

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

02/2019 VOL.62 NO.02

A New Golden Age for Computer Architecture

Agriculture Technology

Monitoring Noise Pollution

The Computational Sprinting Game

Blockchain from a Distributed Computing Perspective



Call for Nominations Editor-in-Chief ACM Books

The ACM Publications Board is seeking an Editor-in-Chief for ACM Books (<http://books.acm.org>).

Established in 2012 as a series of high-quality books for the computer science community, the ACM Books program now lists approximately 25 published titles with a similar number in preparation.

This EiC position is responsible for the editorial management of the Books series, consistent with general ACM policies.

The Publications Board relies on the Books EiC to ensure content maintains its exceptional quality and that the editorial process is both timely and fair.

The EiC will work with the in-house ACM editor to develop an editorial board and appoint associate editors.

Nominations are invited for a three-year term as ACM Books Editor-in-Chief, beginning on 1st May 2019.

The EiC appointment may be renewed at most one time.

This is an entirely voluntary position, but ACM will provide appropriate administrative support.

Nominations should include a brief statement as to why the nominee should be considered and a short statement on the candidate's vision for the future development of ACM Books. Self-nominations are most welcomed.

Please send all nominations to
Ron Perrot r.perrott@gmail.com or
Achi Dosanjh achi.dosanjh@hq.acm.org.

The ACM Publications Board has established a nominating committee to assist in selecting the next EiC.

Nominating committee members are:

Ron Perrot,
University of Oxford, UK, (*Chair*);

David Abramson,
University of Queensland, Australia;

Tiziana Catarci,
University of Rome, Italy;

Mathai Joseph,
Maharashtra, India;

Y Annie Liu,
Stony Brook University, USA;

Thomas J. Misa,
University of Minnesota, USA;

Chris Hankin,
ACM Publications Board Liaison,

Divesh Srivastava,
ACM Publications Board Liaison



Association for
Computing Machinery



BOOKS

Communications of the ACM **Europe Region Special Section**

A collection of articles spotlighting many of the leading-edge industry, academic, and government initiatives under way throughout Europe is coming to *Communications* this spring.

Articles will be authored by many of the region's leading computing professionals, highlighting exciting advances in technologies, diversity, and educational directives.

Among the topics to be explored:

- Web Science: Constructive, Analytics, Truly Social
- The European Perspective on Responsible Computing
- Information for All—A European Initiative
- Connected Things (Connecting Europe)
- Women in STEM in Europe
- EuroHPC

Plus the latest news about Europe's ICT agenda, well-connected consumers, HiPEAC network, enterprises that lead ICT innovation, and much more.



Association for
Computing Machinery



Departments

- 5 **Cerf's Up**
Libraries Considered Hazardous
By Vinton G. Cerf
-
- 6 **Letters to the Editor**
Between the Lines in the China Region Special Section
-
- 8 **BLOG@CACM**
Seeking Digital Humanities, IT Tech Support
Herbert Bruderer explains why the opposite of digital is *not* analog; Robin K. Hill describes how the challenges of user support are aggravated by indeterminate client responsibility.
-
- 31 **Calendar**
-
- 117 **Careers**

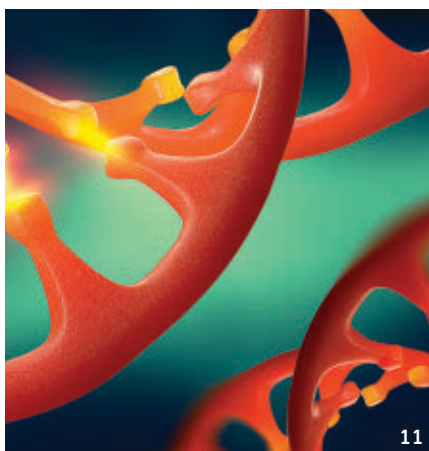
Last Byte

- 120 **Future Tense**
Hawking's Nightmare
By David Allen Batchelor



About the Cover:
John L. Hennessy and David A. Patterson's Turing Lecture (p. 48) traces computing architecture from the 1960s to present day and presents their projections for the field's next "Golden Age" in the coming decade. Cover illustration by Peter Crowther Associates.

News

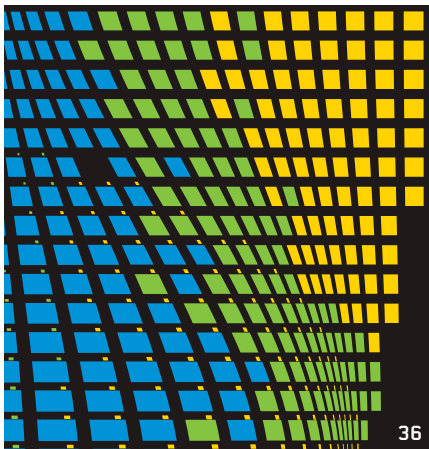


- 11 **A Brave New World of Genetic Engineering**
Genetic engineering technologies are advancing at a furious rate, changing the world one cell at a time.
By Samuel Greengard
-
- 14 **Technologizing Agriculture**
An array of technologies are making farms more efficient, safer, and profitable.
By Keith Kirkpatrick
-
- 17 **Being Recognized Everywhere**
How facial and voice recognition are reshaping society.
By Logan Kugler

Viewpoints

- 20 **Privacy and Security**
2018: A Big Year for Privacy
Retracing the pivotal privacy and security-related events and ensuing issues from the past year.
By Carl Landwehr
-
- 23 **Broadening Participation**
How Computer Science at CMU Is Attracting and Retaining Women
Carnegie Mellon University's successful efforts enrolling, sustaining, and graduating women in computer science challenge the belief in a gender divide in CS education.
By Carol Frieze and Jeria L. Quesenberry
-
- 27 **Code Vicious**
Writing a Test Plan
Establish your hypotheses, methodologies, and expected results.
By George V. Neville-Neil
-
- 28 **Viewpoint**
Tony's Law
Seeking to promote regulations for reliable software for the long-term prosperity of the software industry.
By Dror G. Feitelson
-
- 32 **Viewpoint**
Do We Really Need Computational Thinking?
Considering the expression "computational thinking" as an entry point to understand why the fundamental contribution of computing to science is the shift from solving problems to having problems solved.
By Enrico Nardelli

Practice



- 36 **CodeFlow: Improving the Code Review Process at Microsoft**
A discussion with Jacek Czerwonka, Michaela Greiler, Christian Bird, Lucas Panjer, and Terry Coatta
- 45 **The Importance of a Great Finish**
You have to finish strong, every time.
By Kate Matsudaira

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

Contributed Articles

- 48 **Turing Lecture**
A New Golden Age for Computer Architecture
Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.
By John L. Hennessy and David A. Patterson



To watch Hennessy and Patterson's full Turing Lecture, see <https://www.acm.org/hennessy-patterson-turing-lecture>

- 61 **Even Central Users Do Not Always Drive Information Diffusion**
Diffusion speed and scale depend on all kinds of information, not just which users have the most or fewest connections.
By Chao Gao, Zhen Su, Jiming Liu, and Jürgen Kurths

- 68 **SONYC: A System for Monitoring, Analyzing, and Mitigating Urban Noise Pollution**
SONYC integrates sensors, machine listening, data analytics, and citizen science to address noise pollution in New York City.
By Juan P. Bello, Claudio Silva, Oded Nov, R. Luke Dubois, Anish Arora, Justin Salamon, Charles Mydlarz, and Harish Doraiswamy



Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/sonyc>

Review Articles

- 78 **Blockchains from a Distributed Computing Perspective**
The roots of blockchain technologies are deeply interwoven in distributed computing.
By Maurice Herlihy



Watch the author discuss his work in the exclusive *Communications* video. <https://cacm.acm.org/videos/blockchains-from-a-distributed-computing-perspective>

- 86 **Separation Logic**
Separation logic is a key development in formal reasoning about programs, opening up new lines of attack on longstanding problems.
By Peter O'Hearn

Research Highlights

- 97 **Technical Perspective**
How Economic Theories Can Help Computers Beat the Heat
By Thomas F. Wenisch
- 98 **Distributed Strategies for Computational Sprints**
By Songchun Fan, Seyed Majid Zahedi, and Benjamin C. Lee
- 107 **Technical Perspective**
To Do or Not to Do: Extending SQL with Integer Linear Programming?
By Surajit Chaudhuri
- 108 **Scalable Computation of High-Order Optimization Queries**
By Matteo Brucato, Azza Abouzied, and Alexandra Meliou



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

Vicki L. Hanson

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

Cherri M. Pancake

Vice-President

Elizabeth Churchill

Secretary/Treasurer

Yannis Ioannidis

Past President

Alexander L. Wolf

Chair, SGB Board

Jeff Jortner

Co-Chairs, Publications Board

Jack Davidson and Joseph Konstan

Members-at-Large

Gabriele Anderst-Kotis; Susan Dumais;

Renée McCauley; Claudia Bauzer Medeiros;

Elizabeth D. Mynt; Pamela Samuelson;

Theo Schlossnagle; Eugene H. Spafford

SGB Council Representatives

Sarita Adve; Jeanna Neefe Matthews

BOARD CHAIRS

Education Board

Mehran Sahami and Jane Chu Prey

Practitioners Board

Terry Coatta

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Chris Hankin

ACM India Council

Abhiram Ranade

ACM China Council

Wenguan Chen

PUBLICATIONS BOARD

Co-Chairs

Jack Davidson; Joseph Konstan

Board Members

Phoebe Ayers; Edward A. Fox; Chris Hankin;

Xiang-Yang Li; Nenad Medvidovic;

Sue Moon; Michael L. Nelson;

Sharon Oviatt; Eugene H. Spafford;

Stephen N. Spencer; Divesh Srivastava;

Robert Walker; Julie R. Williamson

ACM U.S. Public Policy Office

Adam Eisgrau,

Director of Global Policy and Public Affairs

1701 Pennsylvania Ave NW, Suite 300,

Washington, DC 20006 USA

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

Computer Science Teachers Association

Jake Baskin

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

Lawrence M. Fisher

Web Editor

David Roman

Editorial Assistant

Danbi Yu

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquiot

Production Manager

Bernadette Shade

Intellectual Property Rights Coordinator

Barbara Ryan

Advertising Sales Account Manager

Ilia Rodriguez

Columnists

David Anderson; 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>

WEB BOARD

Chair

James Landay

Board Members

Marti Hearst; Jason I. Hong;

Jeff Johnson; Wendy E. MacKay

AUTHOR GUIDELINES

<http://cacm.acm.org/about-communications/author-center>

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

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

Andrew A. Chien

eic@cacm.acm.org

Deputy to the Editor-in-Chief

Lihan Chen

cacm.deputy.to.eic@gmail.com

SENIOR EDITOR

Moshe Y. Vardi

NEWS

Co-Chairs

Marc Snir and Alain Chesnaïs

Board Members

Monica Divitini; Mei Kobayashi;

Rajeev Rastogi; François Sillion

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;

John Leslie King; Paul Rosenbloom

Board Members

Michael L. Best; Judith Bishop; Andrew W. Cross;

James Grimmelmann; Mark Guzdial;

Haym B. Hirsch; Richard Ladner;

Carl Landwehr; Beng Chin Ooi;

Francesca Rossi; Loren Terveen;

Marshall Van Alstyne; Jeannette Wing;

Susan J. Winter

PRACTICE

Co-Chairs

Stephen Bourne and Theo Schlossnagle

Board Members

Eric Allman; Samy Bahra; Peter Bailis;

Betsy Beyer; Terry Coatta; Stuart Feldman;

Nicole Forsgren; Camille Fournier;

Jessie Frazelle; Benjamin Fried; Tom Killalea;

Tom Limoncelli; Kate Matsudaira;

Marshall Kirk McKusick; Erik Meijer;

George Neville-Niel; Jim Waldo;

Meredith Whittaker

CONTRIBUTED ARTICLES

Co-Chairs

James Larus and Gail Murphy

Board Members

William Aiello; Robert Austin; Kim Bruce;

Alan Bundy; Peter Buneman; Jeff Chase;

Carl Gutwin; Yannis Ioannidis;

Gal A. Kaminka; Ashish Kapoor;

Kristin Lauter; Igor Markov; Bernhard Nebel;

Lionel M. Ni; Adrian Perrig; Marie-Christine

Rousset; Krishan Sabnani; m.c. schraefel;

Ron Shamir; Alex Smola; Josep Torrellas;

Sebastian Uchitel; Hannes Werthner;

Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Shriram Krishnamurthi

Board Members

Martin Abadi; Amr El Abbadi; Sanjeev Arora;

Michael Backes; Maria-Florina Balcan;

David Brooks; Stuart K. Card; Jon Crowcroft;

Alexei Efros; Bryan Ford; Alon Halevy;

Gernot Heiser; Takeo Igarashi; Sven Koenig;

Greg Morrisett; Tim Roughgarden;

Guy Steele, Jr.; Robert Williamson;

Margaret H. Wright; Nikolai Zeldovich;

Andreas Zeller

SPECIAL SECTIONS

Co-Chairs

Sriram Rajamani and Jakob Rehof

Board Members

Tao Xie; Kenjiro Taura; David Padua

ACM Copyright Notice

Copyright © 2019 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 USA.



Association for
Computing Machinery





DOI:10.1145/3302508

Vinton G. Cerf

Libraries Considered Hazardous

Do you remember the story of the room full of immortal monkeys typing on typewriters forever? Eventually they would produce all works ever written and that would ever be

written. They would capture all truth but also everything that is false or only partly true. Were we to walk into such a place we would be confronted with an ultimate challenge: How to tell that which was true from everything else in this ultimate library?

In some ways, the contents of the Internet and especially the World Wide Web pose a similar challenge. About half the world's population is now online according to estimates by the International Telecommunication Union.^a These approximately 3.8 billion people produce enormous quantities of information on Web pages, in databases, in social media, and other online platforms. While I do not mean to suggest these Internauts are no better than monkeys typing at random, there *is* a great deal of misinformation mixed in with very high-quality content. Some of that misinformation is a consequence of ignorance, but some is deliberately produced disinformation intended to confuse or to bend public opinion to achieve questionable ends. Ironically, some of the best quality, highly endorsed information is also wrong, not out of malevolent intent, but because it has been invalidated by the scientific method: theory, experiment, and measurement leading to proof or refutation.

If we are honest with ourselves, science is, at best, an approximation of

reality. Even when they are not quite right, some theories can still be very useful. Newton's laws are useful for many computations but under conditions of acceleration, high-speed or intense gravity, one needs Einstein's refinements. And when we get to the ultra-small, we must move to quantum theory, but it doesn't account for gravity! The challenge for us is to know under what conditions the approximations are applicable.

How does all this apply to libraries? Libraries are organized accumulations of information. I almost wrote "knowledge" but that term seems to connote "truth" and we know now that all information is not true. As we accumulate more and more information, how can we curate this content so as to correctly distinguish truth from fiction? How do we cope with the discovery that what we thought was true is, in fact, false in the light of new information? Librarians have a role to play here as keepers of knowledge, but even they cannot be expected to be omniscient. What about digital content? What about online content? Can the curators of knowledge use online digital libraries to maintain and curate content, helping the users of the library to find truth and reject fiction (except, perhaps, when looking for entertainment)?

The task of curating the Internet's contents is well beyond any one person's ability, or even any particular group. If we are to curate this content, we will need widespread collaboration, some of it with automated tools

based on AI and machine learning. The libraries of the future cannot merely be catalogs of digital (and older media) content. The objects in the digital library will need to interact in some fashion so that truth value of their contents can be adjusted as new knowledge becomes available and is absorbed into the library. Such a process may actually prove feasible for factual knowledge but even there, *fact* can be elusive. Just as relativity theory shows us that two observers of the same two events may legitimately disagree as to the order in which these events occurred, it is not always clear what is factual and what is speculation.

All this tells us is that persistent accumulation of knowledge requires care and curation over time. One might even imagine that digital online libraries might have the ability to update themselves as new knowledge is added. John McCarthy^b once said to me, "Do you know, 100 years from now they will say, '100 years ago they had books that didn't talk to each other!'" It will be an enormous task to devise methods to accumulate and curate digital content and its relevant metadata including provenance and validity. Will computer, information, and library science be up to the task? We can but try. □

^b 1971 ACM A.M. Turing Award honoree.

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 <https://www.voanews.com/a/more-than-half-the-world-s-population-is-using-the-internet/4692926.html>

Between the Lines in the China Region Special Section

AS I READ the special section on the China Region (Nov. 2018), I thought privacy in China deserved better treatment than was expressed in the section's foreword "Welcome to the China Region Special Section" by co-organizers Wen-guang Chen and Xiang-Yang Li, that "People in China seem less sensitive about privacy." It sounded almost identical to what Robin Li, CEO and co-founder of Baidu, said in a talk at the March 2018 China Development Forum that was not well received by China's Internet users.²

A March 2018 survey of 100,000 Chinese households by CCTV and Tencent Research reported 76.3% of participants view AI as a threat to privacy.¹ Other global privacy surveys, including one by KPMG, reported privacy awareness in China as far more prevalent than the authors seemed to imply.

One of the few critical notes in the special section came near the end of the Elliott Zaagman's article "China's Computing Ambitions" when it called the lack of (Western-style) legal protections and transparency "a real concern." This was followed by a quote on the weaknesses of more-open digital societies. When lack of privacy rights was mentioned elsewhere in the special section, it was described as "an accepted observation."

Feng Chucheng of risk-analysis firm Blackpeak, said, "Rather than simply reflecting [the status quo] that privacy protections are not well-developed in this society, [Baidu] should be leading the charge to improve privacy rights."² Perhaps the professors and analysts who contributed articles to the section should have tried to do the same. It would not have detracted from the quality of their articles.

The "West" itself shows signs of moving toward being a surveillance society, and no amount of "privacy rights" will change that historical direction. More than a few Western governments are actually envious of China's unique ap-

plications of technology in society. We should be suspicious of government agencies and regulators redefining privacy or downgrading it or citing national security to make such applications fit their agenda. A similar observation can be made about privately run corporations as well, especially social networks.

Articles and columns in *Communications* should include, along with technological achievement, considerations on how they might be abused and the lessons that should be learned when they are. It would mean extra work for every author, as well as increased reader skepticism, but would surely increase awareness.

