovery procedure (such as `fsck`) before resuming normal operation. Hoare logic does not provide a notion that at any point during `procedure`'s execution, a recovery procedure may run. The rest of this section describes how CHL extends Hoare logic to handle crashes.

Traditional Hoare logic distinguishes between partial correctness, where we prove that terminating programs give correct answers, and total correctness, where we also prove termination. We use Coq's built-in termination checker to guarantee that our system calls always finish, when no crashes occur. However, we model cases where a program still runs forever, because it keeps crashing and then crashing again during recovery, ad infinitum. For that reason, our specifications can be interpreted as total correctness for crash-free executions and partial correctness for crashing executions, which makes sense, since the underlying hardware platform refuses to give the programmer a way to guarantee normal termination in the presence of crashes.

3.1. Example

Many file-system operations must update two or more disk blocks atomically. For example, when creating a file, the file system must both mark an inode as allocated as well as update the directory in which the file is created (to record the file name with the allocated inode number). To ensure correct behavior under crashes, a common approach is to use a write-ahead log. Logging guarantees that, if a file-system operation crashed while applying its changes to the disk, then after a crash, a recovery procedure can finish the job by applying the log contents. Write-ahead logging makes it possible to avoid the undesirable intermediate state where the inode is allocated but not recorded in the directory, effectively losing an inode. Many file systems, including widely used ones like Linux's `ext4`,³⁴ employ logging exactly for this reason.

The simple procedure shown in Figure 2 captures the essence of file-system calls that must update two or more blocks. The procedure performs two disk writes using a write-ahead log, which supplies the `log_begin`, `log_commit`, and `log_write` APIs. The procedure `log_write` appends a block's content to an in-memory log, instead of updating the disk block in place. The procedure `log_commit` writes the log to disk, writes a commit record, and then copies the block contents from the log to the blocks' locations on disk. If this procedure crashes and the system reboots, the recovery procedure runs. The recovery procedure looks for the commit record. If there is a commit record, it completes the operation by copying the block contents from the log into the proper locations and then cleans the log.

Figure 2. Pseudocode of `atomic_two_write`.

```
def atomic_two_write(a1, v1, a2, v2):
    log_begin()
    log_write(a1, v1)
    log_write(a2, v2)
    log_commit()
```

If there is no commit record, then the recovery procedure just cleans the log. If there is a crash during recovery, then after reboot the recovery procedure runs again. In principle, this may happen several times. If the recovery finishes, however, then either both blocks have been updated or neither has. Thus, in the `atomic_two_write` procedure from Figure 2, the write-ahead log guarantees that either both writes happen or neither does, no matter when and how many crashes happen.

CHL makes it possible to write specifications for procedures such as `atomic_two_write` and the write-ahead logging system, as we will explain in the rest of the section.

3.2. Crash conditions

CHL needs a way for developers to write down predicates about disk states, such as a description of the possible intermediate states where a crash could occur. To do this, CHL extends Hoare logic with crash conditions, similar in spirit to prior work on fail-stop processors^{33, Section 3} and fault conditions from concurrent work,²⁸ but formalized precisely to allow for executable implementations and machine-checked proofs.

For modularity, CHL should allow reasoning about just one part of the disk, rather than having to specify the contents of the entire disk at all times. For example, we want to specify what happens with the two blocks that `atomic_two_write` updates and not have to say anything about the rest of the disk. To do this, CHL employs *separation logic*,³⁰ which is a way of combining predicates on disjoint parts of a store (in our case, the disk). The basic predicate in separation logic is a points-to relation, written as $a \mapsto v$, which means that address a has value v . Given two predicates x and y , separation logic allows CHL to produce a combined predicate $x \star y$. The \star operator means that the disk can be split into two disjoint parts, where one satisfies the x predicate, and the other satisfies y .

To reason about the behavior of a procedure in the presence of crashes, CHL allows a developer to capture both the state at the end of the procedure's crash-free execution *and* the intermediate states during the procedure's execution in which a crash could occur. For example, Figure 3 shows the CHL specification for FSCQ's `disk_write`. (In our implementation of CHL, these specifications are written in Coq code; we show here an easier-to-read version.) We will describe the precise notation shortly, but for now, note that the specification has four parts: the procedure about which we are reasoning, `disk_write(a, v)`; the precondition, **disk**: $a \mapsto \langle v_0, vs \rangle \star other_blocks$; the postcondition if there are no crashes, **disk**: $a \mapsto \langle v, [v_0] \oplus vs \rangle \star other_blocks$; and the crash condition, **disk**: $a \mapsto \langle v_0, vs \rangle \star$

Figure 3. Specification for `disk_write`.

```
SPEC  disk_write(a, v)
PRE   disk:  $a \mapsto \langle v_0, vs \rangle \star other\_blocks$ 
POST  disk:  $a \mapsto \langle v, [v_0] \oplus vs \rangle \star other\_blocks$ 
CRASH disk:  $a \mapsto \langle v_0, vs \rangle \star other\_blocks \vee$ 
        $a \mapsto \langle v, [v_0] \oplus vs \rangle \star other\_blocks$ 
```


$other_blocks \vee a \mapsto \langle v, [v_0] \oplus vs \rangle \star other_blocks$. Moreover, note that the crash condition specifies that `disk_write` could crash in two possible states (either before making the write or after). In all three logical conditions, *other_blocks* stands for the arbitrary contents of all other disk blocks beside *a*, which should be preserved by this operation, even in case of a crash.

The specification in Figure 3 captures asynchronous writes. To do so, CHL models the disk as a (partial) function from a block number to a tuple, $\langle v, vs \rangle$, consisting of the last-written value *v* and a set of previous values *vs*, any of which could appear on disk after a crash. Block numbers greater than the size of the disk do not map to anything. Reading from a block returns the last-written value, since even if there are previous values that might appear after a crash, in the absence of a crash a read should return the last write. Writing to a block makes the new value the last-written value and adds the old last-written value to the set of previous values. Reading or writing a block number that does not exist causes the system to “fail” (as opposed to finishing or crashing). Finally, CHL’s disk model supports a sync operation, which forces the disk to flush pending writes to persistent storage, modeled in the postcondition for sync by discarding all previous values.

Returning to Figure 3, the `disk_write` specification asserts through the precondition that the address being written, *a*, must be valid (i.e., within the disk’s size), by stating that address *a* points to some value $\langle v_0, vs \rangle$ on disk. The specification’s postcondition asserts that the block being modified will contain the new value $\langle v, [v_0] \oplus vs \rangle$; that is, the new last-written value is *v*, and *v*₀ is added to the set of previous values. The specification also asserts through the crash condition that `disk_write` could crash in a state that satisfies $a \mapsto \langle v_0, vs \rangle \star other_blocks \vee a \mapsto \langle v, [v_0] \oplus vs \rangle \star other_blocks$, that is, either the write did not happen (*a* still has $\langle v_0, vs \rangle$) or it did (*a* has $\langle v, [v_0] \oplus vs \rangle$). Finally, the specification asserts that the rest of the disk is unaffected: if other disk blocks satisfied some predicate *other_blocks* before `disk_write`, they will still satisfy the same predicate afterward.

One subtlety of CHL’s crash conditions is that they describe the state of the disk *just before* the crash occurs, rather than just after. Right after a crash, CHL’s disk model specifies that each block nondeterministically chooses one value from the set of possible values before the crash. For instance, the first line of Figure 3’s crash condition says that the disk still “contains” all previous writes, represented by $\langle v_0, vs \rangle$, rather than a specific value that persisted across the crash, chosen out of $[v_0] \oplus vs$. This choice of representing the state before the crash rather than after the crash allows the crash condition to be similar to the pre- and postconditions. For example, in Figure 3, the state of other blocks just before a crash matches the *other_blocks* predicate, as in the pre- and postconditions. However, describing the state after the crash would require a more complex predicate (e.g., if *other_blocks* contains unsynced disk writes, the state after the crash must choose one of the possible values). Making crash conditions similar to pre- and postconditions is good for proof automation.

The specification of `disk_write` captures two important behaviors of real disks—that the disk controller can defer flushing pending writes to persistent storage and can reorder them—in order to achieve good performance. CHL could model a simpler synchronous disk by specifying that *a* points to a single value (instead of a set of values) and changing the crash condition to say that either *a* points to the new value ($a \mapsto v$) or *a* points to the old value ($a \mapsto v_0$). This change would simplify proofs, but this model of a disk would be accurate only if the disk were running in synchronous mode with no write buffering, which achieves lower performance.

The `disk_write` specification states that blocks are written atomically; that is, after a crash, a block must contain either the last-written value or one of the previous values, and partial block writes are not allowed. This is a common assumption made by file systems and we believe it matches the behavior of many disks in practice. Using CHL, we could capture the notion of partial block writes by specifying a more complicated crash condition, but the specification shown here matches the common assumption. We leave to future work the question of how to build a certified file system without that assumption.

Much like other Hoare-logic-based approaches, CHL requires developers to write a complete specification for every procedure, including internal ones (e.g., allocating an object from a free bitmap). This requires stating precise pre- and postconditions. In CHL, developers must also write a crash condition for every procedure. In practice, we have found that the crash conditions are often simpler to state than the pre- and postconditions. For example, in FSCQ, most crash conditions in layers above the log simply state that there is an active (uncommitted) transaction; only the top-level system-call code begins and commits transactions.

3.3. Logical address spaces

The above example illustrates how CHL can specify predicates about disk contents, but file systems often need to express similar predicates at other levels of abstraction as well. Consider the Unix `pwrite` system call. Its specification should be similar to `disk_write`, except that it should describe offsets and values within the file’s contents, rather than block numbers and block values on disk. Expressing this specification directly in terms of disk contents is tedious. For example, describing `pwrite` might require saying that we allocated a new block from the bitmap allocator, grew the inode, perhaps allocated an indirect block, and modified some disk block that happens to correspond to the correct offset within the file. Writing such complex specifications is also error-prone, which can result in significant wasted effort in trying to prove an incorrect spec.

To capture such high-level abstractions in a concise manner, we observe that many of these abstractions deal with *logical address spaces*. For example, the disk is an address space from disk-block numbers to disk-block contents; the inode layer is an address space from inode numbers to inode structures; each file is a logical address space from offsets

to data within that file; and a directory is a logical address space from file names to inode numbers. Building on this observation, CHL generalizes the separation logic for reasoning about the disk to similarly reason about higher-level address spaces like files, directories, or the logical disk contents in a logging system.

As an example of address spaces, consider the specification of `atomic_two_write`, shown in Figure 4. Rather than describe how `atomic_two_write` modifies the on-disk data structures, the specification introduces new address spaces, `start_state` and `new_state`, which correspond to the contents of the logical disk provided by the logging system. Logical address spaces allow the developer of the logging system to state a clean specification, which provides the abstraction of a simple, synchronous disk to higher layers in the file system. Developers of higher layers can then largely ignore the details of the underlying asynchronous disk.

Specifically, in the precondition, $a_1 \mapsto v_x$ applies to the address space representing the starting contents of the logical disk, and in the postcondition, $a_1 \mapsto v_1$ applies to the new contents of the logical disk. Like the physical disk, these address spaces are partial functions from addresses to values (in this case, mapping 64-bit block numbers to 4 KB block values). Unlike the physical disk, the logical disk address space provided by the logging system associates a single value with each block, rather than a set of values, because the transaction system exports a sound synchronous interface, proven correct on top of the asynchronous interface below. Note how we use a variable `others` to stand for untouched disk addresses in the logical disk, just as we did for the physical disk in Figure 3.

Crucial to such a specification are explicit connections between address spaces. In Figure 4, we use a predicate `log_rep`, whose definition we elide here, but which captures how to map a higher-level state into a set of permissible lower-level states. For this example of a logging layer, the predicate maps a “virtual” disk into a “physical” disk that includes a log. Such predicates may take additional arguments, as with the `NoTxn` argument that we use here to indicate that the logging layer is in a quiescent state, between transactions. This technique for connecting logical layers generalizes to stacks of several layers, as naturally appear in a file system.

The crash condition of `atomic_two_write`, from Figure 4, describes all of the states in which `atomic_two_write` could crash using `log_intact(d1, d2)`, which stands for all possible `log_rep` states that recover to transaction

Figure 4. Specification for `atomic_two_write`.

SPEC	<code>atomic_two_write</code> (a_1, v_1, a_2, v_2)
PRE	disk: <code>log_rep</code> (<code>NoTxn</code> , <code>start_state</code>)
	start_state: $a_1 \mapsto v_x \star a_2 \mapsto v_y \star others$
POST	disk: <code>log_rep</code> (<code>NoTxn</code> , <code>new_state</code>)
	new_state: $a_1 \mapsto v_1 \star a_2 \mapsto v_2 \star others$
CRASH	disk: <code>log_intact</code> (<code>start_state</code> , <code>new_state</code>)

states `d1` or `d2`. Using `log_intact` allows us to capture all possible crash states concisely, including states that can appear deep inside any procedure that `atomic_two_write` might call (e.g., crashes inside `log_commit`).

3.4. Recovery execution semantics

Crash conditions and address spaces allow us to specify the possible states in which the computer might crash in the middle of a procedure’s execution. We also need a way to reason about recovery, including crashes during recovery.