As a New Year's resolution, I respectfully invite everyone to read or reread the ACM Code of Ethics and Professional Conduct (<https://www.acm.org/code-of-ethics>), especially sections 1.1, 1.2, and 1.6, and incorporate it into their research and professional practice, especially those with authority and influence—or who publish in its leading publication.

References

1. Hersey, F. Almost 80% of Chinese concerned about AI threat to privacy, 32% already feel a threat to their work. *TechNode* (Mar. 2, 2018); <https://technode.com/2018/03/02/almost-80-chinese-concerned-ai-threat-privacy-32-already-feel-threat-work/>
2. Li, R. Are Chinese people 'less sensitive' about privacy? *Sixth Tone* (Mar. 27, 2018); <http://www.sixthtone.com/news/1001996/are-chinese-people-less-sensitive-about-privacy%3F>

Vincent Van Den Berghe,
Leuven, Belgium

Response from the Editor-in-Chief

Van Den Berghe's letter raises a good point—that articles discussing technology can and should be enriched by discussion of their societal context, including potential abuses. I am pleased to see this topic being raised in the context of the China Region special section and believe it applies much more broadly, both globally and across a variety of topics. This is an important challenge to Communications authors. I am sure they will rise to it.

Andrew A. Chien, Chicago, IL, USA

More to Learn

About Machine Learning

In their Viewpoint "Learning Machine Learning" (Dec. 2018), Ted G. Lewis and Peter J. Denning used a Q&A format to address machine learning and neural nets but, in my view, omitted two fundamental and important questions. The first is:

Q. Is machine learning the best way to get the most reliable and efficient solution to a problem?

A. Not generally.

To explain my answer, I need a definition of "machine learning." Machine learning is a machine collecting data while providing service and using the data to improve the speed or accuracy of the service. This is neither new nor unusual. For example, a search program can reorder its search list to move the most frequently requested items toward the top of the list. This improves performance until there is a major change in the probability of the items being requested. When this happens, performance may degrade until the machine "learns" the new probabilities. Suggestions offered by a search engine are also based on data collected while serving users; the search engine uses the data to "learn" what users are likely to ask.

When machine learning is used to "discover" an algorithm, it may find a local optimum, or an algorithm that is better than similar algorithms but very different from a much better one. A human who took the time to understand the situation might find that algorithm. Machine learning is often a lazy programmer's way to solve a problem. Using machine learning may save the programmer time but fail to find the best solution. Further, the trained network may fail unexpectedly when it encounters data radically different from its training set.

The second Q&A pair Lewis and Denning should have addressed concerns "neural networks":

Q. If developers have constructed (or simulated) a physical neural net-

work and trained it to have the behavior they want, is it possible to replace it with more conventional hardware and software with the same behavior?

A. Yes.

In other words, there is no problem that can be solved using neural nets that could not be solved using other more conventional hardware and programming languages. Some claim the neural net will be faster (or more efficient in some sense), an assertion that remains to be proved. Any performance advantage observed today can be attributed to the highly parallel specialized processors used to implement the nets. Better performance can often be obtained by programming the hardware directly.

David Lorge Parnas, Ottawa, Canada

Authors Respond:

Given the space, we would have answered Parnas's provocative questions much the same way he did. We would have added how difficult it is to beat the performance of neural networks on special-purpose hardware. We also cannot ignore AlphaGo, the machine that played against itself for several days with no outside information and became a grandmaster at Go. The previous IBM chess supercomputer was carefully designed by industrious programmers over many years. Speed to solution is a powerful motivator, even if the solution may not be understandable.

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

No Lack of Newcomer Authors at CS Conferences

Jordi Cabot et al. first outlined their hypothesis about lack of “newcomer” authors being accepted at computer science conferences in their Viewpoint “Are CS Conferences (Too) Closed Communities?” (Oct. 2018) and then, seeking data to evaluate it, succumbed to confirmation bias, unintentionally undermining their own hypothesis. Their stated objective of “opening up” computer science conferences may be a laudable social goal, but they presented no evidence that the technical quality of conferences would be enhanced by doing so. Moreover, they presented little, if any, compelling evidence that the claimed lack of newcomer submissions is due to any

reason beyond the standard criterion—technical merit of the papers.

Although the title of the Viewpoint referred specifically to computer science conferences, Cabot et al. pointed out that the database of papers they included in their survey was limited to the area of computer *software*. They should thus have limited any conclusions to conferences likewise devoted to computer software.

They defined newcomer papers as “...research papers where all authors are new to the conference; that is, none of the authors has ever published a paper of any kind in that same conference.” This brings up two problematic analytical issues. First, is newcomer status binary? That is, does publication of a single paper in a conference render a newcomer author (to use their phrase) a “member of the community?” Second, how different would their statistics have been if they had used a data-collection period different from the seven years on which they based their analysis? These questions went unanswered.

Moreover, they said, “...analysis suggests that newcomer paper submissions represent at least one-third of the total number of submissions” based on the data of one of the Viewpoint authors as a member of the program committee of four software conferences. We cannot ignore the potential correlation among the conferences where he was a committee member. It thus seems unreasonable to conclude the data suggests anything about the set of 65 conferences covered in the study survey. Further, their suggestion that at least one-third of submissions are from newcomer authors was weakened by their later conjecture that “some potential newcomers refrain from submitting in the first place,” saying, “[t]he overall presence of newcomers decreases over time.” This suggests that either newcomers are becoming “established members of the conference community” or the field itself is shrinking. The possibility of computer software research shrinking is unlikely.

It is thus not apparent there is a “problem” involving lack of newcomers submitting papers to computer science conferences or that Cabot et al.’s suggestions are supported by rel-

evant data and would contribute to the health of the field of computer science.

Paul B. Schneck, Bala Cynwyd, PA, USA

Authors Respond:

We agree there is no evidence that opening up conferences increases their technical quality, at least not right away, but believe it is still an important goal for the community and one that will prove beneficial in the long term. We also agree an extended data analysis would be beneficial to continue the discussion. We hope the column triggers it and generates replication studies and some pressure on conference managements to release additional (anonymized) data.

Jordi Cabot, Barcelona, Spain,

Javier Luis Cánovas Izquierdo,
Barcelona, Spain, and

Valerio Cosentino, Madrid, Spain

Home Monitoring for Parkinson's Patients Already . . .

Near the end of Leah Hoffman’s interview with Dina Katabi “Reaping the Benefits of a Diverse Background” (Oct. 2018), Katabi said, “I couldn’t tell you if . . . we should change the dose of her Parkinson’s medication.” In fact, the winner of the 2018 Human-Competitive Award at the ACM Genetic and Evolutionary Computation Conference in Kyoto, Japan (see <http://www.human-competitive.org/awards>) has already done just that.

The prize went to Stephen L. Smith, a senior lecturer in the Department of Electronics in the University of York, York, U.K., for a home-monitoring device for Parkinson’s dyskinesia (involuntary muscle movement).¹ ClearSky’s LID-Monitor, which includes novel signal processing developed through Cartesian genetic programming, reports the severity of shaking associated with the disease to the patient’s medical team, assisting in setting the correct dose of Levodopa.

Reference

1. Lones, M.A. et al. A new evolutionary algorithm-based home-monitoring device for Parkinson’s dyskinesia. *Journal of Medical Systems* 41, 11 (Nov. 2017), article 176; <http://doi.org/10.1007/s10916-017-0811-7>

W.B. Langdon, London, U.K.

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.

© 2019 ACM 0001-0782/19/02

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:10.1145/3297799

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

Seeking Digital Humanities, IT Tech Support

Herbert Bruderer explains why the opposite of digital is not analog; Robin K. Hill describes how the challenges of user support are aggravated by indeterminate client responsibility.



Herbert Bruderer There Are No Digital Humanities

[https://cacm.acm.org/blogs/blog-cacm/232969-there-](https://cacm.acm.org/blogs/blog-cacm/232969-there-are-no-digital-humanities/fulltext)

[are-no-digital-humanities/fulltext](https://cacm.acm.org/blogs/blog-cacm/232969-there-are-no-digital-humanities/fulltext)

November 26, 2018

Digitization and the digital revolution are quite confusing. Probably most people believe digital is something new. Many think the opposite of digital is analog or mechanical. However, the forerunners of electronic or digital journals and books are printed works. I would not call them analog. Historians sometimes speak of a pre-digital era. Even museum experts are surprised when historical mechanical calculating machines are described as digital. For them, digital and electronic are synonymous. A new field of the humanities is named digital humanities.

However, the equation digital = new, analog = old does not work. Digital is not an achievement of the 21st century. Even the antique Salamis counting board (4th

century BC) was digital. The abacus is regarded as the oldest digital calculating aid. The Romans also used digital bead frames. Similar devices are offered today at flea markets. Digital calculating machines appeared in the 17th century (inventions by Wilhelm Schickard, Blaise Pascal, Gottfried Wilhelm Leibniz). In 1614, the Scotsman John Napier invented digital Napier rods, used for multiplication and division. Since the middle of the 19th century, mechanical calculating machines have been mass-produced in France (Thomas Arithmometer, patent 1820). Charles Babbage's (unfinished) analytical engine (1834) and a similar machine of the Spanish engineer Leonardo Torres Quevedo (1920) were also digital, as were the widely used punch card machines (Herman Hollerith, 1890).

Digitalization is therefore nothing new. The first mathematical instrument was not an analog but a digital device, the abacus. Significant phases of digitalization began in the 1940s and 1950s with the advent of relay and vacuum tube

computers. The shift from mechanics to electronics, which began mainly in the 1970s, replaced analog slide rules and digital mechanical calculators with digital electronic computers. For many years, analog and digital electronic computers competed against each other.

In my opinion, the humanities are neither analog nor digital. They increasingly use digital resources. It would be better to speak of computer-aided or computer-assisted humanities. The pre-digital era must have been before the Greek abacus.



Robin K. Hill Tech User Responsibility

[https://cacm.acm.org/blogs/blog-cacm/231489-tech-](https://cacm.acm.org/blogs/blog-cacm/231489-tech-user-responsibility/fulltext)

[user-responsibility/fulltext](https://cacm.acm.org/blogs/blog-cacm/231489-tech-user-responsibility/fulltext)

September 30, 2018

Some years of experience with faculty assistance has led me to speculate that the well-known frustrations of IT user support hide even deeper problems. Many of us with such experience know the chronic difficulty suffered by both client and consultant in the support scenario. Each day promises, and delivers, repeated problems, trivial issues, and deep misunderstandings attendant on the use of applications and devices. Users ask the same questions, individually and severally, over and over, requesting help when what they really want is someone who will do it for them. In my own experience providing technical support to faculty and also to members of a volunteer civic organization, I

deal with well-educated and competent people. Whereas most clients are cooperative and grateful, some are brusque and demanding, some are apologetic and jocular, many are just not listening.

On the tech support side, malfeasance includes overexplanation, underexplanation, incorrect explanation, and impatience, all transgressions of which I have been guilty from time to time. Why is this all so difficult? As the perceived burdens of technology build up on users, cheerful cooperation gives way to weary resignation and then to foot-dragging resentment. And this against an activity that is for their own good! Users resist reading manuals, or even short instructions, let alone working through a checklist, though learning the fundamentals would help them immensely. I have offered the briefest possible explanations of the client-server environment (“where your programs run”), HTML URLs (“how to reach websites”), and cloud storage (“where your files are stored”), to no avail. Direct orders, such as “Read this” or “Practice this,” even to people who are sincerely motivated (no matter their intelligence, job satisfaction, rank, or personality), have no effect. I have gradually come to the unsettling belief that this is not just exasperating, but revealing. (We acknowledge without comment the obvious possibility that I, and my fellow user support professionals, are just lousy instructors or repellent individuals.)

On the happy assumption that the average reader thinks the philosophy of computer science deals with lofty issues, this may seem pedestrian. Yet a problem so perplexing and intractable is ripe for a bit of philosophy. We might learn something about education or training from its apparent failure in such cases and thereby something about intelligence. We might learn something about the acceptance of responsibility from its apparent failure in such cases and thereby something about ethical duty.

As we look more closely (at naive users, at technically competent users, and even at us experts when we are faced with new technology), we see a reluctance to learn definitions, commands, good practices, and workflow. The hapless user does not build the cognitive scaffolding necessary to organize the concepts, so does not grasp which feature is relevant to what; that context is then even farther out of reach for the

consultant. Subsequently we see attenuation of commitment, where follow-up tasks are put aside until a better time, the initial momentum fades away, and the skills necessary for effective participation decay. This leads to an adversarial stance, where frustration morphs into resentment. Whose fault is this?

Although there is plenty of research and commentary on the responsibility of the vendor, there appears to be no inquiry into the responsibility of the consumer with respect to technology selection, mastery, and use. Should there be? Let’s interrogate some analogies: We impose a minimal degree of responsibility on someone checking a book out of a library—he or she should return it. The reading of it may be a norm, not an obligation. We impose a high degree of responsibility for driving a car, because it can kill people. We expect some degree of responsibility in the use of natural resources, because the effects are broadly dispersed. In domestic finances and budgeting, we assume the agent eventually will achieve independence, making unaided decisions and taking appropriate actions, out of self-interest. It’s not clear that any of those inform our view of the products of technology. Indeed, the very idea that software and hardware users have any responsibility toward their technology appears to stand in direct conflict to pervasive expectations on their part, as expressed thus:

This is a nuisance.

My duties involve real things, whereas this is just management of those things, not what I signed up for. Record keeping and bean counting should not take time from the job.

This is clerical.

These tools are complex, sure, and they require skill, the kind of skill embodied in a good secretary, who can handle tedium, the quirks, and the exceptions. But I deliberately avoided that career.

This is supposed to be easy.

These products are supposed to magically improve my life—vocational, social, and intellectual—immediately and painlessly. (This attitude, of course, is cultivated by technology vendors and promoters.) Because the product is fabulous, and intended explicitly for me, the trouble must lie with IT.

There’s not much in those expectations that can be corrected by user support staff. So where does responsibility

lie? Garrath Williams’s treatment of the notion of responsibility¹ notes the emergence of that notion only in the last two or three centuries, a brevity consistent with the lack of scholarship on client responsibility (also raising the question whether there really is any such thing). He locates responsibility not in the person, but in the multifarious modern world. “What is central is the moral division of labor created by our institutional fabric. This scheme of cooperation delimits the normative demands upon each of us, by defining particular spheres of responsibility. Given the fluidity, plurality and disagreement associated with normative demands in modern societies, this limitation is crucial.”

If there is a limit on each sphere of responsibility, then there should be a boundary on user support. Right now, no one understands the proper extent of support; no limiting structure is defined for the benefit of user or support staff. To define such a limit is to grant support staff authority to demur. Unthinkable as it may seem, modern technological society needs to consider, define, and sanction a point at which consultants can say “no.” Better yet, they won’t need to, because everyone will understand the limits; everyone will know where user support ends and user responsibility begins. Everyone will know that the manual should be read (and should be written in the first place), and they will know from accepted and ingrained cultural mores rather than from simply being told so by pesky IT people.

But we can’t work that out here and now! In the best case, the tribulation of tech help is a temporary issue, reflecting workplace stress in the face of upheaval, similar to legal and safety compliance demands. The problem will resolve as society grasps tech more firmly; that, however, will take time. We wait for the emergence of norms of responsibility in this and other aspects of technology. ■

Reference

1. Williams, G. Responsibility as a Virtue. *Ethical Theory and Moral Practice*. 11:4, 455–470. DOI: 10.1007/s10677-008-9109-7.

Herbert Bruderer is a retired lecturer in didactics of computer science at ETH Zürich. **Robin K. Hill** is a lecturer in the Department of Computer Science and an affiliate of both the Department of Philosophy and Religious Studies and the Wyoming Institute for Humanities Research at the University of Wyoming.

13TH ACM INTERNATIONAL CONFERENCE ON DISTRIBUTED AND EVENT-BASED SYSTEMS

DEBS 2019

24 –28th June, 2019

Darmstadtium, Schloßgraben 1, 64283 Darmstadt, Germany

“A forum for academia and industry to discuss cutting-edge research in event-based computing related to Big Data, AI/ML, IoT, and Distributed Systems.”

<http://www.debs2019.org>

Topics:

- ◆ Artificial Intelligence and Real-Time Processes
- ◆ Information-Centric Networking
- ◆ Machine Learning and its Applicability
- ◆ Programmable hardware and its impact on efficient event processing
- ◆ Business Processes and Event Processing
- ◆ In-Network Processing in Distributed and Networked Systems
- ◆ Multimedia Analytics and Event-Based Systems
- ◆ Smart Contracts and Blockchains

Submission Dates:

- | | |
|---|---------------------------------------|
| ◆ Abstract submission | February 19th, 2019 |
| ◆ Research and industry paper submission | February 26th, 2019 |
| ◆ Tutorial submission | March 22nd, 2019 |
| ◆ Grand challenge solution submission | April 7th, 2019 |
| ◆ Author notification research and industry track | April 9th, 2019 |
| ◆ Doctoral symposium poster & demo submissions | April 22nd, 2019 |

General Chairs:

Boris Koldehofe
TU Darmstadt, Germany
Guido Salvaneschi
TU Darmstadt, Germany

Program Chairs:

Badrish Chandramouli
Microsoft Research
Leonardo Querzoni
Sapienza University of Rome

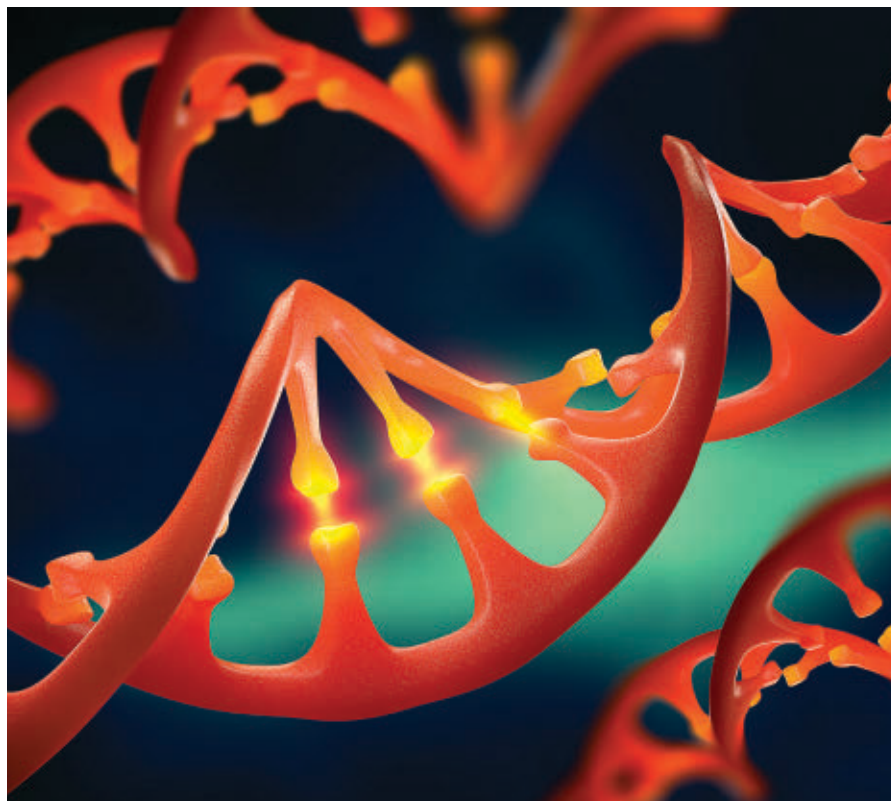


A Brave New World of Genetic Engineering

Genetic engineering technologies are advancing at a furious rate, changing the world one cell at a time.