For example, we want to argue that a transaction provides all-or-nothing atomicity: if `atomic_two_write` crashes prior to invoking `log_commit`, none of the calls to `log_write` will be applied; after `log_commit` returns, all of them will be applied; and if `atomic_two_write` crashes during `log_commit`, either all or none of them will take effect. To achieve this property, the transaction system must run `log_recover` after every crash to roll forward any committed transaction, including after crashes during `log_recover` itself.

The specification of the `log_recover` procedure is shown in Figure 5. It says that, starting from any state matching `log_intact(last_state, committed_state)`, `log_recover` will either roll back the transaction to `last_state` or will roll forward a committed transaction to `committed_state`. Furthermore, the fact that `log_recover`’s crash condition implies the precondition indicates that `log_recover` is *idempotent*, meaning that it can be safely restarted after a crash to achieve the same postcondition. (Strictly speaking, this sense of idempotence differs from the mathematical notion, but follows conventions established in early work on fault-tolerant storage systems.¹⁴)

To formalize the requirement that `log_recover` must run after a crash, CHL provides a recovery execution semantics. The recovery semantics talks about *two* procedures executing (a normal procedure and a recovery procedure) and producing either a failure, a *completed* state (after finishing the normal procedure), or a *recovered* state (after finishing the recovery procedure). This regime models the notion that the normal procedure tries to execute and reach a completed state, but if the system crashes, it starts running the recovery procedure (perhaps multiple times if there are crashes during recovery), which produces a recovered state.

Figure 6 shows how to extend the `atomic_two_write` specification to include recovery execution using the \gg notation. The postcondition indicates that, if `atomic_two_write` finishes without crashing, both blocks were updated, and if one or more crashes occurred, with `log_recover` running after each crash, either both blocks were updated

Figure 5. Specification of `log_recover`.

SPEC	<code>log_recover</code> ()
PRE	disk: <code>log_intact</code> (<code>last_state</code> , <code>committed_state</code>)
POST	disk: <code>log_rep</code> (<code>NoTxn</code> , <code>last_state</code>) \vee <code>log_rep</code> (<code>NoTxn</code> , <code>committed_state</code>)
CRASH	disk: <code>log_intact</code> (<code>last_state</code> , <code>committed_state</code>)

or neither was. The special *ret* variable indicates whether the system reached a completed or a recovered state and in this case enables callers of `atomic_two_write` to conclude that, if `atomic_two_write` completed without crashes, it updated both blocks (i.e., updating none of the blocks is allowed only if the system crashed and recovered).

Note that distinguishing the completed and recovered states allows the specification to state stronger properties for completion than recovery. Also note that the recovery-execution version of `atomic_two_write` does not have a crash condition: if the computer crashes, it will run `log_recover` again, and the specification describes what happens when the computer eventually stops crashing and `log_recover` can run to completion.

In this example, the recovery procedure is just `log_recover`, but the recovery procedure of a system built on top of the transaction system may be composed of several recovery procedures. For example, recovery in a file system consists of first reading the superblock from disk to locate the log and then running `log_recover`.

3.5. Proving

In order to prove the correctness of a procedure, CHL follows the standard Hoare-logic approach of decomposing the procedure into smaller units (e.g., control-flow constructs or lower-level functions with already-proven specifications) and chaining their pre- and postconditions according to the procedure’s control flow. Figure 7

Figure 6. Specification for `atomic_two_write` with recovery. The \gg operator indicates the combination of a regular procedure and a recovery procedure.

```

SPEC atomic_two_write( $a_1, v_1, a_2, v_2$ )  $\gg$  log_recover()
PRE disk: log_rep(NoTxn, start_state)
start_state:  $a_1 \mapsto v_x \star a_2 \mapsto v_y \star others$ 
POST disk: log_rep(NoTxn, new_state)  $\vee$ 
      (ret = RECOVERED  $\wedge$  log_rep(NoTxn, start_state))
new_state:  $a_1 \mapsto v_1 \star a_2 \mapsto v_2 \star others$ 
    
```

shows an example of this for a simple procedure; much of this chaining is automated in CHL. The crash condition for a procedure is the disjunction (i.e., “or”) of the crash conditions of all components of that procedure, as illustrated by the red arrows in Figure 7. Finally, proving the correctness of a procedure together with its recovery function requires proving that the procedure’s crash condition implies the recovery precondition, and that recovery itself is idempotent.

4. PROTOTYPE IMPLEMENTATION

The implementation follows the organization shown in Figure 1. FSCQ and CHL are implemented using Coq, which provides a single programming language for implementations, specifications, and proofs. Figure 8 breaks down the source code of FSCQ and CHL. Because Coq provides a single language, proofs are interleaved with source code and are difficult to separate. The development effort took several researchers about a year and a half; most of it was spent on proofs and specifications. Checking the proofs takes 11 h on an Intel i7-3667U 2.00 GHz CPU with 8 GB DRAM. The proofs are complete; we used Coq’s `Print Assumptions` command to verify that FSCQ did not introduce any unproven axioms or assumptions.

CHL. CHL is implemented as a domain-specific language inside of Coq, much like a macro language (or, in the more technical language of proof assistants, we use a shallow embedding). We specified the semantics of this language and proved that it is sound. For example, we proved the standard Hoare-logic specifications for the `for` and `if` combinators. We also proved the specifications of `disk_read`, `disk_write` (whose spec is in Figure 3), and `disk_sync` manually, starting from CHL’s execution and crash model. Much of the automation (e.g., the chaining of pre- and postconditions) is implemented using `Ltac`, Coq’s domain-specific language for proof search.

FSCQ. We implemented FSCQ also inside of Coq, writing the specifications using CHL. We proved that the implementation obeys the specifications, starting from the basic operations in CHL. FSCQ’s write-ahead log simplified specification and implementation tremendously, because much

Figure 7. Example control flow of a CHL procedure that looks up the address of a block in an inode, with support for indirect blocks. (The actual code in FSCQ checks for some additional error cases.) Gray boxes represent the specifications of procedures. The dark red box represents the recovery procedure. Green and pink boxes represent preconditions and crash conditions, respectively. Blue boxes represent postconditions. Dashed arrows represent logical implication.

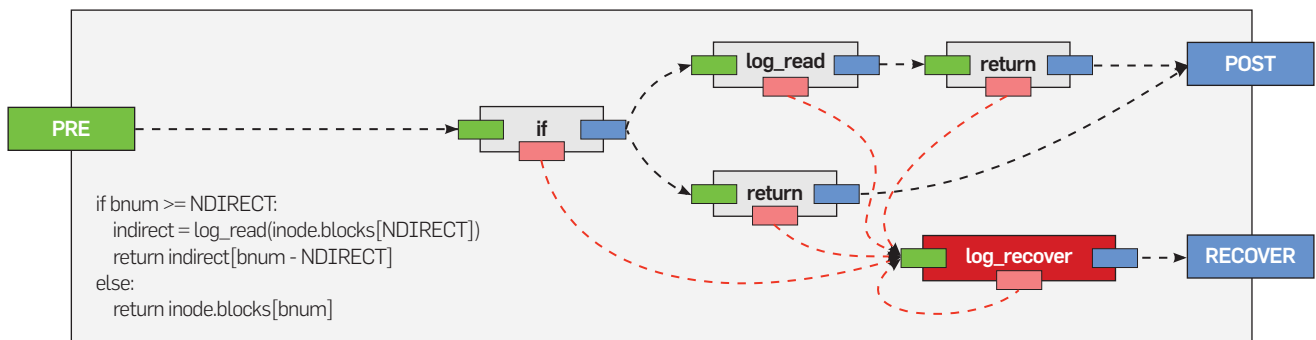


Figure 8. Combined lines of code and proof for FSCQ components.

Component	Lines of code
Fixed-width words	2,709
CHL infrastructure	5,895
Proof automation	2,304
On-disk data structures	7,571
Buffer cache	662
Write-ahead logging	3,191
Bitmap allocator	441
Inodes and files	3,317
Directories	4,451
FSCQ's top-level API	1,198
Total	31,739

of the detailed reasoning about crashes is localized in the write-ahead log.

FSCQ file server. We produced running code by using Coq's extraction mechanism to generate equivalent Haskell code from our Coq implementation. We wrote a driver program in Haskell (400 lines of code) along with an efficient Haskell reimplement of fixed-size words and disk-block operations (350 more lines of Haskell). The extracted code, together with this driver and word library, allows us to efficiently execute our certified implementation.

To allow applications to use FSCQ, we exported FSCQ as a FUSE file system, using the Haskell FUSE bindings² in our Haskell FSCQ driver. We mount this FUSE FSCQ file system on Linux, allowing Linux applications to use FSCQ without any modifications. Compiling the Coq and Haskell code to produce the FUSE executable, without checking proofs, takes a little under 2 min.

Limitations. Although extraction to Haskell simplifies the process of generating executable code from our Coq implementation, it adds the Haskell compiler and runtime into FSCQ's trusted computing base. In other words, a bug in the Haskell compiler or runtime could subvert any of the guarantees that we prove about FSCQ. We believe this is a reasonable trade-off, since our goal is to certify higher-level properties of the file system, and other projects have shown that it is possible to extend certification all the way to assembly.^{6, 15, 22}

Another limitation of the FSCQ prototype lies in dealing with in-memory state in Coq, which is a functional language. CHL's execution model provides a mutable disk but gives no primitives for accessing mutable memory. Our approach is to pass an in-memory state variable explicitly through all FSCQ functions. That variable contains the current buffer-cache state (a map from address to cached block value), as well as the current transaction state, if present (an in-memory log of blocks written in the current transaction). In the future, we want to support multiprocessors where several threads share a mutable buffer cache, and we will address this limitation.

A limitation of FSCQ's write-ahead log is that it does not guarantee how much log space is available to commit a transaction; if a transaction performs too many writes, `log_commit` can return an error. Some file systems deal

with this by reasoning about how many writes each transaction can generate and ensuring that the log has sufficient space before starting a transaction. We have not done this in FSCQ yet, although it should be possible to expose the number of available log entries in the log's representation invariant. Instead, we allow `log_commit` to return an error, in which case the entire transaction (e.g., system call) aborts and returns an error.

5. EVALUATION

To evaluate FSCQ, this section answers several questions:

- Is FSCQ complete enough for realistic applications, and can it achieve reasonable performance? (Section 5.1)
- What kinds of bugs do FSCQ's theorems preclude? (Section 5.2)
- Does FSCQ recover from crashes? (Section 5.3)
- How difficult is it to build and evolve the code and proofs for FSCQ? (Section 5.4)

5.1. Application performance

FSCQ is complete enough that we can use FSCQ for software development, running a mail server, etc. For example, we have used FSCQ with the GNU coreutils (`ls`, `grep`, etc.), editors (`vim` and `emacs`), software development tools (`git`, `gcc`, `make`, etc.), and running a qmail-like mail server. Applications that, for instance, use extended attributes or create very large files do not work on FSCQ, but there is no fundamental reason why they could not be made to work.

Experimental setup. We used a set of applications representing typical software development: cloning a Git repository, compiling the sources of the xv6 file system and the LFS benchmark³¹ using `make`, running the LFS benchmark, and deleting all of the files to clean up at the end. We also run `mailbench`, a qmail-like mail server from the sv6 operating system.⁷ This models a real mail server, where using FSCQ would ensure email is not lost even in case of crashes.

We compare FSCQ's performance to two other file systems: the Linux ext4 file system and the file system from the xv6 operating system (chosen because its design is similar to FSCQ's). We modified xv6 to use asynchronous disk writes and ported the xv6 file system to FUSE so that we can run it in the same way as FSCQ. Finally, to evaluate the overhead of FUSE, we also run the experiments on top of ext4 mounted via FUSE.

To make a meaningful comparison, we run the file systems in synchronous mode; that is, every system call commits to disk before returning. (Disk writes within a system call can be asynchronous, as long as they are synced at the end.) For FSCQ and xv6, this is the standard mode of operation. For ext4, we use the `data=journal` and `sync` options. Although this is not the default mode of operation for ext4, the focus of this evaluation is on whether FSCQ can achieve good performance for its design, not whether its simple design can match that of a sophisticated file system like ext4. To give a sense of how much performance can be obtained through further optimizations or spec changes,

we measure ext4 in three additional configurations: the `journal_async_commit` mode, which uses checksums to commit in one disk sync instead of two (“ext4-journal-async” in our experiments); the `data=ordered` mode, which is incompatible with `journal_async_commit` (“ext4-ordered”); and the default `data=ordered` and `async` mode, which does not sync to disk on every system call (“ext4-async”).

We ran all of these experiments on a quad-core Intel i7-3667U 2.00 GHz CPU with 8 GB DRAM running Linux 3.19. The file system was stored on a separate partition on an Intel SSDSCMMW180A3L flash SSD. Running the experiments on an SSD ensures that potential file-system CPU bottlenecks are not masked by a slow rotational disk. We compiled FSCQ’s Haskell code using GHC 7.10.2.

Results. The results of running our experiments are shown in Figure 9. The first conclusion is that FSCQ’s performance is close to that of the xv6 file system. The small gap between FSCQ and xv6 is due to the fact that FSCQ’s Haskell implementation uses about four times more CPU time than xv6’s. This can be reduced by generating C or assembly code instead of Haskell. Second, FUSE imposes little overhead, judging by the difference between ext4 and ext4-fuse. Third, both FSCQ and xv6 lag behind ext4. This is due to the fact that our benchmarks are bottlenecked by syncs to the SSD, and that ext4 has a more efficient logging design that defers applying the log contents until the log fills up, instead of at each commit. As a result, ext4 can commit a transaction with two disk syncs, compared to four disk syncs for FSCQ and xv6. For example, `mailbench` requires 10 transactions per message, and the SSD can perform a sync in about 2.8 ms. This matches the observed performance of ext4 (64 ms per message) and xv6 and FSCQ (103 and 118 ms per message, respectively).

Finally, there is room for even further optimizations: ext4’s `journal_async_commit` commits with one disk sync instead of two, achieving almost twice the throughput in some cases; `data=ordered` avoids writing file data twice, achieving almost twice the throughput in other cases; and asynchronous mode achieves much higher throughput by avoiding disk syncs altogether (at the cost of not persisting data right away).

5.2. Bug discussion

To understand whether FSCQ eliminates real problems that arise in current file systems, we consider broad categories of bugs that have been found in real-world file systems^{24, 38} and discuss whether FSCQ’s theorems eliminate similar bugs:

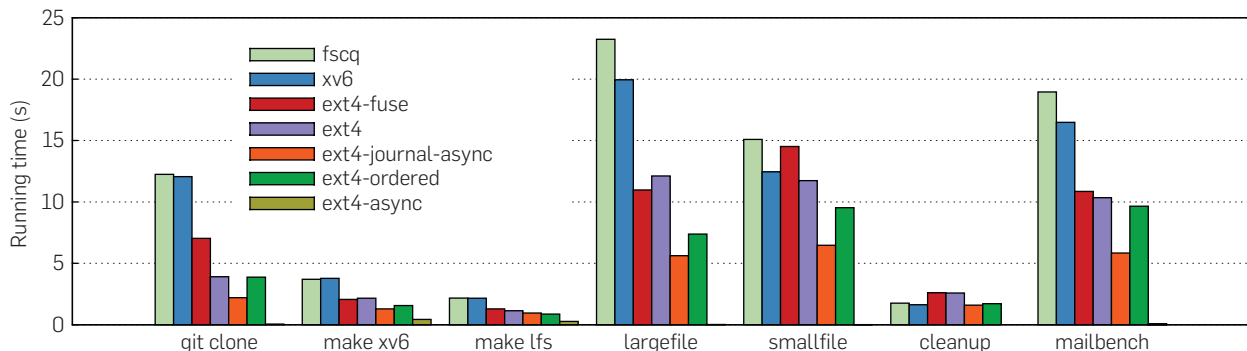
1. Violating file or directory invariants, such as all link counts adding up³⁶ or the absence of directory cycles.²⁶
2. Improper handling of unexpected corner cases, such as running out of blocks during `rename`.¹³
3. Mistakes in logging and recovery logic, such as not issuing disk writes and syncs in the right order.²⁰
4. Misusing the logging API, such as freeing an indirect block and clearing the pointer to it in different transactions.¹⁹
5. Low-level programming errors, such as integer overflows²¹ or double frees.⁴
6. Resource allocation bugs, such as losing disk blocks³⁷ or returning `ENOSPC` when there is available space.²⁷
7. Returning incorrect error codes.⁵
8. Bugs due to concurrent execution of system calls, such as races between two threads allocating blocks.²⁵

Some categories of bugs (#1–5) are eliminated by FSCQ’s theorems and proofs. For example, FSCQ’s representation invariant for the entire file system enforces a tree shape for it, and the specification guarantees that it will remain a tree (without cycles) after every system call.

With regards to resource allocation (#6), FSCQ guarantees resources are never lost, but our prototype’s specification does not *require* that the system be out of resources in order to return an out-of-resource error. Strengthening the specification (and proving it) would eliminate this class of bugs.

Incorrect error codes (#7) can be a problem for our FSCQ prototype in cases where we did not specify what exact code (e.g., `EINVAL` or `ENOTDIR`) should be returned. Extending the specification to include specific error codes could avoid these bugs, at the cost of more complex specifications and proofs. On the other hand, FSCQ can never have a bug where an operation fails without an error code being returned.

Figure 9. Running time for each phase of the application benchmark suite and for delivering 200 messages with `mailbench`.



Multiprocessor bugs (#8) are out of scope for our FSCQ prototype, because it does not support multithreading.

5.3. Crash recovery

We proved that FSCQ implements its specification, but in principle it is possible that the specification is incomplete or that our unproven code (e.g., the FUSE driver) has bugs. To do an end-to-end check, we ran two experiments. First, we ran `fsstress` from the Linux Test Project, which issues random file-system operations; this did not uncover any problems. Second, we experimentally induced crashes and verified that each resulting disk image after recovery is consistent.

In particular, we created an empty file system using `mkfs`, mounted it using FSCQ's FUSE interface, and then ran a workload on the file system. The workload creates two files, writes data to the files, creates a directory and a subdirectory under it, moves a file into each directory, moves the subdirectory to the root directory, appends more data to one of the files, and then deletes the other file. During the workload, we recorded all disk writes and syncs. After the workload completed, we unmounted the file system and constructed all possible crash states. We did this by taking a prefix of the writes up to some sync, combined with every possible subset of writes from that sync to the next sync. For the workload described above, this produced 320 distinct crash states.

For each crash state, we remounted the file system (which runs the recovery procedure) and then ran a script to examine the state of the file system, looking at directory structure, file contents, and the number of free blocks and inodes. For the above workload, this generates just 10 distinct logical states (i.e., distinct outputs from the examination script). Based on a manual inspection of each of these states, we conclude that all of them are consistent with what a POSIX application should expect. This suggests that FSCQ's specifications, as well as the unverified components, are likely to be correct.

5.4. Development effort

The final question is, how much effort is involved in developing FSCQ? One metric is the size of the FSCQ code base, reported in Figure 8; FSCQ consists of about 30,000 lines of code. In comparison, the `xv6` file system is about 3000 lines of C code, so FSCQ is about 10× larger, but this includes a significant amount of CHL infrastructure, including libraries and proof machinery, which is not FSCQ-specific.

A more interesting question is how much effort is required to *modify* FSCQ, after an initial version has been developed and certified. Does adding a new feature to FSCQ require reproofing everything, or is the work commensurate with the scale of the modifications required to support the new feature? To answer this question, the rest of this section presents several case studies, where we had to add a significant feature to FSCQ after the initial design was already complete.

Asynchronous disk writes. We initially developed FSCQ to operate with synchronous disk writes. Implementing asynchronous disk writes required changing about 1000 lines

of code in the CHL infrastructure and changing over half of the implementations and proofs for the write-ahead log. However, layers above the log did not require any changes, since the log provided the same synchronous disk abstraction in both cases.

Buffer cache. We added a buffer cache to FSCQ after we had already built the write-ahead log and several layers above it. Since Coq is a pure functional language, keeping buffer-cache state required passing the current buffer-cache object to and from all functions. Incorporating the buffer cache required changing about 300 lines of code and proof in the log, to pass around the buffer-cache state, to access disk via the buffer cache and to reason about disk state in terms of buffer-cache invariants. We also had to make similar straightforward changes to about 600 lines of code and proof for components above the log.

Optimizing log layout. The initial design of FSCQ's write-ahead log used one disk block to store the length of the on-disk log and another block to store a commit bit, indicating whether log recovery should replay the log contents after a crash. Once we introduced asynchronous writes, storing these fields separately necessitated an additional disk sync between writing the length field and writing the commit bit. To avoid this sync, we modified the logging protocol slightly: the length field was now *also* the commit bit, and the log is applied on recovery if the length is nonzero. Implementing this change required modifying about 50 lines of code and about 100 lines of proof.

5.5. Evaluation summary


Although FSCQ is not as complete and high-performance as today's high-end file systems, our results demonstrate that this is largely due to FSCQ's simple design. Furthermore, the case studies in Section 5.4 indicate that one can improve FSCQ incrementally. In future work, we hope to improve FSCQ's logging to batch transactions and to log only metadata; we expect this will bring FSCQ's performance closer to that of `ext4`'s logging. The one exception to incremental improvement is multiprocessor support, which may require global changes and is an interesting direction for future research.

6. CONCLUSION

This paper's contributions are CHL and a case study of applying CHL to build FSCQ, the first certified crash-safe file system. CHL allowed us to concisely and precisely specify the expected behavior of FSCQ. Via this verification approach, we arrive at a machine-checked proof that FSCQ avoids bugs that have a long history of causing data loss in previous file systems. For this kind of critical infrastructure, the cost of proving seems a reasonable price to pay. We hope that others will find CHL useful for constructing crash-safe storage systems.

Acknowledgments

The authors would like to thank Nathan Beckmann, Butler Lampson, Robert Morris, and the IronClad team for insightful discussions and feedback. The author would

also like to thank the anonymous reviewers for their comments, and to our SOSP shepherd Herbert Bos and CACM Research Highlights editor Martin Abadi. This research was supported, in part, by NSF awards CNS-1053143 and CCF-1253229, Google, and CyberSecurity@CSAIL. 

References

- Amani, S., Hixon, A., Chen, Z., Rizkallah, C., Chubb, P., O'Connor, L., Beeren, J., Nagashima, Y., Lim, J., Sewell, T., Tuong, J., Keller, G., Murray, T., Klein, G., Heiser, G. Cogent: Verifying high-assurance file system implementations. In *Proceedings of the 21th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (Atlanta, GA, Apr. 2016), 175–188.
- Bobbio, J. et al. Haskell bindings for the FUSE library, 2014. <https://github.com/m15k/hfuse>.
- Chen, H., Ziegler, D., Chajed, T., Chlipala, A., Kaashoek, M.F., Zeldovich, N. Using Crash Hoare Logic for certifying the FSCQ file system. In *Proceedings of the 25th ACM Symposium on Operating Systems Principles (SOSP)* (Monterey, CA, Oct. 2015).
- Chinner, D. xfs: Fix double free in xlog_recover_commit_trans, Sept. 2014. <http://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=88b863db97a18a04c90ebd57d84e1b7863114dcb>.
- Chinner, D. xfs: xfs_dir_fsync() returns positive errno, May 2014. <https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=43ec1460a2189fbee87980dd3d3e64cba2f11e1f>.
- Chlipala, A. Mostly-automated verification of low-level programs in computational separation logic. In *Proceedings of the 2011 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)* (San Jose, CA, June 2011), 234–245.
- Clements, A.T., Kaashoek, M.F., Zeldovich, N., Morris, R.T., Kohler, E. The scalable commutativity rule: Designing scalable software for multicore processors. In *Proceedings of the 24th ACM Symposium on Operating Systems Principles (SOSP)* (Farmington, PA, Nov. 2013), 1–17.
- Coq development team. *The Coq Proof Assistant Reference Manual, Version 8.5p1*. INRIA, Apr. 2016. <http://coq.inria.fr/distrib/current/refman/>.
- Cox, R., Kaashoek, M.F., Morris, R.T. Xv6, a simple Unix-like teaching operating system, 2014. <http://pdos.csail.mit.edu/6.828/2014/xv6.html>.
- Ernst, G., Pfahler, J., Schellhorn, G., Reif, W. Inside a verified flash file system: Transactions & garbage collection. In *Proceedings of the 7th Working Conference on Verified Software: Theories, Tools and Experiments* (San Francisco, CA, July 2015).
- Freitas, L., Woodcock, J., Butterfield, A. POSIX and the verification grand challenge: A roadmap. In *Proceedings of 13th IEEE International Conference on Engineering of Complex Computer Systems* (Belfast, Northern Ireland, Mar.–Apr. 2008), 153–162.
- FUSE: Filesystem in userspace, 2013. <http://fuse.sourceforge.net/>.
- Goldstein, A. ext4: Handle errors in ext4_rename, Mar. 2011. <https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=ef6078930263bfcdcf4d4dd2b2cd85254b4cf4f5c>.
- Gray, J. Notes on data base operating systems. In *Operating Systems: An Advanced Course*, R. Bayer, R.M. Graham, and G. Seegmuller, eds. Springer-Verlag, London, UK, 1978, 393–481.
- Hawblitzel, C., Howell, J., Lorch, J.R., Narayan, A., Parno, B., Zhang, D., Zill, B. Ironclad Apps: End-to-end security via automated full-system verification. In *Proceedings of the 11th Symposium on Operating Systems Design and Implementation (OSDI)* (Broomfield, CO, Oct. 2014), 165–181.
- Hoare, C.A.R. An axiomatic basis for computer programming. *Commun. ACM* 12 10 (Oct. 1959), 576–580.
- IEEE (The Institute of Electrical and Electronics Engineers) and The Open Group. The Open Group base specifications issue 7, 2013 edition (POSIX.1–2008/Cor 1–2013), Apr. 2013.
- Joshi, R., Holzmann, G.J. A mini challenge: Build a verifiable filesystem. *Formal Aspects Comput.* 19, 2 (June 2007), 269–272.
- Kara, J. ext3: Avoid filesystem corruption after a crash under heavy delete load, July 2010. <https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=f25f624263445785b94f39739a6339ba9ed3275d>.
- Kara, J. jbd2: Issue cache flush after checkpointing even with internal journal, Mar. 2012. <http://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=79feb521a44705262d15cc819a4117a447b11ea7>.
- Kara, J. ext4: Fix overflow when updating superblock backups after resize, Oct. 2014. <http://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=9378c6768e4fca48971e7b6a9075bc006eda981d>.
- Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cook, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Norrish, M., Kolanski, R., Sewell, T., Tuch, H., Winwood, S. seL4: Formal verification of an OS kernel. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles (SOSP)* (Big Sky, MT, Oct. 2009), 207–220.
- Leroy, X. Formal verification of a realistic compiler. *Commun. ACM* 52, 7 (July 2009), 107–115.
- Lu, L., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H., Lu, S. A study of Linux file system evolution. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST)* (San Jose, CA, Feb. 2013), 31–44.
- Manana, F. Btrfs: Fix race between writing free space cache and trimming, Dec. 2014. <http://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=55507ce3612365a5173dfb080a4baf45d1ef8cd1>.
- Mason, C. Btrfs: Prevent loops in the directory tree when creating snapshots, Nov. 2008. <http://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=ea9e8b11bd1252dcbc23afe9cf1a52ec6aa3c113>.
- Morton, A. [PATCH] ext2/ext3-ENOSPC bug, Mar. 2004. <https://git.kernel.org/cgit/linux/kernel/git/tglx/history.git/commit/?id=5e9087ad3928c9d80cc62b583c3034f864b6d315>.
- Ntzik, G., da Rocha Pinto, P., Gardner, P. Fault-tolerant resource reasoning. In *Proceedings of the 13th Asian Symposium on Programming Languages and Systems (APLAS)* (Pohang, South Korea, Nov.–Dec. 2015).
- Pillai, T.S., Chidambaram, V., Alagappan, R., Al-Kiswany, S., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H. All file systems are not created equal: On the complexity of crafting crash-consistent applications. In *Proceedings of the 11th Symposium on Operating Systems Design and Implementation (OSDI)* (Broomfield, CO, Oct. 2014), 433–448.
- Reynolds, J.C. Separation logic: A logic for shared mutable data structures. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science* (Copenhagen, Denmark, July 2002), 55–74.
- Rosenblum, M., Ousterhout, J. The design and implementation of a log-structured file system. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles (SOSP)* (Pacific Grove, CA, Oct. 1991), 1–15.
- Schellhorn, G., Ernst, G., Pfahler, J., Haneberg, D., Reif, W. Development of a verified flash file system. In *Proceedings of the ABZ Conference* (Toulouse, France, June 2014).
- Schlichting, R.D., Schneider, F.B. Fail-stop processors: An approach to designing fault-tolerant computing systems. *ACM Trans. Comput. Syst.* 1, 3 (1983), 222–238.
- Tweedie, S.C. Journaling the Linux ext2fs filesystem. In *Proceedings of the 4th Annual LinuxExpo* (Durham, NC, May 1998).
- Wang, X., Lazar, D., Zeldovich, N., Chlipala, A., Tatrock, Z. Jitk: A trustworthy in-kernel interpreter infrastructure. In *Proceedings of the 11th Symposium on Operating Systems Design and Implementation (OSDI)* (Broomfield, CO, Oct. 2014), 33–47.
- Wong, D.J. ext4: Fix same-dir rename when inline data directory overflows, Aug. 2014. <https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=d80d448c6c5bdd32605b78a60fe8081d82d4da0f>.
- Xie, M. Btrfs: Fix broken free space cache after the system crashed, June 2014. <https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?id=e570fd27f2c5d7eac3876bccf99e9838d7f911a3>.
- Yang, J., Twohey, P., Engler, D., Musuvathi, M. eXPLoDE: A lightweight, general system for finding serious storage system errors. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI)* (Seattle, WA, Nov. 2006), 131–146.
- Zheng, M., Tucek, J., Huang, D., Qin, F., Lillibridge, M., Yang, E.S., Zhao, B.W., Singh, S. Torturing databases for fun and profit. In *Proceedings of the 11th Symposium on Operating Systems Design and Implementation (OSDI)* (Broomfield, CO, Oct. 2014), 449–464.