ALTERING THE GENETIC code of plants and animals is not a job for the faint of heart. Nevertheless, in research labs around the world, scientists are increasingly peering into the cellular structures of living things—and recombining DNA and RNA molecules to produce everything from new tomatoes to new medicines. “The tools and technologies used for viewing and manipulating genetic materials have become more widely available and much easier to use,” observes George Church, a professor of genetics at Harvard Medical School and a pioneer in genomic research.

It is no small matter, even if the matter involved is at the molecular level. CRISPR, a powerful gene-editing toolkit, is advancing the field of programmable biology by leaps and bounds. It allows researchers to reconfigure genes and create new versions of things. Another technology, cryo-electron microscopy (Cryo-EM), is helping scientists peer into genetic material at a resolution that was once unimaginable. They can view the intricate structures of proteins, nucleic acids and other biomolecules, and even study how they move



and change as they perform various functions. Both of these tools, as well as more advanced computing models, have introduced a brave new world to genetic research.

“Although these two techniques are very different, they both are reshaping biology and genetic engineering,” states Eva Nogales, a professor in the Department of Molecular

and Cell Biology at the University of California, Berkeley, and senior faculty scientist at Lawrence Berkeley National Laboratory. “CRISPR and Cryo-EM allow researchers to perform an array of tasks faster and better.”

Adds Richard Henderson, research scientist at the Medical Research Council Laboratory of Molecular Biology in Cambridge, U.K., and a recipient of the 2017 Nobel Prize in Chemistry for his pioneering work on Cryo-EM, “We are at the cusp of remarkable advances in agriculture, medicine, and many other fields. These technologies will reshape science and the world.”

Cracking the Code on CRISPR

In only a few short years, the ability to reengineer the genetic structure of living things has moved from obscure research labs to the mainstream of science. CRISPR, which stands for Clustered Regularly Interspaced Short Palindromic Repeats, beckons with the promise of producing better tomatoes, insect-resistant grains, malaria-resistant mosquitos, and new types of pharmaceutical drugs to combat conditions ranging from sickle cell anemia and Alzheimer’s disease to cancer. Users can perform direct operations on genes by modifying and recombining molecular structures. “As the technology has advanced, the need to build everything from scratch in a lab has been replaced with commercially available products that produce effective results,” Church says.

Indeed, commercial firms with names like Synthego, Inscripta, and Twist Biosciences have developed kits that advance gene editing in much the way same way visual programming replaced the need to manually write endless lines of code for some software application. Although these firms take aim at the task through approaches that range from providing molecular resources to computational tools in software packages, the common denominator for the end-user is an ability to conduct research faster, more effectively, and at a lower cost. In fact, gene-editing tools that once had a price tag extending into the billions of dollars are now available for less than \$1,000. Essentially, “Any cell biology lab can use CRISPR,”

says Nogales, who visualizes CRISPR molecules using cryo-EM.

For example, Synthego, which Church is affiliated with, has introduced kits designed to address different gene editing tasks. Its \$1,495 Gene Knockout Kit (GKO) drops powerful capabilities into the hands of researchers. It taps predictive software and automation tools that help a researcher select a human gene to modify. It then applies a synthetic RNA gene to direct a protein to the specific location required for a DNA cut. The firm claims this toolkit has boosted the accuracy of CRISPR editing methods from around 50% to as much as 80%, or even more. The net result is an ability to cycle through variations of edited genes faster, speeding research and development for new procedures and drugs.

Paul Dabrowski, co-founder and CEO of Redwood City, CA-based Synthego, has said the firm’s gene editing system reduces the time it takes for a scientist to perform gene edits from several months to approximately one month. This, he has noted, helps researchers focus on results and outcomes, rather than the mechanics of an experiment.

Nogales says that while CRISPR tools fundamentally change the nature of research, they also present challenges. For one thing, because of uncertainty about errors caused by systems, CRISPR is not yet been approved for medical use by the U.S. Food and Drug Administration. For another, there is a learning curve associated with the technology. “Making a cut in the wrong place could be very deleterious. This is one of the reasons why CRISPR is used for agriculture more than human treatment,

“We are at the cusp of remarkable advances in agriculture, medicine, and many other fields. These technologies will reshape ... the world.”

but this will likely change over the coming years.”

Further advances in software and algorithms will drive smarter and better gene editing tools, Nogales adds. For instance, Inscripta, headquartered in Boulder, CO, has focused on developing a biological genetic engineering framework that resembles the all-in-one capabilities of a personal computer, while San Francisco-based Twist Biosciences is developing a system that places custom strands of synthetic DNA—the As, Ts, Cs, and Gs that serve as building blocks for biology—on semiconductor chips. This allows researchers to make up to a million CRISPR edits with a single chip, rather than using multiple systems and software to accomplish the task. The company’s self-described “smart algorithm” informs users within seconds whether the sequence they are testing can be synthesized.

Cryo-EM Enters the Picture

Although gene editing has introduced powerful capabilities into the research lab, scientists continue to struggle with understanding the mechanical functions of basic biological structures. From the invention of the microscope in the 13th century to more advanced forms of electron microscopy, improving resolution and reducing noise—particularly at extremely high levels of magnification—has proved vexing. “Obtaining clearer images is an ongoing challenge,” states Craig Yoshioka, research assistant professor and co-director of the Pacific Northwest Cryo-EM Center at Oregon Health Sciences University (OHSU) in Portland, OR.

For instance, one issue with cryo-electron microscopy is that bombarding a frozen sample with electrons can vaporize the specimen. As a result, Yoshioka says, scientists must essentially collect their images in “low light,” thereby reducing specimen damage, but also resulting in noisy data. The resulting noise makes it more difficult to view the behavior of the molecules and understand how they react to different conditions.

Meanwhile, another technique, called X-ray Crystallography, can produce a three-dimensional (3D) image of a molecular structure at high resolution by measuring how diffracted X-ray

beams scatter from crystallized molecules, but is difficult to apply to all samples. “It can be exceedingly difficult to get proteins to crystalize, sometimes nearly impossible,” Yoshioka explains.

Cryo-electron microscopy fundamentally changes the equation. Researchers place biological specimens under a transmission electron microscope and study them under cryogenic temperature conditions: -130°F or less. The system produces digital images that are run through specialized algorithms that dramatically reduce noise and sharpen the image using a method of frame alignment that studies particle behavior in different images. “The software processes the images and identifies the values of the key parameters,” says Henderson, who pioneered imaging techniques that, along with fellow 2017 Nobel Prize winners Jacques Dubochet and Joachim Frank, led to modern Cryo-EM.

Henderson says Cryo-EM addresses a basic problem with conventional electron microscopy: the interaction of electrons with organic matter causes a breakdown in their molecular structure, which generates a high level of visual noise. “It’s a bit like looking for a roe deer in a forest with dappled sunshine. It’s not easy to pick them out because they’re disguised,” he explains. To bypass the problem, Cryo-EM combines more advanced hardware with image-processing software that averages the position and behavior of thousands of individual particles and extrapolates the data to produce much clearer images of a biological structure. As a result, Cryo-EM can achieve atomic-level resolution models of complex, dynamic molecular assemblies.

Nobel laureate Frank, a professor of biochemistry and molecular biophysics at Columbia University in New York, says advances in graphics processing units (GPUs) and better algorithms have revolutionized the field. “Speed is no longer a problem, with the emergence of GPU software and clever algorithms,” he explains. Moreover, the field is continuing to advance and incorporate new computational methods. For example, “There are now software platforms ... that combine different packages under one umbrella and provide interoperability among

Conventional electron microscopy is “a bit like looking for a roe deer in a forest with dappled sunshine. It’s not easy to pick them out because they’re disguised.”

them so that the user has a choice about which operations to combine for best performance,” he explains.

The result, Yoshioka says, is a new era in understanding the mechanisms of life at a molecular scale. Although Cryo-EM microscopes can cost \$6 million or more, their use lets researchers visualize biological molecules at an atomic scale, and see them in their natural state. In contrast, X-ray crystallography requires scientists to order billions of molecules into well-ordered crystals, which can complicate the process of understanding of how they appear or function in a living cell. Using Cryo-EM, “We are better able to understand how proteins behave, how different substances or drugs affect them, and how modifications can change the way a drug binds to the protein,” Frank says.

Beyond Image

Genetic research is also leading biologists down the path of other computational methods that extend the boundaries of programmable biology. For instance, at the University of Groningen in the Netherlands, biotechnologists have used a modeling method to redesign the enzyme aspartase and convert it into a catalyst for asymmetric hydro-amination reactions that produce larger quantities of the substance. Working with researchers in China, the group was able to produce high volumes of extremely pure building blocks of aspartase that could be used in pharmaceuticals and other bioactive compounds.

Meanwhile, at the University of

Massachusetts, Amherst, researchers led by computational biophysicist Jihan Chen are developing sophisticated computer modeling algorithms and molecular simulation models that allow researchers to study a newly recognized class of substances called intrinsically disordered proteins (IDPs). These proteins contain highly flexible 3D structural properties that are extraordinarily difficult to observe. Chen’s technique relies on sheer computational power, because high-resolution imaging techniques like X-ray crystallography and nuclear magnetic resonance (NMR) cannot provide data about the highly flexible and fast-changing nature of these proteins.

Nogales believes these different genetic observation and engineering techniques will continue to break barriers and further advance science. “We are beginning to understand biology, chemistry, and physics at deeper and broader levels than ever before. We are now studying molecules that were almost unknown in the past, and we are putting the knowledge to work through gene editing tools such as CRISPR. We will see enormous changes in the biological world as a result of these techniques.”

Further Reading

Noble, C., Adlam, B., Church, G.M., Esvelt, K.M., and Nowak, M.A.

Current CRISPR Gene Drive Systems are Likely to be Highly Invasive in Wild Populations, *eLife*, e 2018;7:e33423. DOI: <https://doi.org/10.7554/eLife.33423>.

Cheng, Y., Glaeser, R.M., and Nogales, E. How Cryo-EM Became so Hot. *Benchmarks*. Volume 171, Issue 6, 30 November 2017, Pages 1229-1231. <https://doi.org/10.1016/j.cell.2017.11.016>.

Wright, V.W., Liu, J., Knott, G.J., Doxzen, K.W., Nogales, E., and Doudna, J.A. Structures of the CRISPR genome integration complex. *Science*, 20 Jul 2017: eaao0679. DOI: 10.1126/science.aao0679. <http://science.sciencemag.org/content/early/2017/07/19/science.aao0679>

Russo, C.J. and Henderson, R. Microscopic Charge Fluctuations Cause Minimal Contrast Loss in Cryo-EM, *Ultramicroscopy*, Volume 187, April 2018, Pages 56-63. <https://doi.org/10.1016/j.ultramic.2018.01.011>

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

© 2019 ACM 0001-0782/19/2 \$15.00

Technologizing Agriculture

An array of technologies are making farms more efficient, safer, and profitable.

AGRICULTURAL BUSINESSES usually have a massive number of trackable assets (plants, livestock, and machinery), often operate in wide geographic areas in which these assets are located, and are subject to operational factors often beyond their control, such as the amount of sunlight or rainfall they receive, or temperature fluctuations. As such, agriculture is ripe for the adoption of new technologies to help monitor and manage assets on a granular level, and everything from Internet of Things (IoT) sensors, robots, and drones are being used by farms around the globe.

The U.S. Department of Agriculture's National Institute of Food and Agriculture notes that the farms of today are avid users of agriculture technologies such as robots, temperature and moisture sensors, aerial imaging, and GPS technology, which are more precise and efficient than humans alone, and allow for safer, more efficient, and more profitable operations.

One example of how technology enables new farming techniques is the use of robotic harvesting on indoor farms, which today account for a tiny fraction of the 900 million acres of traditional farmland in the U.S. However, these indoor farms are well suited to the growth of vegetables such as tomatoes, lettuce, and other leafy greens, are highly sustainable, generally feature an average yield per acre more than 10 times higher than that of outdoor farms, and represent a continuation of the agricultural sector's trend toward incorporating precision agriculture techniques to improve yields and become more sustainable.

"Whether it's indoor or outdoor farmers, finding technologies that drive efficiencies is a big deal for [farmers]," says Josh Lessing, co-



A robotic tractor (left) cultivates a field alongside a tractor operated by a human, during a demonstration in Fukushima, Japan.

founder of Root AI, a company developing a robotic platform that allows the inspection, analysis, and harvesting of leafy vine plants grown indoors, such as tomatoes. "[A lot] of work has been done specifically in precision agriculture. 'How do I reduce the amount of herbicide; how do I reduce the amount of pesticide?'"

Lessing notes indoor agricultural practices expands a farm's margins, because less is spent on pesticides, since insects can be kept out of the greenhouse. Furthermore, reducing the use of chemicals can also limit the environmental impact of the operation.

Root AI's robot uses multiple cameras to collect color images and three-dimensional (3D) depth information on growing plants. One camera is located in the arm of the robot itself, while a secondary camera is affixed to

the side of the robot to provide a visual frame of reference.

The platform uses a customized convolutional neural network to detect objects of interest and label them with bounding boxes, which are used to train and build up the system's knowledge. While rolling between rows of plants, the camera captures the location of each fruit or vegetable, while also measuring properties such as ripeness, size, and quality grading. The data capture is done in real time, on the robot itself, without requiring access to a data center or the cloud.

The robot also uses a soft gripper, which looks like a pair of plastic salad tongs, that can pick a fruit or vegetable without damaging it. The idea is to allow the cultivation of these types of plants continuously and more effectively than humans can do, while aug-

menting the labor force.

“What we’re specializing in is being able to contact pieces of produce and harvest them without damage,” says Lessing. “The customers that I’m working with, one of the major problems that they’re running into is they can’t find enough labor to expand their operations. So we’re supplementing the labor force, and we’re delivering to growing managers intelligence to every piece of the operation.”

Another challenge faced by crop growers is weed control. Traditionally, farmers would spray herbicides broadly across their crops, which not only was wasteful, but also potentially harmful to humans, as crops were often overexposed to the chemicals. Companies such as EcoRobotix, and Blue River Technology with its See & Spray agricultural machines equipped with computer vision and machine learning capabilities, claim they can eliminate 90% of herbicide volumes typically used on farms today.

The presence of weeds is not the only enemy of crops (and farmers). Plant diseases, if they are not detected quickly, can spread rapidly, and even incremental changes in the soil’s composition can have a drastic impact on crop yields.

“In terms of precision agriculture, [farms] are using more connected sensors on the ground to test nitrogen levels, for instance,” says Nisarg Desai, director of product management, IoT, GlobalSign, a networking technology company that has worked with agriculture companies to implement IoT communication security technology for plant sensor networks. Desai says IoT sensors are used to test soil moisture levels to identify flooding, overwatering, or ground freezing; IoT-enabled water and fertilizer delivery valves can also be remotely monitored and managed.

Some farms are turning to drone technology, using unmanned aerial vehicles (UAVs) equipped with a package of high-definition cameras, IR sensors, and image-recognition capabilities to monitor crops, which can provide significant increases in efficiency. In a recent study, drone operator PrecisionHawk found that farmers who used drone-based aerial intelligence instead of taking plot-based crop measurements by hand were able to collect

“With the drone, you can go from visual data to multispectral data, to thermal data, to hyperspectral data, all in one flight.”

data 2.5 times more efficiently and 25% more accurately, and the collection itself was more objective, repeatable, and standardized.

Thomas Haun, senior vice president of partnerships with South Carolina-based commercial drone and data company Precision Hawk, says drones provide a significant advantage over not only traditional ground-based visual inspections, but also over satellite-based inspection. From a drone flying overhead, sensors can monitor a variety of conditions, including plant yields and growth information, as well as identifying indications of disease or insect/animal damage, and even tracking temperatures.

“With the drone, you can go from visual data to multispectral data (image data at specific frequencies), to thermal data, to hyperspectral data (from across the electromagnetic spectrum) all in one flight,” Haun says, noting that satellites are generally not equipped with sensing technologies that allow a very granular view of crops or plants. “We’re capturing data at sub-centimeter resolution. There’s an actual spatial resolution that our [sensors] are getting, providing a real advantage.”

Currently, most drone operators are limited by operational regulations, which limit drone flights to those that can be observed by the human operating the drone with his or her own eyes. This generally limits drone operations to about one square mile, according to Haun, though companies can apply for a Beyond Visual Line of Sight Waiver from the U.S. Federal Aviation Administration (FAA), although few such waivers are granted).

Precision Hawk has a waiver that allows the company to operate drones up to four miles away from an operator,

ACM Member News

AT THE INTERSECTION OF CS AND COMPUTATIONAL BIOLOGY



“I had an analog computer kit when I was a kid, where you turned a dial and it did

something,” says Dan Gusfield, Distinguished Professor of Computer Science at the University of California, Davis (UC Davis). Despite his enchantment with that early computer, Gusfield soon learned his real attraction was for discrete math, which offered a natural segue into computer science once he entered college.

Gusfield earned his undergraduate degree in computer science at the University of California at Berkeley, and his master’s degree in the same discipline from the University of California at Los Angeles. After receiving his Ph.D. in engineering science from the University of California at Berkeley, he spent six years as an assistant professor at Yale University, before moving to the University of California, Davis (UC Davis), where he has worked ever since.

He says his interests meet at the intersection of computer science and computational biology, an area on which he is writing his third book. Gusfield explains that the field of biology is becoming more quantitative, mathematical, and algorithmic, and these techniques are percolating down to biologists. In support of that, he is helping to establish an undergraduate major in quantitative biology at UC Davis.

With retirement on the horizon in a few years, Gusfield has no plans to end his career. He considers most of his academic work has been on computational techniques for problems that arise in biology, but he has never had the opportunity to take the next step and apply his techniques and programs to specific diseases.

“I want to delve more into a real disease,” he says.

—John Delaney

but the company expects that as time goes on and the FAA becomes more comfortable with drones, other commercial drone operators—and eventually farmers themselves—will be able to fly drones over a wider expanse beyond visual contact.

However, drones are hardly the only technology that allow agricultural asset monitoring and management. Using IoT-connected sensors placed near plants, farm operators can capture and record environmental conditions, and then send the data back to the farm's data center for analysis and action via a wireless connection. In some farming regions, there is a robust commercial wireless network (or networks) that can serve as a backbone for IoT-connected sensors, and agriculture companies simply need to purchase and place sensors throughout their fields. While they need a sophisticated data capture and analytics solution in order to leverage the data captured, farms like those in the U.S. are generally able to quickly cover their harvesting area and begin to yield real insights by capturing data from the sensors, and acting on those insights immediately.

However, in other parts of the world, particularly in developing regions such as South America, Africa, and parts of Asia, commercial wireless coverage is not ubiquitous, as in low-population areas where much of the farming and harvesting is done. Rajant Corp. of Malverne, PA, is a provider of wireless mesh networking technology that works with large agriculture companies in South America to provide the connectivity required to monitor the huge fields of sugar cane, soybean, corn, and other agriculture crops, which are often located in remote areas with no wireless coverage.

Through the use of Rajant's mobile mesh networking technology, intelligent nodes called BreadCrumbs can be spread out across the fields to capture a variety of attributes, including soil nutrient content, soil pH, and moisture levels that can be tracked in real time, and alerts can signal farmers when a correction is needed. Farmers can then make the necessary adjustments by adding chemicals, water, or nutrients during the prime growing season.

Moreover, the mesh networking technology can be integrated with

Livestock management also is changing, as managers monitor herds in real time with Internet-connected collars and tags.

smart tractors and harvesters, which can include automated functionality, allowing them to easily “carry” the network wherever they are working without requiring a pre-built networking infrastructure.

“The producers in Latin America, specifically in Brazil, do not have connectivity in the field,” says Joeval Martins, Rajant channel sales director for Latin America. Martins explains that Rajant's mobile wireless networking technology is a more affordable building out a fixed network that covers the entire acreage of a farm.

Food production is not limited to crops. The management of livestock is also changing, as managers of cows, hogs, and other animals seek to monitor their herds in real time, using Internet-connected collars and tags. Moocall's Calving Sensor was developed in 2014 by founder Niall Austin, who lost a heifer and her calf during a difficult birth. Noting that cows' tail movement often predicts the onset of calving, Austin and his partners launched the Calving Sensor, which clips on the tail of the cow.

“Based on the movement of the tail and the temperature reading, it detects when the cow is actually calving, and it sends an SMS to the farmer, who then immediately takes action,” says Ludovico Fassati, Head of IoT, Vodafone Americas, which provides the wireless infrastructure for the service. “In the past, the farmer needed to kind of sleep with the cow, but now, he can be there only when needed. It optimizes the farmer's time,” and can reduce the mortality rate of the calving process.

The sensor has sold more than 25,000 units, and the company also offers two other applications, including Breedmanager, a free herd management software app that displays the herd based on breeding status; and Mooheat, a collar worn by a bull, along with a RFID ear tag for each cow or heifer, that can provide detailed information such as exact times of standing heat, due dates, and in-calf notifications.

GlobalSign's Desai notes that this type of tagging can be used to keep track of livestock herds across a wide grazing area. “We have a customer who is providing a solution for automated cattle tracking across a large ranch,” Desai says. “What we came up with as a solution, for lack of better terms, is like a Fitbit for cows,” with tracking collars with signed and encrypted certificates affixed to cows in the herd, to ensure data cannot be stolen and used by malevolent third parties. Information on cattle movement, individually and collectively, is collected and analyzed to determine whether a specific animal or group of animals have been separated from the herd, or are ill or injured.

“The technology greatly reduces the amount of human intervention or human labor actually required to go and check on these animals,” Desai says, reducing labor costs and increasing efficiency and margins for the producer. **C**

Further Reading

TongKe, F.
Smart Agriculture Based on Cloud Computing and IOT,
Journal of Convergence Information Technology, January 2013,
<https://pdfs.semanticscholar.org/62ee/b701c40626811a1111ca5d1db37650f1ea0b.pdf>

Luciano, M.
Satisfying Three Necessary Components for BVLOS Flight,
Wireless Design Magazine, May 9, 2018,
<https://www.wirelessdesignmag.com/blog/2018/05/satisfying-three-necessary-components-bvlos-flight-operations>

Root AI-Reveal
August 8, 2018
<https://www.youtube.com/watch?v=c-JduOfLEpc>

Keith Kirkpatrick is principal of 4K Research & Consulting, LLC, based in Lynbrook, NY, USA.

© 2019 ACM 0001-0782/19/2 \$15.00

Being Recognized Everywhere

How facial and voice recognition are reshaping society.

THANKS TO ADVANCES in artificial intelligence (AI), society is now facing a unique challenge: how do we regulate the usage of human faces and voices?

Facial recognition is the ability of computer systems to identify and us by our faces. Voice recognition is the ability of computer systems to do the same for our words. Both are powered by AI, and both create benefits for consumers and citizens.

These technologies also raise difficult questions about privacy and personal rights.

Voice recognition powers popular consumer devices like Siri and Alexa, but it is also possible these devices are surreptitiously logging conversations and providing law enforcement with information on individuals.

Consider: Amazon sold 2.5 million of its Echo voice-assisted devices in the first quarter of 2018, according to GeekWire, while Google sold 3.2 million of its Google Home devices. Both devices represent one of the main ways that individuals are being listened to by machines and, in turn, by the makers of those machines.

Facial recognition can be used by law enforcement to identify criminals faster, but it is also used by the Chinese government for mass surveillance of its citizenry.

Facebook alone has more than two billion monthly active users, and any of them who post photos are subject to the firm's facial recognition algorithms, which identify and suggest tags to users. This is to say nothing of widespread video surveillance used by national governments to identify citizens. For instance, large-scale facial recognition will be used to identify and monitor hundreds of thousands of people during the 2020 Summer Olympics in Tokyo.

This all raises the question:



A Transportation Security Administration (TSA) screener uses a biometric facial recognition scanner on a traveler at Washington Dulles International Airport.

In an age where technology can recognize you everywhere, visually or audibly, how do you retain your privacy and personal agency?

"Digitization facilitates the tracking of everything we do online," says Eileen Donahoe, executive director of the Global Digital Policy Incubator at Stanford University's Center for Democracy, Development, and the Rule of Law. "If everything we do can be tracked and monitored by government, it will have a chilling effect on what we feel free to say, with whom we feel free to meet, and where we choose to go."

"This loss of privacy in digitized society goes to the heart of free expression, freedom of movement, freedom of assembly and association."

The Dangers of Facial Recognition

Facial recognition is, broadly, the ability of computer vision systems

to identify specific human faces in photos or video. This technology can identify and log facial details of individuals by using cloud infrastructure to process images from a computer, smartphone, or camera. This information then may be used for a range of purposes, from recommending someone to tag on Facebook to catching criminals.

For instance, Amazon has sold facial recognition technology to U.S. law enforcement, where it is used to identify persons of interest. It is also used for mundane functions like checking for identity theft and fraud at a Department of Motor Vehicles (DMV), says Clare Garvie, a facial recognition technology expert at Georgetown University.

Facial recognition also gives centralized authorities like governments and multinational firms the power to

identify people and to control them at scale.

“In China, the government aims to enroll all citizens into a facial recognition database, to facilitate ubiquitous tracking and identification,” says Garvie. “In Russia, face surveillance has been used to monitor and intimidate counter-government protests. In the United States, the Department of Homeland Security and some state and local jurisdictions are exploring the reaches of the technology as well.”

Facial recognition raises questions about a citizen’s right to privacy. The Electronic Frontier Foundation (EFF) published a whitepaper in which it posits that “face recognition disproportionately impacts people of color” by misidentifying African Americans and minorities at higher rates than whites.

This matters, given the ubiquity of facial recognition systems in modern public life.

“As one of the first viable technologies for conducting biometric surveillance, facial recognition funda-

mentally changes how we must view privacy and anonymity in public spaces,” says Garvie. “With access to the right databases, law enforcement may now be able to locate and identify any person walking by a security camera.”

If the data produced by facial recognition is flawed or biased, it is possible law enforcement and government bodies could risk infringing on the rights of citizens by using imperfect data to make legal or enforcement decisions.

“Facial recognition technology will take this loss of privacy and liberty to a new level by taking choice about utilization of the technology away from citizens,” says Donahoe.

“One of the most concerning dimensions of facial recognition technology is that it will be embedded in many different dimensions of daily existence without any choice among citizens, and without even the awareness of and choice about when they are being watched by government, it risks inverting the core democratic

concept that government is accountable to the people.”

Garvie observes that facial recognition is “not a monolith”; the technology is used by many different parties for many different purposes. It can enhance security and empower law enforcement, or it can be used to collect data on citizens.

“As a society, we must think very carefully not just about its benefits, but its risks, and use legislation to guard against the latter.”

Unfortunately, the laws governing the use of facial recognition technology are murky at best, according to *WIRED* magazine, which points out that “state and federal laws generally leave police departments free to do things like search video or images collected from public cameras for particular faces.”

Always Listening

Voice recognition is the ability of natural language processing (NLP) software to “understand” human language. A system like Siri or Alexa

Milestones

ACM Names 2018 Fellows

ACM has named 56 members as 2018 ACM Fellows for theoretical and specific achievements in computer architecture, mobile networks, robotics, and systems security, underpinning the technologies that define the digital age and have had significant ramifications in our lives.

Said ACM President Cherri M. Pancake, “We are honored to add a new class of Fellows to ACM’s ranks and we look forward to the guidance and counsel they will provide to our organization.”

THE NEW ACM FELLOWS ARE:

Gul Agha, *University of Illinois at Urbana-Champaign*
 Krste Asanovic, *University of California, Berkeley*
 N. Asokan, *Aalto University*
 Paul Barham, *Google Brain*
 Peter L. Bartlett, *University of California, Berkeley*
 David Basin, *ETH Zurich*
 Elizabeth M. Belding, *University of California, Santa Barbara*

Rastislav Bodik, *University of Washington*
 Katy Borner, *Indiana University*
 Amy S. Bruckman, *Georgia Institute of Technology*
 Jan Camenisch, *IBM Research/DFINITY Labs Zurich*
 Adnan Darwiche, *University of California, Los Angeles*
 Andre M. Dehon, *University of Pennsylvania*
 Premkumar T. Devanbu, *University of California, Davis*
 Tamal Dey, *Ohio State University*
 Sandhya Dwarkadas, *University of Rochester*
 Steven Feiner, *Columbia University*
 Tim Finin, *University of Maryland, Baltimore County*
 Thomas Funkhouser, *Princeton University, Google*
 Minos Garofalakis, *Athens Research Center, Technical University of Crete*
 Mario Gerla, *University of California, Los Angeles*
 Juan E. Gilbert, *University of Florida*
 Mohammad T. Hajiaghayi, *University of Maryland, College Park*

Dan Halperin, *Tel Aviv University*
 Johan Håstad, *KTH Royal Institute of Technology, Stockholm*
 Tian He, *University of Minnesota, Twin Cities*
 Wendi Beth Heinzelman, *University of Rochester*
 Aaron Hertzmann, *Adobe Research*
 Jessica K. Hodgins, *Carnegie Mellon University*
 John Hughes, *Chalmers University*
 Charles Lee Isbell, *Georgia Institute of Technology*
 Kimberly Keeton, *Hewlett Packard Laboratories*
 Sanjeev Khanna, *University of Pennsylvania*
 Lillian Lee, *Cornell University*
 Tom Leighton, *Akamai Technologies*
 Fei-Fei Li, *Stanford University*
 Michael Littman, *Brown University*
 Huan Liu, *Arizona State University*
 Jiebo Luo, *University of Rochester*
 Bruce M. Maggs, *Duke University*
 Bangalore S. Manjunath, *University of California, Santa Barbara*
 Vishal Misra, *Columbia University, Google*

Frank Mueller, *North Carolina State University*
 David Parkes, *Harvard University*
 Gurudatta Parulkar, *Open Networking Foundation (ONF)*
 Toniann Pitassi, *University of Toronto*
 Lili Qiu, *University of Texas at Austin*
 Matthew Roughan, *University of Adelaide*
 Amit Sahai, *University of California, Los Angeles*
 Alex Snoeren, *University of California, San Diego*
 Gerald Tesauro, *IBM Research, Yorktown*
 Bhavani Thuraisingham, *University of Texas at Dallas*
 Salil Vadhan, *Harvard University*
 Ellen M. Voorhees, *National Institute of Standards and Technology*
 Avi Wigderson, *Institute for Advanced Study*
 Alec Wolman, *Microsoft Research*

More information on the new ACM Fellows is available through the ACM Fellows site at <https://awards.acm.org/fellows>.

hears your voice, processes the language of your speech, then responds to the content of your queries.

We see the value of these systems every day. Voice assistants increasingly help us search online and find relevant content to serve consumer needs. In fact, about 50% of all online searches will be voice searches by 2020, according to media measurement and analytics firm Comscore.

“Voice recognition technology will expand accessibility to many devices and applications, especially for people who are visually impaired,” says Donahoe. “I can imagine voice recognition technologies bringing many beneficial applications and efficiencies to society, and expanding accessibility.” These could include making searching for information, and purchasing online, easier for consumers.

However, she notes, there are downsides.

While voice recognition may empower individuals, the technology also may impact privacy. Voice recognition devices are listening constantly, according to *The Washington Post*. These devices are listening for the “wake up” words that activate them, such as “Hey, Google” or “Alexa,” that users must speak to alert the devices that a request is about to be made. There have been allegations that these devices are always listening, and this information is then being logged in ways that violate user privacy.

Amazon has denied its voice-controlled Echo is always capturing what is said in its presence, saying, “that allegation—that the Echo is possibly recording at all times without the ‘wake word’ being issued—is incorrect,” according to an Amazon spokesperson. “The device is constantly listening but not recording, and nothing is streamed to or stored in the cloud without the wake word being detected.”

This has not stopped law enforcement from lobbying Amazon for user data when investigating potential crimes, in an effort to pull voice logs from the company’s servers. Amazon dropped a motion to protect audio recordings from one of its Echo devices that belonged to a murder suspect. The company had originally argued the data was protected by the First Amendment.

Amazon says the Echo is “constantly listening but not recording, and nothing is streamed to or stored in the cloud without the wake word being detected.”

No Easy Answers

Given the increasing ubiquity of facial and voice recognition, serious impacts on society are inevitable.

“I don’t think society is ready for the new potential of state power to track people,” says Martin Chorzempa, a research fellow at the Peterson Institute for International Economics in Washington, D.C. He cites the Chinese government’s use of facial recognition for law enforcement purposes to track down everyone from wanted criminals to jaywalkers.

“It will be increasingly difficult for individuals to avoid broadcasting to the world where they are,” Chorzempa says. “For example, someone who passes by Times Square on their way to work will likely show up in tourist photos that are posted on social media, and facial recognition could easily piece together their route to work and their schedule using the photos and the times or dates they were taken.”

In an era where devices are always watching and listening, personal privacy is more likely than ever to be assaulted by official institutions, even well-meaning democratic governments.

“We risk chilling free speech and assembly—rights guaranteed to us under the First Amendment,” says Garvie. “Law enforcement agencies themselves recognized this risk in a 2011 Privacy Impact Assessment, stating: ‘The potential harm of surveillance comes from its use as a tool of

social control. The mere possibility of surveillance has the potential to make people feel extremely uncomfortable, cause people to alter their behavior, and lead to self-censorship and inhibition.”

Donahoe is equally skeptical that governments will make the right call when it comes to balancing security and liberty. “The ease of use of facial recognition technology for security purposes will make it less likely that governments will protect citizen liberty to the extent required by democratic values,” she says.

“A core challenge for democratic governments will be continued adherence to the rule of law, where restrictions on individual liberty that flow from use of this technology must be justified by necessity, legitimate purpose, and use of the least restrictive means available.” ■

Further Reading

Simonite, T.

Few Rules Govern Police Use of Facial Recognition, WIRED, May 22, 2018, <https://www.wired.com/story/few-rules-govern-police-use-of-facial-recognition-technology/>

Lynch, J.

Face Off: Law Enforcement Use of Facial Recognition Technology, EFF, February 12, 2018, <https://www.eff.org/wp/law-enforcement-use-face-recognition>

Lapowsky, I.

Schools Can Now Get Facial Recognition Tech for Free. Should They? WIRED, July 17, 2018, <https://www.wired.com/story/realnetworks-facial-recognition-technology-schools/>

Levy, N.

Amazon Hands Over Alexa Data in Arkansas Hot Tub Murder Case, But 1st Amendment Questions Remain, GeekWire, March 7, 2017, <https://www.geekwire.com/2017/amazon-hands-over-alexa-data-in-arkansas-hot-tub-murder-case-but-questions-of-1st-amendment-rights-remain/>

Olson, C.

Just Say It: The Future of Search Is Voice and Personal Digital Assistants, Campaign, April 25, 2016, <https://www.campaignlive.co.uk/article/just-say-it-future-search-voice-personal-digital-assistants/1392459>

Logan Kugler is a freelance technology writer based in Tampa, FL, USA. He has written for over 60 major publications.

© 2019 ACM 0001-0782/19/2 \$15.00

Privacy and Security 2018: A Big Year for Privacy

Retracing the pivotal privacy and security-related events and ensuing issues from the past year.