Tej Chajed, Haogang Chen, Adam Chlipala, M. Frans Kaashoek, Nickolai Zeldovich, and Daniel Ziegler (tchajed,

[@mit.edu](mailto:dmz), hchen, adamc, kaashoek, [@csail.mit.edu](mailto:nickolai)), MIT CSAIL, Cambridge, MA.

Technical Perspective

Building a Safety Net for Data Reuse

By Jonathan Ullman

MOST EVIDENCE IN the empirical sciences is statistical in nature, and scientists rely on a variety of statistical tests to distinguish valid scientific discoveries from spurious ones. Unfortunately, there is a growing recognition that many important research findings based on statistical evidence are not reproducible, raising the question of whether there is a gap between what these statistical tests ensure and the way they are used.

While there are many reasons that research findings may be non-reproducible this work is focused on just one—*interactivity*. Existing statistical methods assume the procedure being used to analyze the data is fixed before the data is collected. For example, performing a regression analysis on a fixed set of variables. However, in practice, the methods used to analyze a dataset are often chosen based on previous interactions with the same dataset. For example, using the same dataset to first select variables and then perform a regression analysis. Analyzing a fixed dataset in this interactive manner is known to lead to spurious conclusions, even when each procedure is statistically sound in isolation.

How should we study interactive data analysis? The most natural approach is to explicitly model the interactive procedure as a single procedure. For example, there is a vast amount of literature on statistically sound ways to select variables then regress on those variables. But how can we model the entire process of interacting with a dataset from start to finish when there are many steps and the way early steps influence subsequent steps is complex and possibly ill defined?

This work takes a different approach and embraces interactivity. They consider the feasibility of designing a *statistical validity safety net* that goes around a fixed dataset so that it may be analyzed interactively without compromising statistical validity even when the dataset is analyzed in an interactive manner. More

concretely, the dramatis personae are an unknown population and a data analyst who wants to study this population, but only has a random sample from this population. The analyst is allowed to ask very general questions about the population, and each question may depend in an arbitrary, unknown way on the answers to previous questions. The answers are filtered through the safety net, which should ensure the answers remain approximately accurate for the population no matter how the analyst chooses the queries.

The authors of the following paper show there is a way to construct such a safety net. They do so using a substantial strengthening of the folklore result that differentially private algorithms automatically ensure statistical validity, and then deploy the rich toolkit of differentially private algorithms for interactive data analysis to construct such procedures.


Of course, the paper itself is the best way to learn about the results. But for those who become interested in this topic, I will give my own incomplete, very biased, probably already out-of-date summary of the body of work on interactive data analysis that has happened since.

Sharper Bounds. What are the best possible statistical guarantees for interactive data analysis? In a subsequent work with Bassily et al.,^a we gave improved bounds on the error for estimating adaptively chosen statistics, but we still do not know if these bounds are optimal. One reason we cannot currently close this gap is we do not know necessary conditions for ensuring statistical validity in interactive data analysis. The authors show a weaker condition than differential privacy—*bounded max-information*—are sufficient, and several other notions have been considered,^{a,b,c} but we still do not have necessary conditions.

Computational and Statistical Bottlenecks. In a concurrent work

with Moritz Hardt,^d we approached this problem from the other direction, and showed that interactive data analysis is inherently more difficult than non-interactive data analysis. Very roughly, we show if either the dimensionality of the dataset is large, or if the procedure is required to be computationally efficient in the worst case over interactions and datasets, then it is infeasible to ensure statistical validity for a large number of rounds of interaction.

Relaxing the Problem. In my opinion, the most promising direction is to find meaningful relaxations of the model that allow for more useful procedures, and circumvent the bottlenecks we just discussed. The authors' *reusable holdout* is one such relaxation. For another example, Blum and Hardt^e gave an efficient algorithm for a specific application in data science competitions, showing the benefit of a more tailored approach. I believe there are many more opportunities to design useful tools by finding the right balance between generality and specificity.

As you can see, the foundations of interactive data analysis are ready to be developed and brought up to speed with the foundations of non-interactive data analysis. I look forward to reading more work on this exciting topic. 

a Bassily et al. Algorithmic stability for adaptive data analysis. In *Proceedings of STOC'16*.

b D. Russo, J. Zou. Controlling bias in adaptive data analysis using information theory. In *Proceedings of AISTATS'16*.

c R. Rogers, A. Roth, A. Smith, O. Thakkar. Max-information, differential privacy, and post-selection hypothesis testing. In *Proceedings of FOCS'16*.

d M. Hardt, J. Ullman. Preventing false discovery in interactive data analysis is hard. In *Proceedings of FOCS'14*.

e A. Blum, M. Hardt. The Ladder: A reliable leaderboard for machine learning competitions. In *Proceedings of ICML'15*.

Jonathan Ullman is an assistant professor in the College of Computer and Information Sciences at Northeastern University, Boston, MA.

Copyright held by author.

Guilt-Free Data Reuse

By Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth

Abstract

Existing approaches to ensuring the validity of inferences drawn from data assume a fixed procedure to be performed, selected before the data are examined. Yet the practice of data analysis is an intrinsically interactive and adaptive process: new analyses and hypotheses are proposed after seeing the results of previous ones, parameters are tuned on the basis of obtained results, and datasets are shared and reused.

In this work, we initiate a principled study of how to guarantee the validity of statistical inference in adaptive data analysis. We demonstrate new approaches for addressing the challenges of adaptivity that are based on techniques developed in privacy-preserving data analysis.

As an application of our techniques we give a simple and practical method for reusing a holdout (or testing) set to validate the accuracy of hypotheses produced adaptively by a learning algorithm operating on a training set.

1. INTRODUCTION: THE PROBLEM AND WHY IT IS IMPORTANT

From discovering new particles and clinical studies to election results prediction and credit score evaluation, scientific research, and industrial applications rely heavily on statistical data analysis. The goal of statistical data analysis is to enable an analyst to discover the properties of a process or phenomenon by analyzing data samples generated by the process. Fortunately, data samples reflect many properties of the process that generated them: if smoking increases the risk of lung cancer, then we should expect to see a correlation between smoking and lung cancer in samples of medical records. However, data will also exhibit idiosyncrasies that result from the randomness in the process of data sampling and do not say anything about the process that generated them—these idiosyncrasies will disappear if we resample new data from the process. Teasing out the true properties of the process from these idiosyncrasies is notoriously hard and error-prone task. Problems stemming from such errors can be very costly and have contributed to a wider concern about the reproducibility of research findings, most notably in medical research.¹⁸

Statisticians have long established a number of ways to measure the confidence in a result of analysis, most famously p -values and confidence intervals. These concepts allow the analyst to express the probability (over random sampling) that the outcome of an analysis is just an idiosyncrasy that does not hold for the true distribution of the data. Accordingly, the results can be declared statistically significant when this probability is sufficiently small. The guarantee that a confidence interval or p -value provide have a critical caveat; however, they apply only if the analysis procedure was chosen without examining the data to which the procedure is applied.

A simple and well-recognized misuse of this guarantee happens when an analyst performs multiple analyses but reports only the most favorable result (e.g., having the lowest p -value). It is known by many names including the multiple comparisons problem, multiple testing, p -hacking, and data dredging. As a result of such cherry picking, the reported analysis depends on the data, its stated p -value is incorrect and conclusions often invalid. A number of techniques, most notably false discovery rate control,³ have been developed to address multiple comparisons when the set of analyses to be performed is known before the data are gathered. At the same time the practice of data analysis goes well beyond picking the best outcome from a fixed collection of analyses. Data exploration inspires hypothesis generation; results from one test determine which analyses are performed next; one study on a large corpus determines the next study on the same corpus. In short, data analysis in practice is inherently an *adaptive* process.

While very useful, reusing data in adaptive analysis can greatly increase the risk of spurious discoveries. Adaptive choices in analysis can lead to an exponential growth in the number of procedures that would have been performed had the analyst received different data samples. In other words adapting the analysis to data results in an implicit and potentially very large multiple comparisons problem aptly referred to as the “garden of forking paths” by Gelman and Loken.¹⁴

Although not usually understood in these terms, “Freedman’s paradox” is an elegant demonstration of the powerful effect of adaptive analysis on the validity of conclusions.¹³ In Freedman’s simulation an equation is fitted, variables with an insignificant t -statistic are dropped and the equation is refit to this new—adaptively chosen—subset of variables, with famously misleading results: when the relationship between the dependent and explanatory variables is nonexistent, the procedure overfits, and erroneously declaring significant relationships. An excellent and interactive demonstration of variable selection on the results of linear regression analysis can be found in the online article of Aschwanden.¹

While our previous discussion was concerned primarily with applications of statistics; adaptive data analysis presents a similar challenge in machine learning. An

This overview covers materials from two papers by the authors: “Preserving Statistical Validity in Adaptive Data Analysis,” which appeared in *ACM Symposium on Theory of Computing (STOC) 2015*, and “Generalization in Adaptive Data Analysis and Holdout Reuse,” which appeared in *Conference on Neural Information Processing Systems (NIPS) 2015*.

important goal in machine learning is to obtain a predictive model that generalizes well, that is, a model whose accuracy on the data is representative of its accuracy on future data generated by the same process. Indeed, a large body of theoretical and empirical research was developed for ensuring generalization in a variety of settings. In theoretical work, it is commonly assumed that the learning algorithm operates on a freshly sampled dataset. In practice, instead, a dataset is split randomly into two (or sometimes more) parts: the training set and the testing, or holdout, set. The training set is used for learning a predictor and the holdout set is used to estimate the true accuracy of the predictor. Because the predictor is independent of the holdout dataset, such an estimate is a valid estimate of the true prediction accuracy.

However, in practice the holdout dataset is rarely used only once. One prominent example in which a holdout set is often adaptively reused is hyperparameter tuning. Similarly, the holdout set in a machine learning competition, such as the famous ImageNet competition, is typically reused many times adaptively. Other examples include using the holdout set for variable selection, generation of base learners (in aggregation techniques, such as boosting and bagging), checking a stopping condition, and analyst-in-the-loop decisions. Such reuse is known to lead to overfitting to the holdout set (e.g., Refs.^{7,22}).

The literature recognizes the risks and proposes solutions in a number of special cases of adaptive data analysis. Most of them address a single round of adaptivity such as variable selection followed by regression on selected variables or model selection followed by testing and are optimized for specific inference procedures (see Chapter 7 in Ref.¹⁷ for an overview). Yet, to our knowledge, there is no prior work giving a general methodology for addressing the risks of adaptive data reuse over many rounds of adaptivity and without restricting the type of procedures that are performed. We describe such a methodology, based on techniques from privacy-preserving data analysis, together with a concrete application we call the *reusable holdout*.

2. OUR APPROACH AND RESULTS^a

Let us establish some simple terminology. We represent a data point as an element of some universe \mathcal{X} and a dataset consists of n data points. A data generating process gives rise to a probability distribution over datasets. We will focus on the most commonly studied setting in which each point of the dataset is drawn randomly and independently from some unknown distribution \mathcal{P} over \mathcal{X} . For example, the dataset may contain the health information and habits of n individuals, and the analyst is trying to learn about medical conditions affecting the population from which the individuals were drawn randomly.

We view adaptive analysis as a process in which an analyst wishes to ask a sequence of *queries* on a given dataset.

Here a query could refer to an execution of some statistical procedure, a learning algorithm, preprocessing step, or any other inspection of the data. Crucially, after asking the first t queries, the analyst can use the results of those queries to pick the query performed at step $t + 1$. While our approach can be applied to a very general definition of queries, for simplicity we first focus on queries that estimate the mean of a function $\phi: \mathcal{X} \rightarrow [0, 1]$ on the unknown distribution \mathcal{P} or $\mathcal{P}[\phi] = \mathbf{E}_{x \sim \mathcal{P}}[\phi(x)]$. The estimate is required to be correct up to some additive error τ usually referred to as *tolerance* with high probability. Such queries allow the analyst to learn a variety of basic statistics of the population, for example, the fraction of the population over six feet tall. More generally, they allow the analyst to estimate the true means and moments of individual attributes, correlations between attributes and the accuracy of any predictive model. Such queries are referred to as *statistical* in the context of the well-studied statistical query model¹⁹ and have also been studied in statistics as *linear statistical functionals*. It is known that many standard data analyses can be performed using access to statistical queries instead of direct access to data (see Refs.^{4,19} for examples).

Even in this relatively simple setting the question of how many adaptively chosen statistical queries can be correctly answered using n samples drawn i.i.d. from \mathcal{P} has not been previously examined. The conservative approach of using fresh samples for each adaptively chosen query requires n to scale linearly with the number of queries m . We observe that such a bad dependence is inherent in the standard approach of estimating expectations by the exact empirical average on the samples. This is directly implied by “Freedman’s paradox”¹³ and we describe an additional simple example in Ref.⁹ This situation is in stark contrast to the nonadaptive case in which $n = O\left(\frac{\log m}{\tau^2}\right)$ samples suffice to answer m queries with a tolerance τ using empirical averages.

We demonstrate that the problem can be addressed using techniques developed in the context of *differential privacy*, a definition of privacy tailored to privacy-preserving data analysis. Roughly speaking, differential privacy ensures that the probability of observing any outcome from an analysis is “essentially unchanged” by modifying any single dataset element (the probability distribution is over the randomness introduced by the algorithm).

The central insight of the differentially private data analysis is that it is possible to learn statistical properties of a dataset, whereas controlling the amount of information revealed about any dataset element. Our approach is based on the same view of the adaptive data reuse problem: the analyst can be prevented from overfitting to the data if the amount of information about the data revealed to the analyst is limited. To ensure that information leakage is limited, the algorithm needs to control the access of the analyst to the data. We show that this view can be made formal by introducing the notion of maximum information between two random variables. This notion allows us to bound the factor by which uncertainty about the dataset is reduced given the output of the algorithm on this dataset. We describe it in more detail in Section 3.1.

^a Additional averaging over k different partitions is used in k -fold cross-validation.

Our main technical result is a broad *transfer theorem* showing that any analysis that is carried out in a differentially private manner must lead to a conclusion that generalizes to the underlying distribution. This theorem allows us to draw on a rich body of results in differential privacy and to obtain corresponding results for our problem of guaranteeing generalization in adaptive data analysis. We describe this general theorem in detail in Section 3.

A direct corollary of our theorem is that, remarkably, it is possible to answer nearly *exponentially many* adaptively chosen statistical queries (in the size of the data set n). Equivalently, this reduces the *sample complexity* of answering m queries from *linear* in the number of queries to *polylogarithmic*, nearly matching the dependence that is necessary for nonadaptively chosen queries.

THEOREM 1. *There exists an algorithm that given a dataset of size at least $n \geq n_0$, can answer any m adaptively chosen statistical queries so that with high probability, each answer is correct up to tolerance τ , where*

$$n_0 = O\left(\frac{(\log m)^{3/2} \sqrt{\log |\mathcal{X}|}}{\tau^{7/2}}\right).$$

In this bound $\log |\mathcal{X}|$ should be viewed as roughly the *dimension* of the domain. For example, if the underlying domain is $\mathcal{X} = \{0, 1\}^d$, the set of all possible vectors of d -boolean attributes, then $\log |\mathcal{X}| = d$.

Unfortunately, this algorithm for answering queries is not computationally efficient (it has running time linear in the size of the data universe $|\mathcal{X}|$, which is *exponential* in the dimension of the data). Still, we show that it is possible to quadratically improve on the naïve empirical-mean-based approach by using a simple and practical algorithm that perturbs the answer to each query with independent noise.

THEOREM 2. *There exists a computationally efficient algorithm for answering m adaptively chosen statistical queries, such that with high probability, the answers are correct up to tolerance τ , given a data set of size at least $n \geq n_0$ for:*

$$n_0 = O\left(\frac{\sqrt{m} (\log m)^{3/2}}{\tau^{5/2}}\right).$$

A natural question raised by our results is whether there is an efficient algorithm that can answer an exponential number of adaptively chosen queries. This question was addressed in Refs.^{16, 25} who show that under standard cryptographic assumptions no algorithm can improve on the upper bound achieved by our simple algorithm: any algorithm that can answer more than $\approx n^2$ adaptively chosen statistical queries must have running time exponential in $\log |\mathcal{X}|$.

This lower bound implies that practical algorithms that can answer an exponential number of arbitrarily and adaptively chosen queries are unlikely to exist. Yet we show that there is an alternative way to apply our techniques to answer an exponentially large number of queries efficiently. In this application, the analyst splits the dataset into a training set and a holdout set. The analyst can then perform any analysis

on the training dataset, but can only access the holdout set via queries to our *reusable holdout* algorithm. The reusable holdout algorithm allows the analyst to validate her models and statistics against the holdout set. More formally, the analyst can pick any function $\phi : \mathcal{X} \rightarrow [0, 1]$. If the empirical mean of ϕ evaluated on the training set is close to the true expectation $\mathcal{P}[\phi]$, in other words ϕ does not overfit to the training set, then the reusable holdout confirms that there is no overfitting (but provides no additional information). Otherwise, the algorithm returns an estimate of $\mathcal{P}[\phi]$ that answers the statistical query for ϕ .

We describe a specific instantiation of reusable holdout referred to as *Thresholdout*. The number of queries that *Thresholdout* can answer is exponential in the size of the holdout set n as long as the number of times that the analyst overfits (to the training set) is at most quadratic in n . The analysis of *Thresholdout* is based on known techniques in differential privacy and our transfer theorem. In Section 4, we describe *Thresholdout* and its guarantees in detail. We then illustrate the properties of *Thresholdout* using a simple classification algorithm on synthetic data. The classifier produced by the algorithm overfits the data when the holdout set is reused in the standard way, but does not overfit if used with our reusable holdout.

In Ref.¹⁰ we describe additional algorithms for validating results of adaptive queries against the holdout that are based on description length. Our application of this simple and classical technique differs from its standard uses to derive generalization. It leads to algorithms with guarantees that are incomparable to those achieved via differential privacy.

2.1. Related work

The classical approach in theoretical machine learning to ensure that empirical estimates generalize to the underlying distribution is based on the various notions of complexity of the set of functions output by the algorithm, most notably the Vapnik–Chervonenkis (VC) dimension (see Ref.²³ for a textbook introduction). If one has a sample of data large enough to guarantee generalization for all functions in some class of bounded complexity, then it does not matter whether the data analyst chooses functions in this class adaptively or nonadaptively. Our goal, in contrast, is to prove generalization bounds *without* making any assumptions about the class from which the analyst can choose query functions. In this case the adaptive setting is very different from the nonadaptive setting.

An important and related line of work^{6, 20, 24} establishes connections between the *stability* of a learning algorithm and its ability to generalize. Stability is a measure of how much the error of a function output by a learning algorithm is perturbed by the changes to its input dataset. It is known that certain stability notions are necessary and sufficient for generalization.²⁴ Unfortunately, the stability notions considered in these prior works do not compose in the sense that running multiple stable algorithms sequentially and adaptively may result in a procedure that is not stable. Differential privacy is stronger than these previously studied notions of stability, and in particular enjoys strong composition guarantees. This provides a calculus for building up complex

algorithms that satisfy stability guarantees sufficient to give generalization. Our work can thus be interpreted as showing that differential privacy plays the role of stability in the multistep adaptive analysis setting.

There is a very large body of work designing differentially private algorithms for various data analysis tasks, some of which we leverage in our applications (see Ref.⁸ for a short survey and Ref.¹² for a textbook introduction to differential privacy).

For differentially private algorithms that output a hypothesis it has been known as folklore that differential privacy implies stability of the hypothesis to replacing (or removing) an element of the input dataset. Such stability is long known to imply generalization *in expectation* (e.g., Ref.²⁴). Our technique can be seen as a substantial strengthening of this observation: from expectation to high probability bounds (which is crucial for answering many queries), from pure to approximate differential privacy (which is crucial for our improved efficient algorithms), and beyond the expected error of a hypothesis.

Building on our work, Blum and Hardt⁵ showed how to reuse the holdout set to maintain an accurate leaderboard in a machine learning competition that allows the participants to submit adaptively chosen models in the process of the competition (such as those organized by Kaggle Inc.).

Finally, in a recent follow-up work, Bassily et al.² strengthen the link between generalization and approximate differential privacy quantitatively and extend it to the more general class of low-sensitivity queries. Their result leads to bounds on the number of samples that are needed to guarantee generalization that improve on our theorems by a factor of $O(\sqrt{\log(m)/\tau})$.

3. DIFFERENTIAL PRIVACY AND GENERALIZATION

Our results rely on a strong connection we make between differential privacy and generalization. At a high level, we prove that if \mathcal{M} is a differentially private algorithm then the empirical average of a function that it outputs on a random dataset will be close to the true expectation of the function with high probability over the choice of the dataset and the randomness of \mathcal{M} . More formally, for a dataset $S = (x_1, \dots, x_n)$ and a function $\phi : \mathcal{X} \rightarrow [0, 1]$, let $\mathcal{E}_S[\phi] = \frac{1}{n} \sum_{i=1}^n \phi(x_i)$ denote the empirical average of ϕ . We denote a random dataset chosen from \mathcal{P}^n by \mathcal{S} . Standard Chernoff–Hoeffding concentration inequalities for sums of independent random variables imply that for any fixed function ϕ , the empirical average $\mathcal{E}_S[\phi]$ is strongly concentrated around its expectation $\mathcal{P}[\phi]$. However, this statement is no longer true if ϕ is allowed to depend on \mathcal{S} (i.e., what happens if we choose functions adaptively, using previous estimates on \mathcal{S}). However, for a hypothesis output by a differentially private \mathcal{M} on \mathcal{S} (denoted by $\phi = \mathcal{M}(\mathcal{S})$), we show that $\mathcal{E}_S[\phi]$ is close to $\mathcal{P}[\phi]$ with high probability. Before making our statements formal we review the definition of differential privacy.¹¹

DEFINITION 3. A randomized algorithm \mathcal{M} with domain \mathcal{X}^n is (ϵ, δ) -differentially private if for all $\mathcal{O} \subseteq \text{Range}(\mathcal{M})$ and for all pairs of datasets $S, S' \in \mathcal{X}^n$ that differ in a single element:

$$\Pr[\mathcal{M}(S) \in \mathcal{O}] \leq e^\epsilon \cdot \Pr[\mathcal{M}(S') \in \mathcal{O}] + \delta,$$

where the probability space is over the coin flips of the algorithm \mathcal{M} . The case when $\delta = 0$ is sometimes referred to as pure differential privacy, and in this case we may say simply that \mathcal{M} is ϵ -differentially private.