THE YEAR 2018 may in the future be seen as a turning point for privacy incidents and associated privacy-policy concerns. In March, the Cambridge Analytica/Facebook incident opened many eyes to the unanticipated places personal data reaches, and it continues to generate repercussions.⁴ Google shut down its struggling Google Plus social networking system in October, after announcing it had exposed the data of approximately 500,000 users,¹⁵ only 1% as many as involved in the Cambridge Analytica case. Facebook revealed another data breach in October, this one affecting a reported 29 million users.¹⁴

The open GEDmatch genomics database, developed for genealogy research, was used by police and genetics experts to identify alleged murderers in two “cold cases” and several other crimes.⁸ The site’s founders, at first uncomfortable with its use by law enforcement, seem to now be more comfortable with it. Researchers subsequently

estimated that today approximately 60% of Americans of European descent could be identified from their DNA, even if they had never registered their DNA with any site.⁶ Further, they forecast the figure will rise to 90% in only two or three years.⁹

The John Hancock Life Insurance Company announced it would sell life insurance only through “interactive” policies that provide financial incentives to track policyholders’ fitness and health data through wearable devices and smartphones;² and the latest Apple Watch can take your electrocardiogram.

**Innovation has
its downside and
loss of privacy is
not easy to remedy.**

On the policy front, the long-awaited implementation of the EU’s General Data Protection Regulation (GDPR) in late May¹² triggered many reviews of corporate data privacy policies globally. These revisions required untold numbers of clicks by users asked to acknowledge policy changes.

About a month later, under threat from a strong privacy ballot initiative, California passed the California Consumer Privacy Act of 2018.¹ It incorporates some features of the GDPR and gives California consumers the right to know what personal information businesses have about them. Consumers control whom the information is shared with or sold to, and can request that information be deleted. This law begins to require consumer-facing businesses to live up to some of the Fair Information Practice Principles that were mandated for U.S. government systems (but not commercial enterprises) by the Privacy Act of 1974.¹³

“Personal information” in the California law is broadly defined. It includes biometric information, but



also “information that identifies, relates to, describes, is capable of being associated with, or could reasonably be linked, directly or indirectly, with a particular consumer or household.” The law enumerates almost a dozen categories of personal information, but exempts “publicly available” information (also defined in the law). Implementation details must be worked out before the law takes effect in 2020. The law has triggered national discussion and legislative proposals in other states.

Also in June, the U.S. Supreme Court handed down a decision in *Carpenter v. U.S.*³ This decision represents a notable limitation of the “third-party doctrine” wherein a government request to a third party to produce data an individual has voluntarily surrendered to it does not require a warrant. This doctrine, in place in the U.S. since 1979, is the basis for the idea that once a consumer surrenders data to a company as part of a transaction, the consumer loses any expectation of privacy for that data. As such, it has had major impli-

cations for, among other things, Internet-based transactions of all kinds.

The 5-4 decision had four separate dissenting opinions. The majority characterized the decision as “narrow” because it did not overturn the third party doctrine *per se*. Rather, it recognized the information in this case (cellphone site location information or CSLI records) deserves separate treatment because it is so invasive of “the privacies of life.” Further, Justice Gorsuch’s dissent argues for overturning the third-party doctrine. He proposes the consumer may well have a property interest in CSLI records held by the telephone company, although that argument was not put forth in this case. Other classes of data routinely collected by third parties could be equally invasive to the privacies of life; more litigation may follow.

In the fall, NIST initiated the development of a privacy framework.¹⁰ Like the cybersecurity framework it released in 2014 and updated in April 2018,¹¹ the privacy framework is not to be a standard, but a guide to common

privacy practices that will help companies assess privacy risk and adopt measures appropriate to the risk. In parallel, the NTIA, also part of the Department of Commerce, released a Request for Comments (RFC) on a two-part approach to consumer privacy: the first part describes desired user-centric privacy outcomes and the second sets high-level goals outlining an ecosystem to achieve those outcomes.⁵ The RFC proposes no changes to existing sectoral privacy laws, and, perhaps because it was developed in cooperation with the National Economic Council, the second part on high-level goals emphasizes maintaining “the flexibility to innovate” and proposes to employ a “risk and outcome-based” approach as opposed to one of compliance.

While no one loves red tape, innovation has its downside (remember those collateralized debt obligations?), and loss of privacy is not easy to remedy. Companies already have the option of building in “privacy by design,” but relatively few have done so. To me, a requirement

for some baseline of measures seems warranted, even essential.

And Congress, for the first time in years, is showing some interest in drafting comprehensive privacy legislation. This may become a hot topic for the 116th U.S. Congress if public interest continues to be strong.

Returning to the Facebook/Cambridge Analytica incident, this is of immediate importance to those in the computing profession, particularly those conducting research. A researcher with academic connections gained permission from Facebook to put up an app to collect data for research purposes in 2014. This app collected data from some Facebook users who consented to the collection, but also from millions of others without their knowledge or consent. This collection would now violate Facebook's policies, but it was not a violation at the time. The researcher provided this data to Cambridge Analytica, presumably in violation of Facebook's policies. Cambridge Analytica exploited the data for commercial purposes.

The primary issue here is accountability. This was either a violation of the academic's agreement with Facebook, or evidence that the agreements were insufficient to meet Facebook's 2011 consent decree with the Federal Trade Commission (FTC). The privacy of millions of people was violated and the reputation of legitimate academic researchers was tarnished. Facebook apparently had little incentive to hold the researcher and Cambridge Analytica to account. Aware of what happened over a year before the disclosure, Facebook belatedly issued yet another in a long history of privacy apologies.⁷

The FTC and the Securities and Exchange Commission (SEC) are investigating this incident. The SEC could find Facebook liable for failing to inform its shareholders of the incident when discovered. The FTC could find Facebook violated the terms of their 2011 consent agreement by failing to protect their customers' data in accordance with the consent decree. A court could make Facebook pay fines large enough to give it sufficient incentive to enforce the correct privacy policies on researchers and

Congress, for the first time in years, is showing some interest in drafting comprehensive privacy legislation.

those commercial entities that use Facebook data. The U.K. has already levied a fine of £500,000, the largest its legislation allows, but this is unlikely to provide much incentive to a company whose 2017 net income was over \$15 billion. The GDPR permits penalties of up to 4% of global revenues, which for Facebook would be well over \$1 billion, but the incident occurred before the GDPR took effect. The threat of future fines should give Facebook incentive to prevent recurrence.

Fines levied by the FTC go into the U.S. Treasury. Facebook's users took the risks and are suffering the consequences. Should they be compensated? A penny or dime for each user whose privacy was violated might not be the answer. Perhaps more progress would come from financing investigative journalism or other controls, but might not be within the scope of actions regulatory agencies can take. Imagination might be required to help Facebook hold their clients to account in ways that compensate Facebook users.

Computing professionals involved in "big data" research should pay attention if they wish to gain access to datasets containing or derived from personal information. They must abide by agreements made with dataset providers and remember that exposing data improperly damages public trust in research. Accidental or intentional release of personal data provided for research purposes to anyone else, even if aggregated and anonymized⁸ attracts public attention. Researchers who

abuse data entrusted to them must expect to be held accountable.

Facebook/Cambridge Analytica was not the first example of abuse, nor will it be the last. The FTC's privacy protection is evidently not working very well. Maybe the time has come for comprehensive privacy legislation focused on aligning corporate incentives so their products provide the privacy people expect and deserve. The California law might be a step in this direction.

A society where individuals are willing to share data for social benefit must make individuals confident that shared data are unlikely to be abused and that abusers can be identified and made accountable. **C**

- a Research into the weaknesses of anonymization or de-identification schemes is needed to understand the limitations of these techniques. Like research that exposes security weaknesses in systems, it must respect the concerns of those whose data is being studied.

References

1. Assembly Bill 375, California Consumer Privacy Act of 2018; <https://bit.ly/2z68PC0>
2. Barlyn, S. Strap on the Fitbit: John Hancock to sell only interactive life insurance. Reuters (Sept. 19, 2018); <https://reut.rs/2DbAq84>
3. *Carpenter v. U.S.*, 16-402. Decided June 22, 2018; <https://bit.ly/2MdFKaE>
4. Confessore, N. Audit approved of Facebook policies, even after Cambridge Analytica leak. *The New York Times* (Apr. 19, 2018); <https://nyti.ms/2vBnFI>
5. Department of Commerce, NTIA, RIN 0660-XC043. Developing the administration's approach to consumer privacy. *Federal Register* 83,187 (Sept. 26, 2018); <https://bit.ly/2AErZP>
6. Erlich, Y. et al. Identity inference of genomic data using long-range familial searches. *Science* (Oct. 11, 2018); <https://bit.ly/2CadGTP>
7. Hempel, J. A short history of Facebook's privacy gaffes. *WIRED* (Mar. 30, 2018); <https://bit.ly/2GjTPVD>
8. Murphy, H. How an unlikely family history website transformed cold case investigations. *The New York Times* (Oct. 15, 2018); <https://nyti.ms/2EnGHhE>
9. Murphy, H. Most white Americans' DNA can be identified through genealogy databases. *The New York Times* (Oct. 11, 2018); <https://nyti.ms/2pRFhBX>
10. NIST Privacy Framework Fact Sheet, Sept. 2018; <https://bit.ly/2AcYZOH>
11. NIST Framework for Improving Critical Infrastructure Cybersecurity, Version 1.1 (Apr. 16, 2018); <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf>
12. Official Journal of the European Union. General Data Protection Regulation. 4.5.2016. (English version); <https://bit.ly/2s7bupy>
13. Public Law 93-579. Privacy Act of 1974. (Dec. 31, 1974); <https://bit.ly/2yKCboA>
14. Vengattil, M. and Paresch, D. Facebook now says data breach affected 29 million users, details impact. Reuters (Oct. 12, 2018); <https://reut.rs/2CGewZz>
15. Wasabayashi, D. Google Plus will be shut down after user information exposed. *The New York Times* (Oct. 8, 2018); <https://nyti.ms/20KoFtH>

Carl Landwehr (carl.landwehr@gmail.com) is Lead Research Scientist the Cyber Security Policy and Research Institute (CSPRI) at George Washington University in Washington, D.C., and Visiting McDewitt Professor of Computer Science at LeMoyne College in Syracuse, NY.

Copyright held by author.

► Richard Ladner, Column Editor

Broadening Participation

How Computer Science at CMU Is Attracting and Retaining Women

Carnegie Mellon University's successful efforts enrolling, sustaining, and graduating women in computer science challenge the belief in a gender divide in CS education.

THE PERSISTENT UNDERREPRESENTATION of women in computing has gained the attention of employers, educators, and researchers for many years. In spite of numerous studies, reports, and recommendations we have seen little change in the representation of women in computer science (CS)—consider that only 17.9% of bachelor's degrees in computer science were awarded to women in 2016 according to the annual Taulbee Survey.¹⁵ At Carnegie Mellon University (CMU) we do not believe the situation is an intractable problem.

By paying close attention to culture and environment, and taking a cultural approach rather than a gender difference approach, our efforts continue to pay off. The percentage of women enrolling and graduating in CS at CMU has exceeded national averages for many years (see the accompanying figure and table). Indeed, the school gained attention when 48% (of the total 166 students), 49+% women (of the total 205 students), and just shy of 50% when 105 women (out of 211 students) entered the CS major in 2016, 2017, and 2018 respectively.^a But CMU is not alone—other institutions have also had



Women comprised more than 48% of incoming first-year undergraduate students at Carnegie Mellon University's School of Computer Science in fall 2016, establishing a new school benchmark for diversity.

success in addressing the gender gap. Harvey Mudd College, for example, went from 10% women in CS in 2006, the year Maria Klawe took over as college president, to 40% women in CS by 2012.² These institutions, and the many others who are investing in change to improve gender balance, are proof that—as CMU CS Professor Lenore Blum says—“it’s not rocket science!”

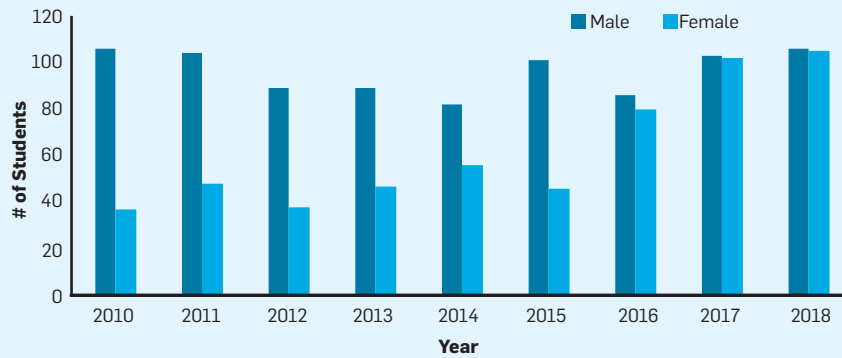
This column summarizes CMU's successful efforts in enrolling, sus-

taining, and graduating women in CS. Since 2002 we have conducted ongoing case studies to understand the CMU story.^b We have learned many valuable lessons. In a nutshell, for women to be

^b Case studies were conducted in 2002, 2004, 2009–2010, 2011–2012, and 2016–2017 and included a variety of data-collection tools including face-to-face interviews, surveys, focus groups, and observations. Participants included current undergraduate and graduate students, faculty, and staff.

^a See <https://bit.ly/2ULGgBS>

Percentage of male and female first-year students by year of enrollment.



First-year enrollment by gender (rounded to the nearest full number).

Year	Enrolled	# Male	% Male	# Female	% Female
2010	143	106	74%	37	26%
2011	152	104	68%	48	32%
2012	127	89	70%	38	30%
2013	136	89	65%	47	35%
2014	138	82	59%	56	41%
2015	147	101	69%	46	31%
2016	166	86	52%	80	48%
2017	205	103	50%	102	50%
2018	211	106	50%	105	50%

successful in CS we needed to change the culture and environment, and develop and sustain programs that work to level the playing field without making women feel like a separate species. However, we did not need to change the curriculum to be “pink” in any way. Indeed, gender difference approaches, which tend to assume CS should be changed to suit women’s presumed interests, have not provided satisfactory explanations for the low participation of women in CS. Indeed, beliefs in a gender divide may actually be deterring women from seeing themselves in male-dominated fields.

We hope the CMU story can help challenge the gender divide in CS, show that women can master this field successfully, and inspire others to think more broadly about intellectual and academic expectations. We acknowledge that the CMU experience may not be fully generalizable. For example, CMU is a private institution that may not have some of the constraints state institutions have because of various laws and regulations. While recognizing the potentially limited generalizability of

our experiences, we summarize five key takeaways we believe may be replicated at other institutions where there is the motivation for change.

Women Do Not Need a Female-Friendly Curriculum

From 1999 onward some dramatic changes occurred at CMU, changes that contributed to a successful and much-improved undergraduate experience for students in the CS major. Most significantly these changes led from women feeling out of place and small in number to being well represented, being an integral part of the CS culture, contributing to the culture, and being successful in the field alongside their male peers. Indeed, men and women graduate at the same rate. This success occurred without compromises to academic integrity, without changing the curriculum to suit women, nor by accommodating what are perceived to be “women’s” learning styles and attitudes to CS. Changes to the CMU curriculum, as in any department committed to providing the best academic program possible are made for the

benefit of all students. CMU, with its School of Computer Science and the seven departments within the school, offers a wide variety of courses—some of which are applications focused—but the core CS curriculum and a wide variety of advanced courses have become increasingly theory driven and rigorous without impacting students’ retention and success.

Cultural Change Is Key—And It Can Change at the Micro Level

In 1999, CMU dropped the programming/CS background requirement from the admissions criteria and added leadership potential while keeping high SAT scores, particularly in math and science. Dropping this requirement was prompted by a valuable finding from the 1995–1999 research studies.¹¹ Various entry levels into the first-year courses were created for students with little to no background. Other major contributing factors included: CMU Dean Raj Reddy’s vision to produce leaders in the field that also brought institutional support for change; Lenore Blum joined the CS faculty bringing long-standing expertise and advocacy for women in science and math; and the development of Women@SCS, an organization of faculty and students (mostly, but not all, women) led by a Student Advisory Committee, working to ensure that the professional experiences and social opportunities for women reflect the *implicit* opportunities for those in the majority (see <https://www.women.cs.cmu.edu/>).

These changes brought in many more women, and more students—both male and female—with a broader range of characteristics and interests. We started to see a more balanced student body, balanced in terms of gender, of student characteristics, and balanced in terms of leveling-the-playing-field opportunities for women through Women@SCS. In this more balanced environment our observations and series of studies, including our 2016–2017 study,^{3–7} found CMU students relating to CS through a spectrum of attitudes along with many more similarities than differences. For example, we found most students (men and women) have a deep interest in computer science and want to

do something useful with their skills in order to contribute to the social good.

Institutional Support Is Critical

We believe sustained student leadership, with women at the helm, has been critical for building a more inclusive community at CMU, and for enhancing the academic and social life of the entire community. At the same time, cultural change requires serious institutional support and cannot be left to chance, especially in a stubbornly male-dominated field like CS.

At CMU, we have found that institutional investment, providing funding, guidance, and endorsement for programs developed through Women@SCS, has paid off. The organization has become a valuable resource for everyone while strengthening the image of women in CS and challenging the stereotypes about who fits the field.

Cultural Factors Are More Important than Gender Differences

Gender difference approaches often argue that there are strong gender differences in the way girls and boys, or men and women, relate to the field; gender differences that work in favor of men and against women. To solve this problem and increase women's participation in CS it is suggested that we need to pay more attention to women's interests and attitudes and change CS accordingly. But approaches that recommend accommodating differences—without recognizing that such differences can change according to the culture and environment—risk perpetuating the gender divide.

This has not been our approach. Indeed, we questioned these assumptions and constraints. Gender is first and foremost a cultural issue not a women's issue, so rather than looking at "gender differences" as our working model we need to address the underlying culture in which attitudes and opportunities for equality are influenced and situated. This approach is supported by evidence from other cultures outside the U.S. Galpin describes the participation of women in undergraduate computing in more than 30 countries concluding "(t)he reasons that women choose to study computing will vary from culture to culture, and from country to country." Studies of women

We see culture as a dynamic process; shaping and being shaped by those who occupy it, in a synergistic diffusive process.

in computing in Mauritius and in Malaysia found no problem with women's participation concluding "the underrepresentation of women in CS is not a universal problem."⁹

But the gender difference mindset—epitomized by the bestseller *Men Are from Mars, Women Are from Venus*¹⁰—has a strong hold on public thinking in the U.S. and many parts of the Western world. For example, "... anonymous, aggregate data from Google searches suggests that contemporary American parents are far more likely to want their boys smart and their girls skinny."¹³ The belief that men are innately better at coding than women, is a case in point. This mindset, fed by stereotypes, is relentlessly perpetuated. In turn stereotypes feed our unconscious biases, which, if left unchecked, can often lead to negative consequences for women in computing, and ultimately for the field itself.