The concentration bounds we obtain for pure differential privacy are almost as strong as those given by the standard Chernoff–Hoeffding concentration inequalities.

THEOREM 4. Let \mathcal{M} be an ϵ -differentially private algorithm that outputs a function from \mathcal{X} to $[0, 1]$. For a random variable S distributed according to \mathcal{P}^n we let $\phi = \mathcal{M}(S)$. Then

$$\Pr[|\mathcal{P}[\phi] - \mathcal{E}_S[\phi]| > \epsilon] \leq 6 \cdot e^{-\epsilon^2 n}.$$

This statement also holds more broadly for an important class of low sensitivity functions. These are functions of a dataset that for some sensitivity Δ satisfy: $|f(S) - f(S')| \leq \Delta$ for all datasets $S, S' \in \mathcal{X}^n$ that differ in a single element. Note that the sensitivity of the empirical average of any function with range $[0, 1]$ on a dataset of size n is at most $1/n$.

We outline the proof idea of this result in Section 3.1. A similar concentration result can also be obtained for (ϵ, δ) -differentially private algorithms although it is not quite as strong and requires a substantially different and more involved proof. Our result for this case (see Ref.⁹) has been recently strengthened and generalized to low-sensitivity queries using a new proof technique by Bassily et al.²

Theorem 4 implies that $|\mathcal{P}[\phi] - \mathcal{E}_S[\phi]| \leq \tau$ holds with high probability whenever ϕ is generated by a differentially private algorithm \mathcal{M} . This might appear to be different from what we need in our application because there the queries are generated by an arbitrary (possibly adversarial) adaptive analyst and we only have control over the query answering algorithm. The connection comes from a crucial property of differential privacy, known as its post-processing guarantee: Any algorithm that can be described as the (possibly randomized) post-processing of the output of a differentially private algorithm is itself differentially private (e.g., Ref.¹²). Hence, although we do not know how an arbitrary analyst is adaptively generating her queries, we do know that if the only access she has to \mathcal{S} is through a differentially private algorithm, then her method of producing query functions must be differentially private with respect to \mathcal{S} . We can therefore, without loss of generality, think of the query answering algorithm and the analyst as a *single* algorithm \mathcal{M} , that is, given a random data set \mathcal{S} and returns a differentially private output query $\phi = \mathcal{M}(\mathcal{S})$.

We also note that the bound in Theorem 4 gives the probability of correctness for each individual answer to a query, meaning that the error probability is for each query and not for all queries at the same time. The bounds we state in Theorems 1 and 2 hold with high probability for all m queries and to obtain them from the bounds in this section, we apply the union bound.

All we are missing now to get an algorithm for answering adaptively chosen statistical queries is an algorithm that satisfies the following two conditions:

1. The algorithm can answer every query ϕ with a value, that is, close (up to error α) to the empirical average of ϕ on the dataset.
2. The algorithm is differentially private.

The problem of providing accurate answers to a large number of queries for the average value of a function on the dataset has been the subject of intense investigation in the differential privacy literature. Such queries are usually referred to as (fractional) *counting queries* or *linear queries* in this context. This allows us to obtain statistical query answering algorithms by using various known differentially private algorithms for answering counting queries. Specifically, our Theorem 1 relies on the algorithm in Ref.¹⁵ that uses the multiplicative weights update algorithm to answer the queries. Our Theorem 2 relies on the basic Laplace noise mechanism and strong composition properties of differential.

In the resulting algorithm, α should be viewed as bounding the empirical error, ϵ should be viewed as bounding the generalization error and $\tau = \alpha + \epsilon$ as bounding the total error. Notice that the standard approach of using empirical averages has the optimal empirical error—it has $\alpha = 0$. However, it is not ϵ -differentially private for any ϵ and, as we pointed out earlier, does not provide any guarantee on the generalization error. At the opposite end, an algorithm which answers queries with a constant, independent of the data, has optimal generalization error, but horrible empirical error. Differentially private algorithms for answering counting queries allow us to explicitly trade off empirical error α with generalization error ϵ to obtain a strong bound on the total error τ .

3.1. Max-information

Intuitively, one way to ensure that the function output by an algorithm \mathcal{M} generalizes is to guarantee that the function does depend too much on the input dataset S . We demonstrate that this intuition can be captured via the notion of *max-information* that we introduce.

DEFINITION 5. *Let X and Y be jointly distributed random variables. The max-information between X and Y , denoted $I_\infty(X; Y)$, is the minimal value of k such that for every x in the support of X and y in the support of Y we have $\Pr[X = x \mid Y = y] \leq 2^k \cdot \Pr[X = x]$.*

It follows immediately from Bayes' rule that $I_\infty(X; Y) = I_\infty(Y; X)$. In our use (X, Y) is going to be a joint distribution (S, ϕ) on (dataset, function) pairs. The dataset S is drawn from distribution \mathcal{P}^n that corresponds to n points drawn i.i.d. from \mathcal{P} . Random variable ϕ represents the function generated by the analyst, whereas interacting with S through our mechanism. Importantly, the analyst may arrive at the function after observing the evaluations of other functions on the same dataset S . Now with each possible function ϕ in the support of ϕ we associate a set of “bad” datasets $R(\phi)$. We later choose $R(\phi)$ to mean the empirical value $\mathcal{E}_S[\phi]$ is far from the true value $\mathcal{P}[\phi]$, that is, ϕ overfits to S . Maximum information gives a bound on the probability that S falls in $R(\phi)$.

THEOREM 6. *For $k = I_\infty(S; \phi)$, $\Pr[S \in R(\phi)] \leq 2^k \cdot \max_\phi \Pr[S \in R(\phi)]$.*

The proof follows easily by first decomposing the event $S \in R(\phi)$ into events, $S \in R(\phi) \& \phi = \phi$ for all ϕ . Namely,

$$\Pr[S \in R(\phi)] = \sum_\phi \Pr[S \in R(\phi) \& \phi = \phi].$$

Since

$$\Pr[S \in R(\phi) \& \phi = \phi] = \Pr[S \in R(\phi) \mid \phi = \phi] \cdot \Pr[\phi = \phi],$$

we can apply the definition of max-information and obtain that $\Pr[S \in R(\phi) \mid \phi = \phi] \leq 2^k \Pr[S \in R(\phi)]$. Substituting this bound back into the decomposition gives the desired result:

$$\begin{aligned} \Pr[S \in R(\phi)] &\leq \sum_\phi 2^k \cdot \Pr[S \in R(\phi)] \cdot \Pr[\phi = \phi] \\ &\leq 2^k \cdot \max_\phi \Pr[S \in R(\phi)]. \end{aligned}$$

Our theorem is completely general in the sense that the random variable ϕ does not have to be supported on functions over \mathcal{X} and could instead assume values in any other discrete domain. For example, such output could be a set of features of the data to be used for a subsequent supervised learning task. For our main application ϕ refers to a function, and we denote the set of datasets on which the empirical estimator has error greater than τ as

$$R_\tau(\phi) = \{S \in \mathcal{X}^n : \mathcal{E}_S[\phi] - \mathcal{P}[\phi] > \tau\}. \quad (1)$$

By Hoeffding's bound we know that $\max_\phi \Pr[S \in R_\tau(\phi)] \leq \exp(-2\tau^2n)$. This gives the following immediate corollary.

COROLLARY 7. *If $I_\infty(S; \phi) \leq \log_2 e \cdot \tau^2n$, then $\Pr[S \in R_\tau(\phi)] \leq \exp(-\tau^2n)$.*

To apply this corollary all we need is to show that pure differential privacy implies a sufficiently strong bound on max information $I_\infty(S; \phi)$.

THEOREM 8. *Let \mathcal{M} be an ϵ -differentially private algorithm. Let S be any random variable over n -element input datasets for \mathcal{M} and let Y be the corresponding output distribution $Y = \mathcal{M}(S)$. Then $I_\infty(S; Y) \leq \log_2 e \cdot \epsilon n$.*

The proof of this theorem follows from observing that, any two datasets S and S' differ in at most n elements. Therefore, applying the guarantee of differential privacy n times, we obtain that for every y ,

$$\Pr[Y = y \mid S = S] \leq e^{\epsilon n} \Pr[Y = y \mid S = S'].$$

As there must exist a dataset y such that $\Pr[Y = y \mid S = S'] \leq \Pr[Y = y]$ we can conclude that for every S and every y it holds that $\Pr[Y = y \mid S = S] \leq e^{\epsilon n} \Pr[Y = y]$. This yields the desired bound $I_\infty(S; Y) = I_\infty(Y; S) \leq \log_2 e \cdot \epsilon n$.

From Theorem 8 and Corollary 7, we see that ensuring τ^2 -differential privacy over the entire interaction with the dataset strictly controls the probability that the adversary can choose a function that overfits to the dataset. This is somewhat worse than the claim in Theorem 4 which requires τ -differential privacy. In Ref.,¹⁰ we show that by considering a simple relaxation of max-information, referred to as approximate max-information, it is possible to prove the stronger bound on max-information of differentially private algorithms for datasets consisting of i.i.d. samples. Interestingly, it is not hard to show that algorithms whose output has short description length (in bits) also have low approximate max-information. Thus approximate max-information unifies generalization bounds obtained via (pure) differential privacy and description length. In addition, composition properties of approximate max-information imply that one can easily obtain generalization guarantees for adaptive sequences of algorithms, some of which are differentially private, and others of which have outputs with short description length.

4. THE REUSABLE HOLDOUT

In this section, we describe a practical application of our framework, which gives a method for safely reusing a holdout set many times. In this application, the analyst splits the dataset into a training set and a holdout set. An advantage of this approach is that the data analyst will have full, unrestricted access to the training set and can use it in any way that she desires. The holdout set can only be accessed through a reusable holdout algorithm. The goal of this algorithm is to validate the results of analyses performed on the training set.

We describe a specific instantiation of reusable holdout, referred to as Thresholdout, that validates the values of statistical queries and is based on the “Sparse Vector” technique from differential privacy (e.g., Chapter 3 of Ref.¹²). Specifically, for every function $\phi: \mathcal{X} \rightarrow [0, 1]$ given by the analyst, the algorithm checks if the empirical average of ϕ on the training set is close to the true mean of ϕ (up to some tolerance τ). If the values are close the algorithm does not provide any additional information to the analyst. Only if ϕ overfits the training set does the algorithm provide a valid estimate of the true expectation of ϕ . The result is that for all of the queries that the analyst asks, she has correct estimates of the true expectation—either our algorithm certifies that the estimate from the training set is approximately correct or else it provides a correct estimate using the holdout set. The analysis of the algorithm shows that the number of samples needed by Thresholdout depends only logarithmically on the total number of queries asked by the data analyst as long as the total number of queries that overfit the training set (and have to be answered using the holdout set) is not too large. As a result, this simple and computationally efficient algorithm can potentially answer an exponential (in n) number of queries.

More formally, Thresholdout is given access to the training dataset S_t and holdout dataset S_h and a budget limit B . It allows any query of the form $\phi: \mathcal{X} \rightarrow [0, 1]$ and its goal is

to provide an estimate of $\mathcal{P}[\phi]$. To achieve this the algorithm gives an estimate of $\mathcal{E}_{S_h}[\phi]$ in a way that prevents overfitting of functions generated by the analyst to the holdout set. In other words, responses of Thresholdout are designed to ensure that, with high probability, $\mathcal{E}_{S_h}[\phi]$ is close to $\mathcal{P}[\phi]$ and hence an estimate of $\mathcal{E}_{S_h}[\phi]$ gives an estimate of the true expectation $\mathcal{P}[\phi]$. Given a function ϕ , Thresholdout first checks if the difference between the average value of ϕ on the training set S_t (or $\mathcal{E}_{S_t}[\phi]$) and the average value of ϕ on the holdout set S_h (or $\mathcal{E}_{S_h}[\phi]$) is below a certain threshold $T + \eta$. Here, T is a fixed number such as 0.01 and η is a Laplace noise variable whose standard deviation needs to be chosen depending on the desired guarantees. If the difference is below the threshold, then the algorithm returns $\mathcal{E}_{S_h}[\phi]$. If the difference is above the threshold, then the algorithm returns $\mathcal{E}_{S_h}[\phi] + \xi$ for another Laplacian noise variable ξ . Each time the difference is above threshold the “overfitting” budget B is reduced by one. Once it is exhausted, Thresholdout stops answering queries. In Figure 1, we provide the pseudocode of Thresholdout.