Cultural Interventions Are Needed for Change

We see culture as a dynamic process; shaping and being shaped by those who occupy it, in a synergistic diffusive process. A cultural approach examines a range of factors beyond gender as determinants of women's participation in CS including (but not limited to) the parts played by the K-12 curriculum, stereotype threat, opportunities for engagement in CS, opportunities for leadership, confidence levels, gender ratios, implicit bias, myths and stereotypes. A cultural approach examines these factors and develops actions and programs to intervene as

needed. Our latest intervention—Bias-Busters@CMU—developed in collaboration with CMU's College of Engineering and Google, works with the entire campus on the difficult issue of mitigating implicit bias.⁸

Interventions from Women@SCS have increased the visibility of women, placing them in leadership positions, providing opportunities for them to demonstrate their abilities, and to challenge stereotypes, all with the critical support of our deans, faculty and staff. For example, recognizing an often-familiar situation in which students can go through their entire school life without having a female instructor, Women@SCS developed a faculty-student lunch series, providing female students an opportunity to meet role models and have personal interactions in an informal setting. Most importantly Women@SCS has not been inward-looking. The organization has facilitated many outside the classroom programs for the benefit of the entire student body such as peer-to-peer interview and speaking skills workshops, outreach in the community, and peer-to-peer advice sessions. In 2014, Women@SCS was asked to take the lead on SCS4ALL—<http://www.scs4all.cs.cmu.edu/>—a student organization reaching out beyond gender. Women@SCS has shown that a women's organization can be much more than a "support" group for each other, they can be a valuable resource for building an inclusive community.

Conclusion

We have found that cultural change, not curriculum change (often recommended by gender-difference approaches), is the key to sustaining a community of women in CS. Indeed, we advise caution when making changes based on appealing to stereotypes—this may perpetuate the gender divide.

Institutional support is also critical for real change and ultimate success—this includes funding, guidance, and philosophical advocacy for leveling the playing field. CMU has not been afraid to give women a voice, to listen to women, and let women take the lead, enabling them to play a valuable role in changing the culture.

We suggest monitoring student attitudes toward, and experiences in, the

**The Seven Tools
of Causal Inference**

**With Reflections on
Machine Learning**

**Metamorphic Testing
of Driverless Cars**

**Beyond Worst-Case
Analysis**

**Telling Stories about Birds
From Telemetric Data**

**The Compositional
Architecture
of the Internet**

**From Computational
Thinking to
Computational Action**

**Benchmarking
‘Hello, World!’**

**Understanding Database
Reconstruction**

Attacks on Public Data

**Design Patterns
for Managing Up**

**A Hitchhiker’s Guide to
the Blockchain Universe**

**Predicting Program
Properties From Big Code**

Plus the latest news about
rare Earth, exoskeletons, and
advances in energy storage.

The persistent gender gap in computer science is well documented, but there is less sharing of success stories.

CS major. Are men and women getting similar opportunities for such things as leadership, visibility, networking, mentoring, and advocacy? Are women involved and given a central voice in shaping the culture?

While a good academic life is critical for success, students also need to feel like they belong socially¹⁴— this will enhance their sense of academic fit. Indeed college life is best viewed holistically. Do not underestimate the value of student organizations, and of social events where information is exchanged, friendships and communities are formed, and where everyone gets a chance to be included in the latest student discussions.

The persistent gender gap in CS is well documented, but there is less sharing of the success stories. By telling the CMU story we hope to illustrate a successful approach, one that can help the field of computing become more inclusive.^c At the same time, we cannot become complacent. Gender balance at the undergraduate level is not an end in itself and our efforts need to continue. Success with gender diversity is one important step in developing strategies to be more inclusive of all who are underrepresented in the field of computing. In doing so we believe the CMU approach, with a focus on culture is particularly advantageous because culture is mutable

^c We recognize that women and men are not single separate categories and yet we are as guilty as anyone for using the term “women” and “men.” We are all shaped by complex identities and experiences and a multitude of determinants are involved in our choosing or not choosing to study computer science.

and potentially open to the changes we seek. This means we aim to continue to pay close attention to the issue, provide institutional support, a willingness to act, and flexibility to enable change. The CMU approach recognizes that ultimately diversity and inclusion benefit the school, the community, and field of computing. **Q**

References

1. Adams, J. Bauer, V. and Baichoo, S. An expanding pipeline: Gender in Mauritius. In *Proceedings of the 2003 ACM SIGCSE* (Reno, Nevada, 2003), ACM Press, New York, 59–63.
2. Alvarado, C., Dodds, Z., and Libeskind-Hadas, R. Increasing Women’s participation in computing at Harvey Mudd College. *ACM Inroads*, 4 (Apr. 2012), 55–64.
3. Blum, L. and Frieze, C. As the culture of computing evolves, similarity can be the difference. *Frontiers* 26, 1 (Jan. 2005), 110–125.
4. Blum, L. and Frieze, C. In a more balanced computer science environment, similarity is the difference and computer science is the winner. *Computing Research News* 17, 3 (Mar. 2005).
5. Frieze, C. et al. Where are you really from? Mitigating unconscious bias on campus. EasyChair Preprint no. 531 (2108); <https://doi.org/10.29007/345g>
6. Frieze, C. and Quesenberry, J.L. Kicking Butt in Computer Science: Women in Computing at Carnegie Mellon University. Dog Ear Publishing, 2015.
7. Frieze, C. and Quesenberry, J.L. From difference to diversity: Including women in *The Changing Face of Computing*. In *Proceedings of the 2013 ACM SIGCSE* (Denver, Colorado, 2013), ACM Press, New York, 445–450.
8. Frieze, C. et al. Diversity or difference? New research supports the case for a cultural perspective on women in computing. *Journal of Science Education and Technology* 21, 4 (Apr. 2011), 423–439.
9. Galpin, V. Women in computing around the world. *ACM SIGCSE Bulletin—Women in Computing* 34, 2 (Feb. 2002), 94–100.v
10. Gray, J. *Men Are from Mars, Women Are from Venus*. HarperCollins, New York, 1992.
11. Margolis, J., and Fisher, A. *Unlocking the Clubhouse: Women in Computing*. MIT Press, Cambridge, MA, 2002.
12. Othman, M. and Latih, R. Women in computer science: No shortage here! *Commun. ACM* 49, 3 (Mar. 2006), 111–114.
13. Stephens-Davidowitz, S. Google, tell me. Is my son a genius? *The New York Times* (Jan. 18, 2014).
14. Veilleux, N. et al. The relationship between belonging and ability in computer science. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (2013), 65–70.
15. Zweben, S. and Bizot, B. 2016 Taulbee Survey. *Computing Research Association* 29, 5 (May 2017), 3–51; <https://bit.ly/2STxBeJ>

Carol Frieze (cfrieze@cs.cmu.edu) is Director of Women@SCS and SCS4ALL, organizations that build community on campus, provide leadership and networking opportunities, and promote diversity in computer science, at the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

Jeria L. Quesenberry (jeriaq@andrew.cmu.edu) is an Associate Teaching Professor of Information Systems in the Dietrich College of Humanities and Social Sciences at Carnegie Mellon University, Pittsburgh, PA, USA.

This column is derived from the authors’ book *Kicking Butt in Computer Science: Women in Computing at Carnegie Mellon University*;⁶ the authors’ next book, *Global Perspectives on Women in Computing* (working title), will be published in early 2019 by Cambridge University Press.

The opinions expressed in this column are the authors alone and do not reflect the opinions of the Carnegie Mellon University or any other employee thereof.

Copyright held by authors.



DOI:10.1145/3300228

George V. Neville-Neil

Article development led by queue.acm.org

Kode Vicious

Writing a Test Plan

Establish your hypotheses, methodologies, and expected results.

Dear KV,

We are getting ready for a project release at work, and since we are a small startup, all the developers have been asked to test the code of one of the other developers. We did this by lottery, each of us drawing a name from a hat (we were not allowed to draw our own name). It was an odd way to select testers, but it seems no worse than the processes I have seen at larger companies. The problem for me is not that I have to write tests, but that I also have to write a test plan, one of the requirements imposed by our CEO, who is also the VP of engineering, aka my boss. I have never written an actual test plan, just collections of tests. Of course, I test my own code, but because I wrote the code, I know what I am testing, and it has always been a straightforward process. Should I just write the tests and then list them in the plan? Somehow that does not seem to be what my boss is looking for.

A Man Without a Plan

Dear Planless,

Ah, a test plan, which can be an incredibly useful document or a massive time sink and distraction. Most good test plans start out as one-page documents, because what you must avoid is setting out to test everything—all at once. Instead of just trying to poke at various things that you think you need to test, you need to have a plan of attack as to what and how to test your colleague's code.

A good test plan is a lot like the lab reports some of us had to write for high school science classes. You won't use the



word *hypothesis*, but each test is basically testing one. The plan should start with an outline so that you know you are covering the basics and the main thrust of the code. In place of a hypothesis, you have a statement about what you expect the code to do: "Given input X, we expect to see output Y." Of course, it is not enough to have just a hypothesis; you have to say how you're going to prove or disprove the hypothesis. What is your test method? Do not answer this with, "Run the code."

Now that you know you have to do more than "run the code," let's look at some more useful valid test methods. Describing the test inputs you intend to use is a good start. You do not need to list every possible input, but you should describe the range or shape of what the inputs might be. For a networked system, you might describe the types of messages you will use in your test: "We will send packets of between 64 and 1,500 bytes, with most messages being power-of-two size bytes and containing random bit patterns in their payload sections." That is the test input, but you also must describe the test output. Again, taking a networked system as an example, you might say, "A correct test result is one where all messages are forwarded with-

out any messages being dropped, lost, or corrupted."

If your test has special setup requirements, such as a particular configuration of software or hardware, these must also be included in the plan, probably under their own section marked "Configuration." At the present time, you are the one writing the plan and the tests and probably executing them, but in the future, it may not be you running the tests. All the assumptions that are in your head while writing the test plan must be sought out and then written down. A test plan that leaves out an important but obvious (to you, anyway) requirement is going to be a source of maddening frustration for the next person who tries to use it.

Two more items to note in the test plan are the framework you are using and where it will store its results. Unlike a lab report, your test plan does not need to contain the results of running the test, and, in fact, I would expect that the results would be stored somewhere by the test framework that you are using.

If you can think of each of your tests as an experiment with a hypothesis, a test methodology, and a test result, it should all fall into place rather than falling through the cracks.

KV

Related articles
on queue.acm.org

Debugging on Live Systems

Kode Vicious

<https://queue.acm.org/detail.cfm?id=2031677>

Quality Assurance: Much More than Testing

Stuart Feldman

<https://queue.acm.org/detail.cfm?id=1046943>

Thinking Clearly about Performance

Cary Millsap

<https://queue.acm.org/detail.cfm?id=1854041>

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.

Viewpoint

Tony's Law

Seeking to promote regulations for reliable software for the long-term prosperity of the software industry.

SOMEONE DID NOT tighten the lid, and the ants got into the honey again. This can be prevented by placing the honey jar in a saucer of water, but it is a nuisance, occupies more counter space, and one must remember to replenish the water. So we try at least to remember to tighten the lid.

In the context of security, the software industry does not always tighten the lid. In some cases it fails to put the lid on at all, leaving the honey exposed and inviting. Perhaps the most infamous example of recent years is the WINvote voting machine, dubbed the worst voting machine in the U.S. A security analysis by the Virginia Information Technologies Agency in 2015 found, among other issues, the machines used the deprecated WEP encryption protocol, that the WEP password was hardwired to “abcde,” that the underlying Windows XP (which had not been patched since 2004) administrator password was set to “admin” with no interface to replace it, and that the votes database was not secured and could be modified.⁷ These machines had been used in real elections for more than 10 years.

Such cases constitute malpractice, and call for regulation. Regulation is necessary because not everything can be trusted to market forces. There are many examples in diverse industries. The sale of alcohol to minors is prohibited. Construction and housing cannot use asbestos and lead-based paints due to public health concerns. The automotive industry is required to install seat belts and report pollution levels. Aviation is strictly regulated, in-



cluding airspace utilization (distances between planes), aircrew work schedules, aircraft noise levels, and more. Advertisers are required to add warning labels on advertising for cigarettes and other tobacco products.

Computers are regulated in terms of electrical properties, such as the FCC regulations on radiation and communication. But the software running on computers is not regulated. Nearly 40 years ago, in his ACM A.M. Turing Award acceptance speech, Tony Hoare had the following to say about the principles that guided the implementation of a subset of Algol 60:² “The first principle was security. [...] A consequence of this principle is that every occurrence of every subscript of every subscripted variable was on every occasion checked at runtime against both the upper and the lower declared bounds of the array. Many years later we asked our custom-

ers whether they wished us to provide an option to switch off these checks in the interests of efficiency on production runs. Unanimously, they urged us not to—they already knew how frequently subscript errors occur on production runs where failure to detect them could be disastrous. I note with fear and horror that even in 1980, language designers and users have not learned this lesson. *In any respectable branch of engineering, failure to observe such elementary precautions would have long been against the law.*” [emphasis added].

Hoare said this when personal computers and the Internet were in their infancy, long before the Web, DDoS attacks, and data breaches. Indeed, a lot has changed during this time (see Table 1). But one thing that has not changed is the lack of any meaningful regulation on the software industry.

In retrospect, Hoare’s pronounce-

ment exhibited great foresight. To this day buffer errors represent the single most common vulnerability,^a even more so among high-severity vulnerabilities (see Figure 1 and Figure 2). Just imagine if a law requiring bounds checks had been enacted more than 40 years ago, and there were no buffer overflows today. As it stands, Microsoft for one instituted its Security Development Lifecycle as a mandatory policy in 2004. This includes—among many other features—the option to require compilation with flags that insert bounds checks and the option to ban unsafe library functions. On the one hand this demonstrates that such practices are just a matter of deciding to use them. On the other hand they are still not universally required, and indeed even Microsoft products still occasionally suffer from buffer issues.^b

Similar sentiments have been repeated several times since Hoare’s speech. Twelve years ago, ACM President David Patterson put forward the “SPUR manifesto,”³ suggesting the development of 21st-century computer (software) systems should focus on security, privacy, usability, and reliability—SPUR. The goal should be to be as safe as 20th-century banking, as low maintenance as 20th-century radio, and as reliable as 20th-century telephony. But more than a decade has passed, and it seems the focus on low cost, multiple features, and above all time to market is as strong as ever. Manufacturers of home appliances compete, among other ways, by offering superior warranties for their products. The software industry, in contradistinction, has been getting away with software that comes “without warranty of any kind, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.”

Indeed, lectures such as Patterson’s are typically either ignored or stir up a chorus of naysayers. The typical argu-

a The NIST National Vulnerability Database uses 124 of the nearly 1,000 types listed in the Common Weakness Enumeration to categorize vulnerabilities. In 2015–2017, buffer errors CWE-119 accounted for 15.2%–18.4% of all vulnerabilities each year. The next highest categories were information leak/disclosure CWE-200 at 9.3%–10.9%, permissions, privileges, and access control CWE-264 at 8.2%–10.0%, and cross-site scripting CWE-79 at 7.3%–11.2%.

b One example: Microsoft Office Equation Editor stack buffer overflow; see <https://bit.ly/2zTngss>

Table 1. Changes in software and computing in the last 30 years.

1980s	2010s
C pointers	Java garbage collection
Emacs	Eclipse
Math library	Frameworks
Ad hoc programming	Agile methodology
Waterfall	Evolution/continuous integration
Flowcharts	UML
Write your own sort	Copy from Stack Overflow
Computer room	Computer in your pocket
Hard disk	Cloud
Text terminals	Touch screens
Email	Internet of Things
No regulation	No regulation

Figure 1. The number of software vulnerabilities cataloged by the NIST National Vulnerability Database skyrocketed in 2017, and the fraction of vulnerabilities involving buffers (either categorized as “buffer error” or containing the keyword “buffer”) kept pace.

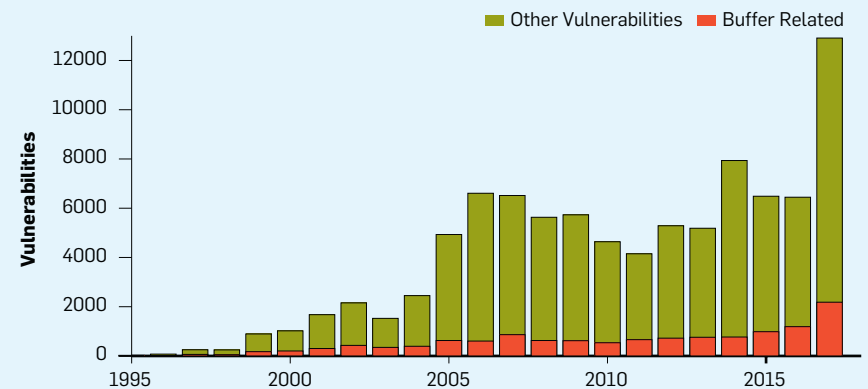
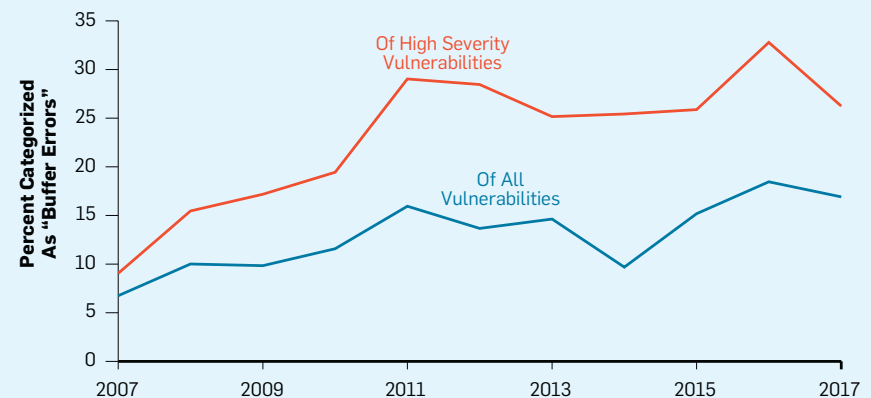


Figure 2. According to the National Vulnerability Database, since the beginning of the decade approximately 15% of all vulnerabilities have been related to buffer errors, and this rises to between one-quarter and one-third of the vulnerabilities if only those with a high severity score are considered.



acm

Advertise with ACM!

Reach the innovators
and thought leaders
working at the
cutting edge
of computing
and information
technology through
ACM's magazines,
websites
and newsletters.



Request a media kit
with specifications
and pricing:

Ilia Rodriguez

+1 212-626-0686

acmm mediasales@acm.org

acm

media

Table 2. Notable security incidents from 2007–2017.

Year	Incident	Significance
2007	Massive DDoS attacks on organizations and infrastructure in Estonia	First demonstration of extensive countrywide disruptions, possibly in connection to international relations
2010	The Stuxnet cyber-weapon is used to disable physical centrifuges used in Iran's nuclear program	Demonstration of potential impact on computer-controlled physical infrastructure, and demonstration of cyber-weapons that jump air-gaps and remain undetected for long periods
2013	Yahoo is hacked and data about all three billion user accounts is stolen	Biggest data breach of its kind
2016	Hackers break into DNC computers and disseminate confidential documents	Strategic hacking with possible effect on the outcome of the U.S. presidential election
	DDoS attacks using a botnet of some 1.5 million IoT devices (ironically, mainly security cameras)	Demonstration of new vulnerabilities resulting from technological progress and insufficient consideration of security
2017	The WannaCry ransomware infects more than 200,000 computers in 150 countries, causing disruptions such as the closing down of 16 hospitals in the U.K.	Demonstration of global-scale cyber crime and putting human lives at risk

ments are the perceived monetary costs, the difficulties or even the impossibility of implementation, and the fear of reduced innovation and technological progress. Schneider, in a recent *Communications Viewpoint*, also notes the need for a detailed cost/benefit analysis to ascertain what society is willing to pay for improved security, where the costs also include reduced convenience (due to the need for authentication) and functionality (due to isolation).⁴ And indeed all regulations are, by definition, limiting. But do we really need to wait for a large-scale security catastrophe, possibly including significant loss of life, before we act at all? As the Microsoft example shows, extensive technological solutions and best practices actually already exist. It is just a matter of making their use pervasive.

So why are software security faults tolerated? A possible explanation is that software deficiencies have so far been less tangible than those of traditional industries. Many people install multiple locks on their doors and would consider holding intruders to their homes at gunpoint, but fail to take sufficient safeguards to protect their home computers from hackers. The problems resulting from identity theft are much more common but also much more bureaucratic, boring, and less visual compared to more dramatic problems such as exploding gas tanks in pickup trucks.

But above all else, it seems there is a market failure in incentivizing the indus-

try to take the required actions.^{1,6} Buyers will not pay a premium for value (security) they cannot measure, and which in many cases does not affect them personally and directly. Approaches suggested by economists to measure the value of protection do not help because the cost of a security catastrophe is up to anyone's imagination. This has prevented an insurance industry for software producers from emerging, and as Anderson and Moore write, "if this were the law, it is unlikely that Microsoft would be able to buy insurance."¹ In practice, the reduction in stock value after disclosing a vulnerability is less than 1%.⁵ The abstract danger of large-scale attacks leading to financial loss and even loss of human life is not enough to change this.

At the same time, we are inundated by increasing numbers of reports of data breaches and hackers infiltrating various systems (see Table 2 for prominent recent examples). Some of these incidents demonstrate that extensive physical civil infrastructures are at peril across the globe—including hospitals, power plants, water works, transportation systems, and even nuclear facilities. And the root cause at least in some cases is the failure of the software to take appropriate precautions.

The software systems in a modern car—not to mention a passenger plane or a jet fighter—are of a scope and complexity that rivals any operating system or database produced by the traditional software industry. Indeed, every industry

is now a software industry. And the products of every industry are vulnerable due to software defects. In such a context, required software regulation includes:

- Transparency: the obligation to investigate and report all exploits including their technical details.
- The prohibition of dangerous practices, such as not using type-safe languages and appropriate encryption.
- Holding companies accountable for their unsafe practices.

These requirements need the backing of legal regulations, because market forces compel industry not to invest in security too much. The market promotes a race to the bottom; except in niche applications, whoever is faster to market and cheaper wins, and whoever is tardy due to excessive investment in security loses. Regulation is the only way to level the playing field, forcing everybody to invest in what they know to be needed but think they cannot afford to do when the competition does not.

Of course, it will not be easy to implement these ideas and agree on the myriad details that need to be settled. Who gets to decide what is a “dangerous practice”? How do we deal with installed systems and legacy code? Who is charged with enforcing compliance? Moreover, it is not clear how to make this happen at the political level. In addition, no single country has jurisdiction over all software production. So a system of certification is required to enable software developers to identify reliable software, and to perform due diligence in selecting what other software to use.

International frameworks already exist demonstrating these issues can be solved. The EU General Data Protection Regulation (GDPR), which concerns the rights of individuals to control how their personal information is collected and processed, is an encouraging example. Another example is the Common Criteria for Information Technology Security Evaluation, an international framework for the mutual recognition of secure IT products. But this covers only high-level desiderata for security, not the regulation of low-level technicalities. This gap is partly filled by the Motor Industry Software Reliability Association (MISRA), which has defined a set of suggested safe coding practices for the automotive indus-

Regulation is in the interest of the long-term prosperity of the software industry.

try. However, these are not required by any formal regulations.

Protracted discussions on what to do and what we are willing to pay for are counterproductive. Such things cannot be planned in advance. Instead we should learn from the iterative approach to constructing software: try to identify the regulations that promise the highest reward for the lowest cost, work to enact them, learn from the process and the results, and repeat.

Regulation is in the interest of the long-term prosperity of the software industry no less than in the interest of society as a whole. Software vendors with integrity should stop resisting regulation and instead work to advance it. The experience gained will be extremely important in discussing and enacting further regulations, both in a preemptive manner and—in the worst-case scenario—in the aftermath of a security catastrophe. ■

References

1. Anderson, R. and Moore, T. The economics of information security. *Science* 314, 5799 (Oct. 26, 2006), 610–613; <https://bit.ly/2GctSYd>.
2. Hoare, C.A.R. The emperor's old clothes. *Commun. ACM* 24, 2 (Feb. 1981), 75–83; DOI: 10.1145/358549.358561.
3. Patterson, D.A. 20th century vs. 21st century C&C: The SPUR manifesto. *Commun. ACM* 48, 3 (Mar. 2005), 15–16; DOI: 10.1145/1047671.1047688.
4. Schneider, F.B. Impediments with policy interventions to foster cybersecurity. *Commun. ACM* 61, 3 (Mar. 2018), 36–38; DOI: 10.1145/3180493.
5. Telang, R. and Wattal, S. An empirical analysis of the impact of software vulnerability announcements on firm stock price. *IEEE Trans. Softw. Eng.* 33, 8 (Aug. 2007), 544–557; DOI: 10.1109/TSE.2007.70712.
6. Vardi, M.Y. Cyber insecurity and cyber libertarianism. *Commun. ACM* 60, 5 (May 2017), DOI: 10.1145/3073731.
7. Virginia Information Technologies Agency. Security assessment of WINvote voting equipment for department of elections. (Apr. 14, 2015); <https://bit.ly/2EgvBct>

Dror G. Feitelson (feit@cs.huji.ac.il) is the Berthold Badler Chair in Computer Science at The Rachel and Selim Benin School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel.

Copyright held by author.

Calendar of Events

February 11–15

WSDM 2019: The 12th ACM International Conference on Web Search and Data Mining, Melbourne, VIC, Australia, Co-Sponsored: ACM/SIG, Contact: Alistair M. Moffat, Email: ammoffat@unimelb.edu.au

February 16–20

PPoPP '19: 24th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Washington, DC, Co-Sponsored: ACM/SIG, Contact: Jeff Hollingsworth, Email: hollings@cs.umd.edu

February 24–26

FPGA '19: The 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Seaside, CA, Sponsored: ACM/SIG, Contact: Kia Bazargan, Email: generalchair@isfpga.org

February 25–26

HotMobile '19: The 20th International Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, Sponsored: ACM/SIG, Contact: Alec Wolman, Email: alec.wolman@gmail.com

February 25–March 3

SIGCSE '19: The 50th ACM Technical Symposium on Computing Science Education, Minneapolis, MN, Sponsored: ACM/SIG, Contact: Manuel A. Perez-Quinones, Email: perez.quinones@uncc.edu

March

March 10–14

CHIIR '19: Conference on Human Information Interaction and Retrieval, Glasgow, United Kingdom, Sponsored: ACM/SIG, Contact: Martin Halvey, Email: martin.halvey@gmail.com

Viewpoint

Do We Really Need Computational Thinking?

Considering the expression “computational thinking” as an entry point to understand why the fundamental contribution of computing to science is the shift from solving problems to having problems solved.

I CONFESS UPFRONT, the title of this Viewpoint is meant to attract readers’ attention. As a computer scientist, I am convinced we need the *concept* of computational thinking, interpreted as “being able to think like a computer scientist and being able to apply this competence to every field of human endeavor.”

The focus of this Viewpoint is to discuss to what extent we need the expression “computational thinking” (CT). The term was already known through the work of Seymour Papert,¹³ many computational scientists,⁵ and a recent paper¹⁵ clarifies both its historical development and intellectual roots. After the widely cited *Communications* Viewpoint by Jeannette Wing,¹⁹ and thanks to her role at NSF,⁶ an extensive discussion opened with hundreds of subsequent papers dissecting the expression. There is not yet a commonly agreed definition of CT—what I consider in this Viewpoint is whether we really need a definition and for which goal.

To anticipate the conclusion, we probably need the expression as an instrument, as a shorthand reference to a well-structured concept, but it might be dangerous to insist too much on it and to try to precisely characterize it. It should serve just as a brief explanation of why computer science (or informatics, or computing; I will use these terms interchangeably) is a novel and independent scientific subject and to argue for the need of teaching informatics in schools.



Wing discussed CT to argue it is important every student is taught “*how a computer scientist thinks*,”¹⁹ which I interpret to mean it is important to teach computer science to every student. From this perspective, what is important is stressing the educational value of informatics for all students—Wing was in line with what other well-known scientists had said earlier; I mention several here.

Donald Knuth, well known by mathematicians and computer scientists, in 1974 wrote: “Actually, a person does not really understand something until he can teach it to a computer.”¹⁰ George

Forsythe, a former ACM president and one of the founding fathers of computer science education in academia, in 1968 wrote: “The most valuable acquisition in a scientific or technical education are the general-purpose mental tools which remain serviceable for a lifetime. I rate natural language and mathematics as the most important of these tools, and computer science as a third.”⁹ Even if both citations are not relative to a school education context, in my view they clearly support the importance of teaching computer science in schools to all students.

However, the wide popularity gained by CT after Wing’s *Communications*

Viewpoint risks spoiling the original aim. Increasingly, people are considering CT a new subject, somehow different or distinct from computer science. In the quest to identify the definition that Wing did not provide, people are stressing one or other aspect (abstraction, recursivity, problem solving, ...) and in doing so they obscure its meaning. See Armoni² and Denning⁵ for clear and illuminating discussions of this issue.

This situation becomes even more garbled when it comes to education. Speaking about teaching CT is a very risky attitude: philosophers, rightly, ask what we mean by “teaching thinking”; mathematicians appropriately observe that many characteristics of CT (such as abstraction, recursivity, problem solving, ...) are also proper of mathematics (which they do not call “mathematical thinking”); pedagogues ask how we can be sure CT is really effective in education; teachers want to know which are the methods and the tools for teaching this new discipline and how they can learn to teach it; and parents are alternately happy because it appears school has finally started to align itself to the digital society while they are also concerned about what will happen to their children in the future if they just learn to code with the language of today.

I think a large part of the community of computing scientists and educators is convinced the original *Communications* Viewpoint by Wing was aiming at “start rolling the ball” and what needs to be done is teaching informatics in schools, possibly beginning at an early age. Moreover, I am convinced the same people are fully able to understand the meaning of Wing’s expression “to think like a computer scientist” without the need of exactly explaining it. Or, if it is absolutely needed, they might agree with the self-referential sentence “CT is the set of mental and cognitive competences obtained by the study and practice of computer science”: the “tacit knowledge” defined by Polanyi.¹⁴

Already in 1974 Knuth warned, in discussing computer science, that “the underlying concepts are much more important than the name.”¹⁰ It is much more so, I think, for CT. What really counts is the fact that computing is taught early in schools. This is actually the path being followed by some ma-

ior countries. Here, I discuss the three most relevant ones.

In England, the national computing programmes of study,^a published by the Department of Education in September 2013 and mandatory since school year 2014–2015, uses CT in the presented sense of what one gets by the study and practice of computing. In fact, it uses it in the opening statement “A high-quality computing education equips pupils to use CT and creativity to understand and change the world” and then just two more times, in goals for Key Stage 3 “understand several key algorithms that reflect CT” and KS4 “develop and apply their analytic, problem-solving, design, and CT skills.” The curriculum never defines the term.

In the U.S., the “Every Student Succeeds Act” (ESSA), approved by Congress in 2015 with bipartisan support, has introduced computer science among the “well rounded educational subjects” that needs to be taught in schools “with the purpose of providing all students access to an enriched curriculum and educational experience,” and does not contain at all the term “computational thinking.” In January 2016, President Obama launched the initiative “CS For All” whose goal is “to empower all American students from kindergarten through high school to learn computer science and be equipped with the CT skills they need ...”. Once again, CT is what you get when you have learned computer science.

In France, the Académie des Sciences—the highest institution representing French scientists—published in May 2013 the report “L’enseignement de l’informatique en France. Il est urgent de ne plus attendre,” (“Teaching computer science in France. Tomorrow can’t wait.”) recommending—for what regard the teaching of computer science (“informatique”)—“teaching should start at the primary level, through exposure to the notions of computer science and algorithms, ... <and> should be further developed in middle and secondary school.” Analyzing their use of CT (“pensée informatique”), it is clear that in their vision the term denotes the specific habits of thinking developed by learning computer science. Just a couple of examples: “computing ... leads to

a different way of thinking, called CT” and “learning about programming is a way to discover the rudiments of CT.”

It emerges, from these three examples, that CT is not a new subject to teach and what should be taught in school is informatics.

But on the other side, the high number of papers published with CT in their title or abstract (the ACM Digital Library alone contains more than 400) indicates a lot of people seem to argue (and even Wing seemed to agree²¹) that CT is something new and different. Some even say “coding” (which they consider different from “programming”) is all you need to learn it! A discussion of risks related to this approach and other delicate issues regarding CT appeared in a recent *Communications* column.⁸

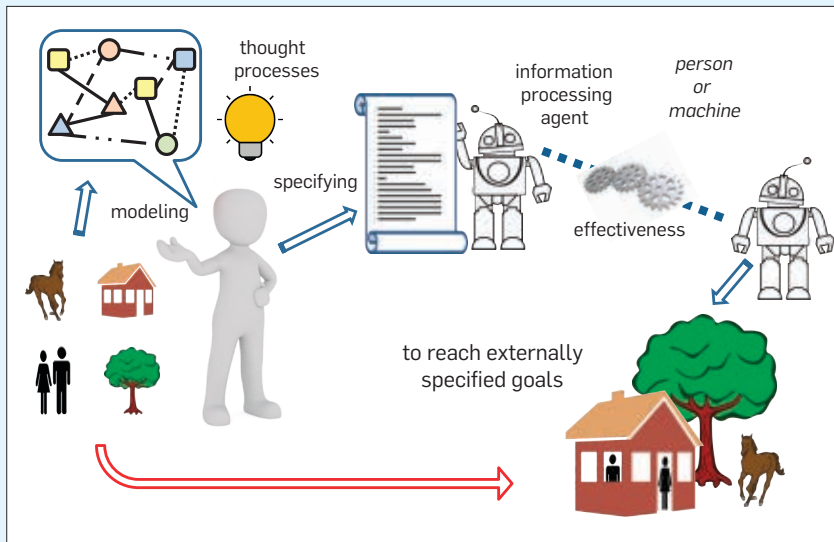
I am convinced that considering CT as something new and different is misleading: in the long run it will do more harm than benefit to informatics. After all, they do not teach “linguistic thinking” or “mathematical thinking” in schools and they do not have “body of knowledge” or “assessment methods” for these subjects. They just teach (and assess competences in) “English”^b and “Mathematics.” Subsequently, the various linguistic (resp. mathematical) competences gained by study of English (resp. Mathematics), beyond being used in themselves, find additional uses in other disciplines. Between CT and computing there exists the same relation. Therefore, we should discuss what to teach and how to evaluate competences regarding informatics in primary/middle/secondary schools, and forget about teaching and evaluating competences in CT.

In summary, speaking about CT helps people understand that: we are focusing on scientific and cultural aspects of computing; *we are not dealing with system and tools, but with principles and methods*; we are focusing on the core scientific concepts of computing, on its *conceptual kernel*.¹¹ Different from what happens with language and math, we are forced to explicit this distinction since computers are what embodies informatics for most of people. In addition, we do not think the “computer scientists’ way of thinking” is better than others, just that it offers a complemen-

a See <https://bit.ly/1f7PIFU>

b Or the relevant native language.

Modeling a situation and specifying the ways an information-processing agent can effectively operate within it to reach an externally specified (set of) goal(s).



tary and useful conceptual paradigm to describe reality.⁷

At this point people usually ask which is this “conceptual kernel” and which examples can we provide. This is a critical passage to explain to people the novelty of informatics among scientific disciplines and its educational value. For this purpose, the formulation attributed to Cuny, Snyder and Wing¹⁶ is appropriate: “CT is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.” This is almost the same definition given by Aho¹ “CT is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms” and Wing acknowledges the input received by him.²⁰ The big issue, as Armoni has clearly pointed out,² is that by taking any of these as the definition of a new discipline instead of as an explanation and trying to fully operationalize it causes more problems than benefits.