We now state the formal generalization guarantees that the entire execution of Thresholdout enjoys. They are based on the privacy guarantees of the “Sparse Vector” technique given in Chapter 3 of Ref.¹² and the generalization properties of differential privacy. For pure differential privacy we rely on Theorem 4 and for (ϵ, δ) -differential privacy we use the bound in Ref.²¹

THEOREM 9. *Let $\beta, \tau > 0$ and $m \geq B > 0$. We set $T = 3\tau/4$ and $\sigma = \tau/(96 \ln(4m/\beta))$. Let \mathcal{S} denote a holdout dataset of size n drawn i.i.d. from a distribution \mathcal{P} and S_t be any additional dataset over \mathcal{X} . Consider an algorithm that is given access to S_t and adaptively chooses functions ϕ_1, \dots, ϕ_m while interacting with Thresholdout which is given datasets \mathcal{S}, S_t and values σ, B, T . For every $i \in [m]$, let \mathbf{a}_i denote the answer of Thresholdout on function $\phi_i: \mathcal{X} \rightarrow [0, 1]$. Further, for every $i \in [m]$, we define the counter of overfitting*

$$\mathbf{Z}_i \doteq |\{j \leq i : |\mathcal{P}[\phi_j] - \mathcal{E}_{S_t}[\phi_j]| > \tau/2\}|.$$

Then

$$\Pr[\exists i \in [m], \mathbf{Z}_i < B \ \& \ |\mathbf{a}_i - \mathcal{P}[\phi_i]| \geq \tau] \leq \beta,$$

whenever $n \geq n_0$ for

$$n_0 = O\left(\frac{\ln(m/\beta)}{\tau^2}\right) \cdot \min\{B, \sqrt{B \ln(\ln(m/\beta)/\tau)}\}.$$

Note that in the bound on n , the term $O\left(\frac{\ln(m/\beta)}{\tau^2}\right)$ is equal (up to a constant factor) to the number of samples that are necessary to answer m nonadaptively chosen queries with tolerance τ and confidence $1 - \beta$. Further, this bound allows m to be exponentially large in n as long as B grows subquadratically in n (that is, $B \leq n^{2-c}$ for a constant $c > 0$).

We remark that the same approach also works for the class of low sensitivity queries. In Ref.,¹⁰ we also give an incomparable version of this algorithm with guarantees that derive from description length arguments rather than from

differential privacy. The advantage of that variant is that its use is not limited to low sensitivity queries.

4.1. Illustrative experiments

We describe a simple experiment on synthetic data that illustrates the danger of reusing a standard holdout set and how this issue can be resolved by our reusable holdout. In our experiment the analyst wants to build a classifier via the following common strategy. First the analyst finds a set of single attributes that are correlated with the class label. Then the analyst aggregates the correlated variables into a single model of higher accuracy (e.g., using boosting or bagging methods). More formally, the analyst is given a d -dimensional labeled data set S of size $2n$ and splits it randomly into a training set S_t and a holdout set S_h of equal size. We denote an element of S by a tuple (x, y) where x is a d -dimensional vector and $y \in \{-1, 1\}$ is the corresponding class label. The analyst wishes to select variables to be included in her classifier. For various values of the number of variables to select k , she picks k variables with the largest absolute correlations with the label. However, she verifies the correlations (with the label) on the holdout set and uses only those

variables whose correlation agrees in sign with the correlation on the training set and both correlations are larger than some threshold in absolute value. She then creates a simple linear threshold classifier on the selected variables using only the signs of the correlations of the selected variables. A final test evaluates the classification accuracy of the classifier on both the training set and the holdout set.

In the experiments, we used an implementation of Thresholdout that differs somewhat from the algorithm we analyzed theoretically (given in Figure 1). Specifically, we set the parameters to be $T = 0.04$ and $\sigma = 0.01$. This is lower than the values necessary for the proof (and which are not intended for direct application) but suffices to prevent overfitting in our experiment. Second, we used Gaussian noise instead of Laplacian noise as it has stronger concentration properties (in many differential privacy applications similar theoretical guarantees hold for mechanisms based on Gaussian noise—although not for ours).

No correlation between labels and data. In our first experiment, each attribute is drawn independently from the normal distribution $N(0, 1)$ and we choose the class label $y \in \{-1, 1\}$ uniformly at random so that there is no correlation between the data point and its label. We chose $n = 10,000$, $d = 10,000$ and varied the number of selected variables k . In this scenario no classifier can achieve true accuracy better than 50%. Nevertheless, reusing a standard holdout results in reported accuracy of over 63% for $k = 500$ on both the training set and the holdout set (the standard deviation of the error is less than 0.5%). The average and standard deviation of results obtained from 100 independent executions of the experiment are plotted in Figure 2 which also includes the accuracy of the classifier on another fresh data set of size n drawn from the same distribution. We then executed the same algorithm with our reusable holdout. The algorithm Thresholdout was invoked with $T = 0.04$ and $\sigma = 0.01$ explaining why the accuracy of the classifier reported by Thresholdout is off by up to 0.04 whenever the accuracy on the holdout set is within 0.04 of the accuracy on the training set. Thresholdout prevents the algorithm from overfitting

Figure 1. The details of Thresholdout algorithm.

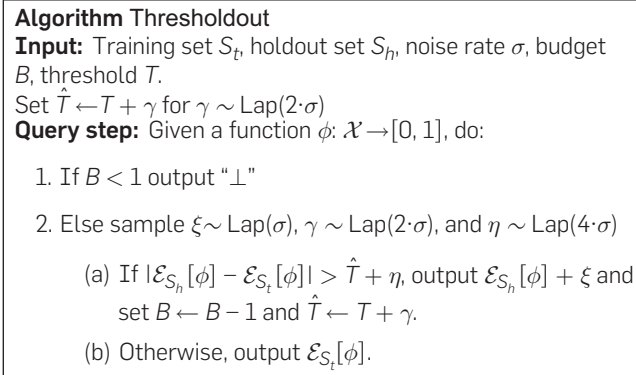


Figure 2. No correlation between class labels and data points. The plot shows the classification accuracy of the classifier on training, holdout, and fresh sets. Margins indicate the standard deviation.

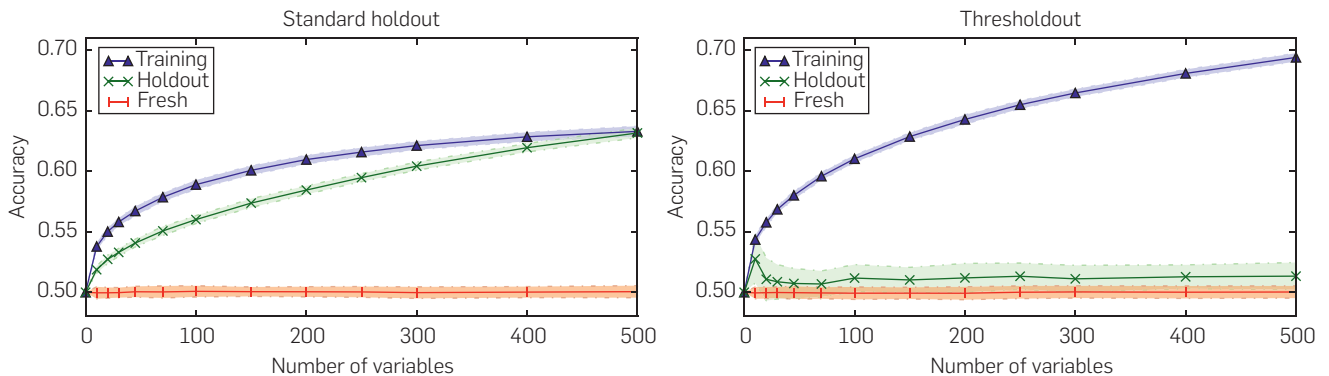
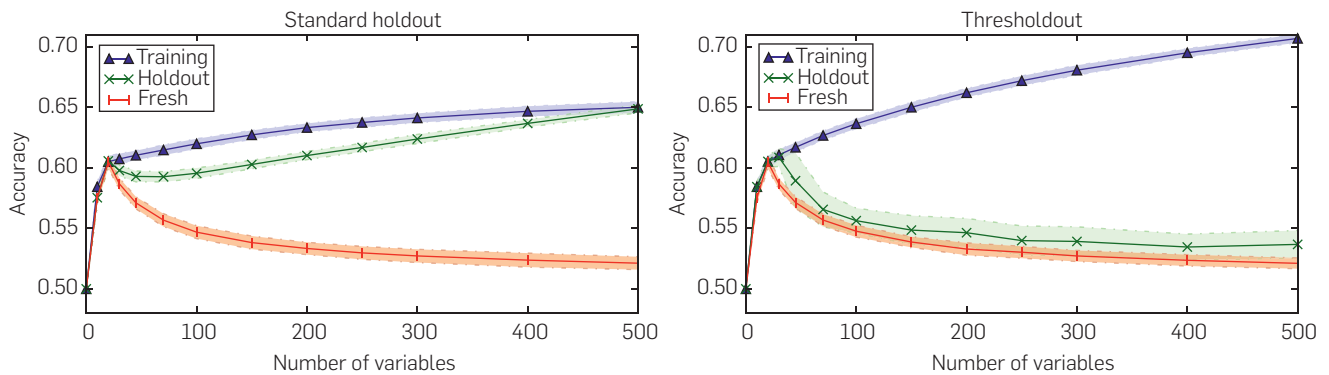


Figure 3. Some variables are correlated with the label.



to the holdout set and gives a valid estimate of classifier accuracy.

High correlation between labels and some of the variables. In our second experiment, the class labels are correlated with some of the variables. As before the label is randomly chosen from $\{-1, 1\}$ and each of the attributes is drawn from $N(0, 1)$ aside from 20 attributes which are drawn from $N(y \cdot 0.06, 1)$ where y is the class label. We execute the same algorithm on this data with both the standard holdout and Thresholdout and plot the results in Figure 3. Our experiment shows that when using the reusable holdout, the algorithm still finds a good classifier while preventing overfitting. This illustrates that the reusable holdout simultaneously prevents overfitting and allows for the discovery of true statistical patterns.

Acknowledgments

We would like to thank S. Arora, M.F. Balcan, A. Blum, D. Foster, M. Kearns, J. Kleinberg, A. Rakhlin, P. Rigollet, W. Su, and J. Ullman for the enlightening discussions about this work. We also thank the Simons Institute for Theoretical Computer Science at Berkeley where part of this research was done. This work was supported by the Alfred P. Sloan Foundation and NSF grant CNS 1253345.

References

1. Aschwanden, C. Science isn't broken.
2. Bassily, R., Nissim, K., Smith, A.D., Steinke, T., Stemmer, U., Ullman, J. Algorithmic stability for adaptive data analysis. In *STOC*, Cambridge, MA, USA (2016), 1046–1059.
3. Benjamini, Y., Hochberg, Y. Controlling the false discovery rate – A practical and powerful approach to multiple testing. *J. R. Stat. Soc. Ser. B* 57 (1995), 289–300.
4. Blum, A., Dwork, C., McSherry, F., Nissim, K. Practical privacy: The SuLQ framework. In *PODS*, Baltimore, Maryland, USA (2005), 128–138.
5. Blum, A., Hardt, M. The ladder: A reliable leaderboard for machine learning competitions. In *ICML*, Lille, France (2015), 1006–1014.
6. Bousquet, O., Elisseeff, A. Stability and generalization. *JMLR* 2 (2002), 499–526.
7. Cawley, G.C., Talbot, N.L.C. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.* 11 (2010), 2079–2107.
8. Dwork, C. A firm foundation for private data analysis. *CACM* 54, 1 (2011), 86–95.
9. Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., Roth, A. Preserving statistical validity in adaptive data analysis. *CoRR*, abs/1411.2664, 2014. Extended abstract in *STOC* 2015.
10. Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., Roth, A. Generalization in adaptive data analysis and holdout reuse. *CoRR*, abs/1506.2015. Extended abstract in *NIPS* 2015.
11. Dwork, C., McSherry, F., Nissim, K., Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC*, New York, NY, USA (2006), 265–284.
12. Dwork, C., Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.* 9, 34 (2014), 211–407.
13. Freedman, D.A. A note on screening regression equations. *Am. Statist.* 37, 2 (1983), 152–155.
14. Gelman, A., Loken, E. The statistical crisis in science. *Am. Statist.* 102, 6 (2014), 460.
15. Hardt, M., Rothblum, G. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, Las Vegas, Nevada, USA (2010), 61–70.
16. Hardt, M., Ullman, J. Preventing false discovery in interactive data analysis is hard. In *FOCS*, Philadelphia, PA, USA (2014), 454–463.
17. Hastie, T., Tibshirani, R., Friedman, J.H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer-Verlag, New York (2009).
18. Ioannidis, J.P.A. Why most published research findings are false. *PLoS Med.* 2, 8 (Aug. 2005), 124.
19. Kearns, M. Efficient noise-tolerant learning from statistical queries. *J. ACM* 45, 6 (1998), 983–1006.
20. Mukherjee, S., Niyogi, P., Poggio, T., Rifkin, R. Learning theory: Stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Adv. Comput. Math.* 25, 1–3 (2006), 161–193.
21. Nissim, K., Stemmer, U. On the generalization properties of differential privacy. *CoRR* (2015), abs/1504.05800.
22. Reunanen, J. Overfitting in making comparisons between variable selection methods. *J. Mach. Learn. Res.* 3 (2003), 1371–1382.
23. Shalev-Shwartz, S., Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 32 Avenue of the Americas, New York, NY 10013-2473, USA (2014).
24. Shalev-Shwartz, S., Shamir, O., Srebro, N., Sridharan, K. Learnability, stability and uniform convergence. *J. Mach. Learn. Res.* 11 (2010), 2635–2670.
25. Steinke, T., Ullman, J. Interactive fingerprinting codes and the hardness of preventing false discovery. In *COLT*, Paris, France (2015), 1588–1628.

Cynthia Dwork (dwork@microsoft.com), Microsoft Research, Mountain View, CA.

Vitaly Feldman (vitaly@post.harvard.edu), IBM Almaden Research Center, CA.

Moritz Hardt (m@mrtz.org), Google Research, Mountain View, CA.

Toniann Pitassi (toni@cs.toronto.edu), Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Omer Reingold (reingold@stanford.edu), Computer Science Department, Stanford University, Stanford, CA.

Aaron Roth (aaroth@cis.upenn.edu), Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

Copyright held by owners/authors. Publication rights licensed to ACM. \$15.00.

9th International Conference on
**Automotive User Interfaces and
Interactive Vehicular Applications**

September 24 - 27, 2017 in Oldenburg, Germany



AutomotiveUI, the International Conference on Automotive User Interfaces and Interactive Vehicular Applications, is the premier forum for UI research in the automotive domain. AutomotiveUI brings together researchers and practitioners interested in both the technical and the human aspects of in-vehicle user interfaces and applications. Consistent with prior conferences, AutomotiveUI'17 will address novel in-vehicle services, models of and concepts for enhancing the driver experience, driver performance and behavior, development of (semi-) autonomous driving, and the needs of different user groups. The overarching goal is to support the development of interfaces that are safe, easy to use, useful, and desired by users.

Important Dates

Full Paper & Notes	April 27
Workshops & Tutorials	June 2
Work-in-Progress	July 11
Interactive Demos	July 11
Doctoral Colloquium	July 11
Video Showcases	July 11
Industry Showcases	July 11

www.auto-ui.org/17

General Chair

Susanne Boll (University of Oldenburg, Germany)

Program Chair

Bastian Pfleging (University of Munich, Germany)

Birsen Donmez (University of Toronto, Canada)

Work-in-Progress Chair

Andreas Löcken (University of Oldenburg, Germany)

Ronald Schroeter (Queensland University of Technology, Australia)

Demo Chairs

Martin Baumann (University of Ulm, Germany)

Ignacio Alvarez (Intel Corporation, US)

Workshop & Tutorial Chairs

Lewis Chuang (Max Planck Institute for Biological Cybernetics Tübingen, Germany)

Sebastian Feuerstack (OFFIS - Institute for Information Technology, Germany)

Doctoral Colloquium

Andreas Riemer (University of Applied Sciences Ingolstadt, Germany)

Wendy Ju (Stanford University, US)

Video Chairs

Myounghoon "Philart" Jeon (Michigan Tech, US)

Hanneke Hooft van Huysduynen (Technische Universiteit Eindhoven, Netherlands)

Industry Showcase

Nora Broy (BMW Group, Germany)

Sebastian Osswald (Audi, Germany)



Association for
Computing Machinery



Southern Methodist University
Department of Computer Science and
Engineering
Open Rank Faculty Positions

The Computer Science and Engineering Department at Southern Methodist University invites applications for several faculty positions beginning Fall 2017. Individuals with experience and research interests in all areas of computing are encouraged to apply. Priority will be given to individuals with expertise in AI, data sciences, software systems, operating systems, networks, and the intersection of cybersecurity with social sciences. Candidates at all ranks will be considered. The successful candidates must have or expect to have a Ph.D. in Computer Science, Computer Engineering, or a closely related area by date of hire, and must demonstrate a strong commitment to excellence in teaching and research. In addition to the tenure or tenure-track appointment in the CSE Department, it is anticipated that successful candidates will also be engaged in some aspect of cybersecurity research at the Darwin Deason Institute for Cybersecurity at SMU.

The Dallas/Fort Worth area, one of the top three high-tech industrial centers in the country, has the largest concentration of telecommunications corporations in the US, providing abundant opportunities for industrial research cooperation and consulting. Dallas/Fort Worth is a multifaceted business and engineering community, offering exceptional museums, diverse cultural attractions, and a vibrant economy.

The CSE Department resides within the School of Engineering and offers BS, MS, and Ph.D. degrees in Computer Science and in Computer Engineering, BA in Computer Science, MS in Software Engineering, MS in Security Engineering, and D.Eng in Software Engineering. The department currently has 12 faculty members with research concentrations in cybersecurity, computer engineering, software engineering, computer arithmetic, algorithms, and related areas. Additional information may be found at: <http://www.smu.edu/Lyle/Departments/CSE/> and <http://www.smu.edu/Lyle/Institutes/DeasonInstitute>.

Interested individuals should send a complete resume and names of three references, including a one-page statement of research interests and accomplishments to:

Beth Minton, Administrative Assistant
Department of Computer Science and
Engineering
SMU
Dallas, TX 75275-0122

The committee will begin its review of the applications on March 31, 2017. To ensure full consideration, applications must be postmarked before March 31, 2017. However, the committee will continue to accept applications until all positions are filled. The committee will notify applicants of its employment decisions after the position is filled. Hiring is contingent upon the satisfactory completion of a background check.

SMU will not discriminate in any program or activity on the basis of race, color, religion, national origin, sex, age, disability, genetic information, veteran status, sexual orientation, or gender identity and expression. The Executive Director for Access and Equity/Title IX Coordinator is designated to handle inquiries regarding nondiscrimination policies and may be reached at the Perkins Administration Building, Room 204, 6425 Boaz Lane, Dallas, TX 75205, 214-768-3601, accessequity@smu.edu.

Position Numbers 53269 and 002205.



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**



香港中文大學
The Chinese University of Hong Kong

Applications are invited for:-

Faculty of Engineering Professors / Associate Professors / Assistant Professors (Ref. 170004N)

The Faculty of Engineering is seeking several faculty posts at Professor / Associate Professor / Assistant Professor levels with prospect for substantiation. The professors will play a significant role in the Cyber Security Center, which will be established by the Faculty of Engineering.

Cyber security is identified as one of the Faculty's strategic research areas, to be developed by both the Department of Computer Science & Engineering and Department of Information Engineering. Talented candidates are sought to complement existing efforts and create new synergies. Candidates in the following areas are encouraged to apply:

- cryptography and computational theory in security
- network, system and software security
- data security and privacy
- computer forensic
- hardware and IoT security

Applicants should have a relevant PhD degree and a good scholarly record demonstrating potential for teaching and research excellence.

Appointments will normally be made on contract basis for up to three years initially commencing August 2017, which, subject to performance and mutual agreement, may lead to longer-term appointment or substantiation later. The exact start date can be worked out with the successful applicants.

Applications will be accepted until the posts are filled.

Application Procedure

Applicants please upload the full resume with a cover letter, copies of academic credentials, publication list with abstracts of selected published papers, a research plan, a teaching statement, together with names and e-mails addresses of three to five referees to whom the applicant's consent has been given for their providing reference (unless otherwise specified).

The University only accepts and considers applications submitted online for the posts above. For more information and to apply online, please visit <http://career.cuhk.edu.hk>.



DOI:10.1145/3040969

Dennis Shasha

Upstart Puzzles

Stacking the Deck

CONSIDER 16 CARDS consisting of the ace through 8 of hearts and the ace through 8 of spades. You are allowed to arrange the cards as you wish. Your opponent chooses a number between 1 and 8. You deal that many cards from the top of the deck and put the last card face up, with a value of, say, k . You next deal k cards (ace is considered 1) and put the last card face up, with a value of, say, k' . You then deal k' cards and so on. You continue until the number revealed is more than the remaining cards, in which case your opponent wins or the last deal ends with the final card of the 16 and is an ace, in which case you win.

Warm-Up. Find an arrangement in which you can win this game.

Solution. There are many. Here is one possibility

3 4 5 6 7 8 7 6 5 4 3 2 1 2 8 1

If your opponent chooses, say, 2, you deal the first two cards, so the last card is a 4. Turning over the four cards after the 4 lands on an 8, then eight cards after

that lands on a 2. Turning over two more cards lands on the final ace. This will work no matter which number between 1 and 8 inclusive your opponent chooses.

Now suppose your opponent takes the following eight cards and arranges them like this

5 2 2 3 3 4 4 1

Can you insert the remaining cards among and perhaps before or after these eight and still guarantee to win?

Solution. Here is one solution, with the inserted cards bracketed

5 2 2 3 3 4 4 [1 7 6 5 6 7 8 8 1]

Consider the same problem, but your opponent starts, as in the Figure here, with

8 6 5 7 8 6 3 7

Solution.

8 6 5 7 8 6 3 [1] 7 [2 5 4 3 2 4 1]

Upstart 1. Suppose your opponent gets to arrange all cards $\geq d$. You are allowed to insert the remaining cards anywhere you like. Now find the minimum d that will still allow you to be sure to win and show how you did it.

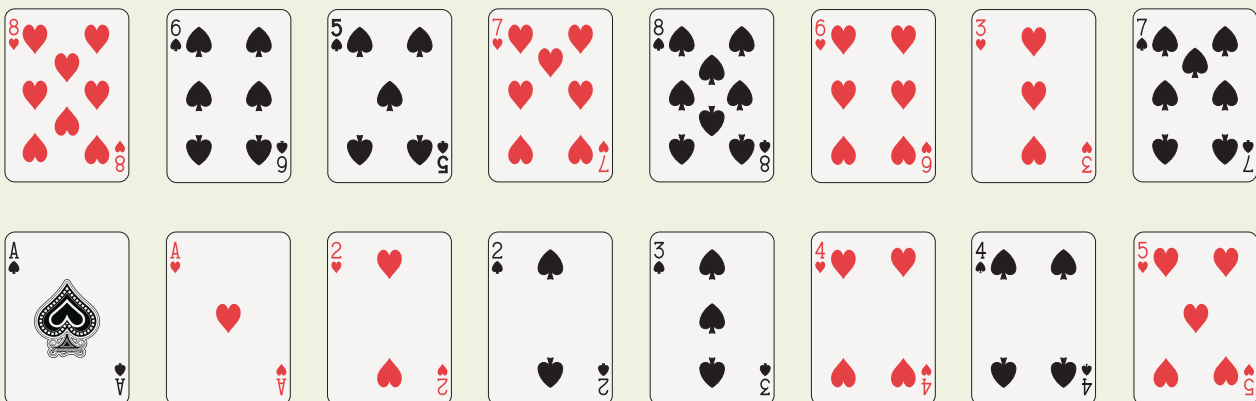
Upstart 2. Suppose you win if your opponent ever lands on an ace, whether of hearts or spades, no matter where it is. Now suppose your opponent gets to arrange all cards $\geq d$. You are allowed to insert the remaining cards anywhere you like. Find the minimum d that will still allow you to be sure to win and show how you did it.

All are invited to submit their solutions to upstartpuzzles@cacm.acm.org; solutions to upstarts and discussion will be posted at <http://cs.nyu.edu/cs/faculty/shasha/papers/cacmpuzzles.html>

Dennis Shasha (dennisshasha@yahoo.com) is a professor of computer science in the Computer Science Department of the Courant Institute at New York University, New York, as well as the chronicler of his good friend the omniheurist Dr. Ecco.

Copyright held by the author.

Consider a sequence of cards in the order 8 6 5 7 8 6 3 7 of, say, hearts and spades and a collection of hearts and spades 1 1 2 2 3 4 4 5. Can you intersperse the second set of cards in some order into the first sequence to force your opponent to land on the final card, which will be the ace of spades, based on the rules outlined here?



f @SpatialUserInteraction

🐦 @acm_sui



Deadlines

Papers - 30th of July, 2017

Posters - 11th of August, 2017

Demos - 11th of August, 2017

Visit sui2017.org to find out more

Spatial User Interaction (SUI 2017)

ACM SUI is the premier venue for presenting research in the design and use of technologies that focus on the challenges that users face when interacting with the volumetric, 3D space we live in.



Brighton, United Kingdom

The city of Brighton is nestled between the sea and the countryside providing a wealth of things to do.



Adalberto L. Simeone
General Chair



Kyle Johnsen
Program Chair



Rob Teather
Program Chair



Christian Sandor
Program Chair

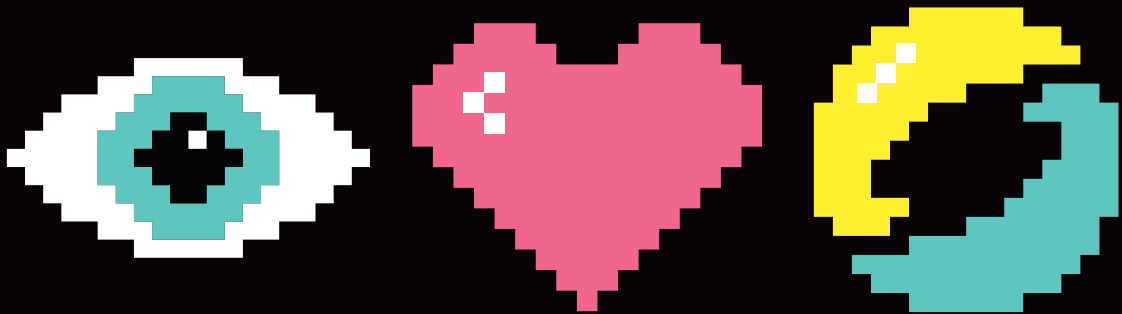
♥ Graciously Sponsored By





SIGGRAPH2017

AT THE  *of* **COMPUTER GRAPHICS &** **INTERACTIVE TECHNIQUES**



REGISTER TODAY *and* **SAVE**

30 JULY – 3 AUGUST

Los Angeles, California

[S2017.SIGGRAPH.ORG/REGISTRATION](https://s2017.siggraph.org/registration)



Sponsored by ACM SIGGRAPH