The issue of explaining in which sense “the way a computer scientist thinks” is different from “the way a mathematician thinks” is indeed an important one. Knuth had a brilliant example in his 1974 paper, which, unfortunately, is not at a level laypeople can understand. It regarded the problem of finding the “greatest common right divisor” of two $n \times n$ integer matrices A and B . The math-

ematician’s answer is: “Let R be the ring of integer matrices; in this ring the sum of two principal left ideals is principal, so let D be such that $RA + RB = RD$. Then D is the greatest common right divisor of A and B .”¹⁰ Clearly unsatisfactory for a computer scientist, for whom *a solution is provided by a process computing the answer and not by an equation defining the answer*. I have intentionally used the word “process” instead of the more usual “algorithm” to stress the fact that we have a “process” only when the algorithm has been implemented in a suitable “language” and an “automaton” executes the obtained code. In such a way three of the main pillars on which computer science is based—algorithm, language, and machine—are all involved in characterizing the difference between the viewpoints of the mathematician and the computer scientist.

I therefore think that, whenever either the Cuny, Snyder, and Wing’s formulation or Aho’s one is used for this explanatory purpose, the utmost stress must be put on the involvement of the *information processing agent* (that is, the “automaton,” be it a machine or a person acting mechanically). *Without the agent and its capability to operate effectively, there is no informatics*, just mathematics, which indeed has been solving problems for millennia, discovering and applying along the way abstraction, decomposition, recursion, and so on.

Indeed, in looking backward toward how computer science was born, it is clear the cultural seeds are in the mathematicians’ quest for automatizing theorem proving, in their efforts to unload the burden of solving problems onto machines. This shift in viewpoint, *from solving problems to having problems solved* is the intellectual birth of informatics and is the “difference which makes a difference,”³ setting informatics in its own proper and unique place in the context of all sciences. The importance of the “automaton” to give full sense to CT was also made explicit^c and emphasized.^{1,6}

I also dare to provide, for the same demonstrative purpose, a more general explanation of what CT is, which is somehow along a direction already hinted at by Wing,¹⁶ who clarified: “My interpretation of the words ‘problem’ and ‘solution’ is broad. I mean not just mathematically well-defined problems whose solutions are completely analyzable, for example, a proof, an algorithm, or a program, but also real-world problems whose solutions might be in the form of large, complex software systems.” Nevertheless, Wing still used the word “problem,” which conveys the meaning of something that needs to be solved.

Since solving a problem is just an instance of a situation where one wants to reach a specified goal, here is my formulation: *Computational thinking is the thought processes involved in modeling a situation and specifying the ways an information-processing agent can effectively operate within it to reach an externally specified (set of) goal(s).* (See the accompanying figure.)

There are two main differences: one is speaking about a situation where the agent operates instead of a problem it has to solve, the other is clarifying the agent does not define by itself its overall (set of) goal(s) but gets it from the outside.^d My formulation is also closer to more recent characterizations of computation as an unbounded process.¹⁸

^c Aho wrote: “An important part of this process is finding appropriate models of computation with which to formulate the problem and derive its solutions.” We could say, in a somewhat literary style, “the model is the agent is the model.”

^d If we allowed the agent to choose its own goals, we would leave computing and enter the realm of free-will entities.

We have thus a more general explanation of what CT is, covering also cases that are of high interest for schools and education: simulations in other disciplines, where one has to build and manipulate a visible representation of physical laws and/or natural/social phenomena (that is, to model a situation and explore its possible evolution) rather than to solve a problem. Simulation is a very powerful tool to improve understanding and computing is unique in its capability of making concrete the abstract models defined by a simulation.² In addition, we have a formulation that can be used to explain why mathematics or other sciences are not enough for these purposes.

In such a way informatics can more clearly explain its dual role¹² both as a *fundamental scientific subject*, with its own independent set of concepts, and as a *discipline of transversal value*, providing methods contributing to a better understanding of other disciplines.⁷ This latter role of computing is also of particular importance for its introduction as a regular subject in schools, and can constitute a solid argument for con-

sidering it as a foundational discipline, on par with mathematics.^{4,17} C

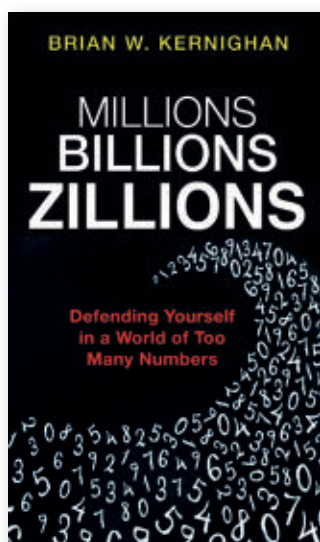
References

1. Aho, A.V. Computation and computational thinking. *Ubiquity*, vol.2011, issue January, Article no. 1, January 2011. ACM Press. DOI: <https://doi.org/10.1145/1922681.1922682>
2. Armoni, M. Computer science, computational thinking, programming, coding: The anomalies of transitivity in K–12 computer science education. *ACM Inroads* 7, 4 (Dec. 2015), 24–27.
3. Bateson, G. Form, substance and difference. In *Steps to an Ecology of Mind*. University of Chicago Press, 1972.
4. Caspersen, M.E. et al. *Informatics for All: The Strategy*. ACM/Informatics Europe, NY, 2017; <https://doi.org/10.1145/3185594>
5. Denning, P.J. Computational thinking in science. *American Scientist* 105, (Jan.–Feb. 2017); 13–17.
6. Denning, P. Remaining trouble spots with computational thinking. *Commun. ACM* 60, 6 (June 2017), 33–39.
7. Denning, P.J. and Rosenbloom, P.S. Computing: The fourth great domain of science. *Commun. ACM* 52, 9 (Sept. 2009), 27–29.
8. Denning, P.J., Tedre, M., and Yongpradit, P. Misconceptions about computer science. *Commun. ACM* 60, 3 (Mar. 2017), 31–33.
9. Forsythe, G.E. What to do till the computer scientist comes. *The American Mathematical Monthly* 75, (May 1968), 454–462; <https://bit.ly/2S19xXo>
10. Knuth, D.E. Computer science and its relation to mathematics. *The American Mathematical Monthly* 81, 4 (Apr. 1974), 323–343; <https://bit.ly/2ErMMU>
11. Lodi, M., Martini, S., and Nardelli, E. Abbiamo davvero bisogno del pensiero computazionale? *Mondo Digitale* 72 (Nov. 2017), AICA, Milan; <https://bit.ly/2CLJcr5>
12. Nardelli, E. Informatica nella scuola: disciplina fondamentale e trasversale, ovvero “di cosa parliamo quando parliamo di pensiero computazionale.” *Scienze e Ricerche Magazine* (Apr. 2017), 36–40; <https://bit.ly/2GqszFk>
13. Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.
14. Polanyi, M. *The Tacit Dimension*. The University of Chicago Press, 1966.
15. Tedre, M. and Denning, P.J. The long quest for computational thinking. In *Proceedings of the 16th Koli Calling Conference on Computing Education Research*. (Nov. 2016), 120–129.
16. The LINK. Research Notebook: Computational Thinking—What and Why?. The Magazine of Carnegie Mellon University’s School of Computer Science, March 2011; <https://bit.ly/2UTeAed>
17. Vahrenhold, J. et al. *Informatics Education in Europe: Are We All In The Same Boat?* ACM/Informatics Europe, NY, 2017; <https://doi.org/10.1145/3106077>
18. van Leeuwen, J. and Wiedermann, J. Computation as an unbounded process. *Theoretical Computer Science* 429, (2012), 202–212.
19. Wing, J. Computational thinking. *Commun. ACM* 49, 3 (Mar. 2006), 33–35.
20. Wing, J. Computational thinking benefit society. *Social Issues in Computing blog*, January 2014; <https://bit.ly/2SONisk>
21. Wing, J. Computational thinking and thinking about computing. *Philosophical Transactions of The Royal Society A366*, 37 (2008): 3717–3725.

Enrico Nardelli (nardelli@mat.uniroma2.it) is a Full Professor in Informatics in the Department of Mathematics at the University of Rome “Tor Vergata,” Italy. He is currently the president of Informatics Europe, the association representing the academic and research Informatics community in Europe.

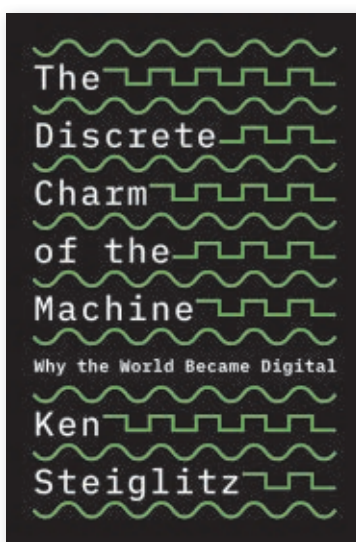
Discussions with Mehdi Jazayeri, Jan van Leeuwen, Michael Lodi, Simone Martini, and Guido Proietti have been useful to focus ideas and improve presentation; comments from referees have also been greatly helpful. Many of the ideas first presented in this Viewpoint have been further developed by the author in subsequent papers since this material was reviewed, revised, and accepted for publication in early 2017.

Copyright held by author.



“The indispensable guide to numerical trickery, deception, and flimflam!”
—Harry Lewis,
coauthor of *Blown to Bits*

Cloth \$22.95



“An inspirational must-read and delightful guide for anyone interested in traveling from the computational past through to the present.”
—Andrew Adamatzky,
University of the West of England

Cloth \$27.95



“What Can Be Computed? should succeed brilliantly in capturing the imagination of students.”
—Matt Franklin,
University of California, Davis

Cloth \$85.00

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

**A discussion with Jacek Czerwotka,
Michaela Greiler, Christian Bird,
Lucas Panjer, and Terry Coatta**

CodeFlow: Improving the Code Review Process at Microsoft

YOU MAY BE wondering, “Code review process? Isn’t that obvious?” But code reviews are pervasive. Any developer is likely to be asked at any time to review someone else’s code. And you can be sure your code is reviewed. For some developers, code reviews take up a portion of each day. So there is your answer: large numbers of very well-compensated people spend a great deal of time on this activity, meaning the aggregate costs are substantial. If you’re talking about a development shop the size of, say, Microsoft ... well, then, the investment regularly made in code reviews

can amount to something quite impressive indeed.

That is only one of the reasons that **Jacek Czerwotka** and his Tools for Software Engineers (TSE) team at Microsoft set out to study how the code-review process plays out across the company. Another reason had to do with taking on a challenge they found interesting in the sense that, beyond their important role in software engineering integration, code reviews involve some rather complex social dynamics that elude simple modeling.

Then there also was the fact that Microsoft’s code-review tool represented an opportunity to touch every developer throughout the entire company. For a group charged with boosting developer productivity, that is just the sort of lever dreams are made of. What’s more, the tool also offered TSE’s researchers something they could instrument to collect data and generate metrics that, in turn, could be used to enable further research.

So, that is why the group set out on this journey. To recount what it was like, where it led, and what was learned along the way, Czerwotka discusses the undertaking here, along with fellow researchers **Michaela Greiler** and **Christian Bird**. Also on hand to help steer the discussion are **Lucas Panjer**, the senior director of engineering at Tasktop, and **Terry Coatta**, the CTO at Marine Learning Systems, a Vancouver-based startup working to develop a learning platform.

LUCAS PANJER: What exactly is it that initially moved you to zero in on the code-review process?

JACEK CZERWOTKA: This group was formed several years ago with the goal of encouraging the adoption of a common set of software engineering tools across the whole of Microsoft. We have been on this path for a while now. We are not done yet. But there are a few places where we’ve managed to centralize the tools quickly, and one of those is in code-review tooling.

Clearly, in looking at that aspect of the engineering workflow, we saw



there were already some tools in place, so we just concentrated on determining what we could do to make improvements. First we wanted to learn what we could from actual experience since you always want to start with a foundation grounded in practice, as well as theory. So, we started looking at any qualitative or quantitative data we could get our hands on that had to do with the code-review tooling and process already in place at Microsoft. That's how we started on this journey of trying to understand where the process originated and how it has evolved over time. What are the factors that drove that evolution? How is the process currently applied? How does it work

with open source? How does it work within Microsoft? And what happens when we find ourselves collaborating with others?

LP: What did you end up initially focusing on?

CHRISTIAN BIRD: In general, we wanted to find out what prompted people to do code reviews in the first place. How many people were usually involved? What types of issues were being raised? What was it that led people to make changes? And what typically led people not to make changes?

TERRY COATTA: Were the engineering teams themselves pushing for this line of inquiry? That is, were people coming to you to say, "We're sure spending a lot of time with code reviews, but it

doesn't seem like we're getting all that much out of it?"

CB: Mostly it was because this was an area where the data was both plentiful and readily available. With that being said, once people found out what we were doing, they proved to be quite receptive. It wasn't like they wondered why we were doing this research. In fact, it was just the opposite. People generally were very supportive of improving the code-review process and, if anything, said they wished it was treated as a first-class citizen. Also, many were pretty excited to learn there was data available they would be able to track themselves.

LP: Once people engaged with you and told you what they thought was



JACEK CZERWONKA

One of the most interesting things to surface from instrumenting CodeFlow was just how much time people were actively spending in the review tool.



valuable, did they also let you know what else they wanted?

CB: What people wanted for the most part was the ability to do their own tracking, along with a way to look at how they were doing in comparison to other teams. We came up with metrics that align with some of the targets teams at Microsoft have for what they want to achieve at different points in the software development process. For example, they would want to know if they were on track for getting a commit into master within a month. Or they would want to see if they were well on their way to achieving 80% test coverage.

Similarly, for code review some teams had targets, while others did not since they didn't have a way to measure that. So, they might decide that at least two people should sign off on every code review and that each review would have to be completed within a 24-hour period. Until we started collecting the data around code reviews, analyzing it, and then making it more generally available, teams had no way of measuring that. Yet they *wanted* to be able to do that since they were already measuring other parts of their development process. As a consequence, people started coming to tell us what metrics they would find useful. Then we would just add those to metrics we were already collecting. It turns out that much of our effort was actually driven by what the development teams themselves were telling us they wanted to be able to measure.

TC: Since you say this tooling for code reviews is something everybody at Microsoft now uses, can you give us a brief description of the features it offers and how you think those compare with what is available to most people outside of Microsoft?

JC: Well, we're talking now about things we did with our tool [called Code Flow] a few years ago, and tooling has a way of converging out in the world at large over that much time. So, some of the changes we made back then might now seem fairly obvious to people who are using other code-review tools that have since come to work in much the same way.

The brief summary is that we made a number of changes to finely tune the underlying subsystem. We also trained the tool to be super-precise in terms of tracking changes as people move through numerous software iterations. That is, as you move from one revision to the next, you can imagine that your code changes end up moving around as some code gets deleted, some new lines are added, and chunks of code are shuffled around. That can throw your comment tracking severely out of sync with what you had once intended. Overcoming that took work, but we now know from feedback that it's greatly appreciated and thus well worth the effort.

Another thing we focused on was performance. For that reason, even today CodeFlow remains a tool that works client-side, meaning you can download your change first and then



MICHAELA GREILER

While the popular notion is that code reviews are mostly about finding bugs, only a very small percentage of the code-review comments we studied actually had anything to do with bugs at all.



interact with it, which makes switching between files and different regions very, very fast.

It also helps that CodeFlow has essentially become ubiquitous throughout Microsoft. That's because we used something like a viral marketing strategy in that the moment you were added as a reviewer, you received a notification, which allowed you to open the review by simply clicking on a link. Then the CodeFlow client would be installed and the review would be opened. So, soon after the tool was introduced to a group, it would start to permeate the fabric of that team pretty much all on its own. The choice not to require a special install for CodeFlow proved to be a really good one.

LP: Is there anything in particular from the user's perspective that distinguishes CodeFlow from either Git or Gerrit? How would you say it differs from what you find with pull requests and patch set-based tooling?

CB: It comes down to being a native app rather than a Web capability, meaning it enables much richer interactions than you would get otherwise. I've been through the Git and the pull request stuff, and it's absolutely the case that you can easily jump around from comment to comment, and you also get things that work like score boxes. Which is to say they feel like rich native clients, so I realize you can accomplish this with a Web experience.

As for Git and Gerrit code reviews, what you get there just amounts to lists

of diffs. I mean, you also can add comments, but, in the end, that just makes it more difficult to track things or navigate everything effectively.

So, the fact that CodeFlow is native and is treated like a first-class citizen on the desktop makes it more usable.

MICHAELA GREILER: I also really like the richness of CodeFlow's commenting features. You can, for example, mark just a single character within a line instead of calling out the whole line of code. That way, people can immediately see exactly where the issue is. Also, to this day, very few code-review tools let you span regions, but with CodeFlow you can attach a comment at the same time to a number of deleted lines and inserted lines—and then track all of that through succeeding iterations. Another feature worth pointing out is comment threading, which lets you resolve an entire thread of comments at the same time rather than dealing with each comment individually.

Code reviews generally conjure up notions of troubleshooting. More specifically, people tend to associate them with the never-ending search for bugs.

It turns out that is not nearly as central to the code-review process as you might think. Which is not to say that finding bugs is unimportant or discouraged. And yet it seems the real win comes in the form of improved long-term code maintainability.



CHRISTIAN BIRD

The code-review process we now have at Microsoft has more or less grown organically—through experimentation—from the grassroots.



LP: Which problems did you decide to attack first?

JC: Most of the issues we chose to focus on were process oriented. The tool itself is quite flexible and adaptable to practically any process. We spent a lot of time trying to understand the benefits of code review and what was getting in the way of achieving those advantages. Also, we wanted to understand how the existing code-review tool was being used. We were interested in learning more about the costs and the turnaround times in hopes we would be better able to see what the drivers were.

MG: Also, one of the issues we looked at was how to create a reviewer recommender since programmers had been complaining to us about how difficult it was to find the right people to look over their code. Chris started working on a tool that would deliver a listing of people with the expertise to match the sorts of problems addressed by your code, along with suggestions as to which of these people you might want to add to a review.

Something else Chris and I studied for a while was code-review usefulness. That wasn't a problem we were trying to solve, of course, but we did want to understand which aspects of code reviews tend to be most valued by engineers—that is, by both reviewers and programmers. What did they see as being most useful? It didn't take us long to conclude that it was not the mere decision to accept or rework the code that

the reviewers were interested in, it was the comments that added to the value of the review. On the other hand, some comments just increase the burden of the code review and slow down the development process. So then we wanted to know what kinds of comments they found most useful, since we could then start thinking about how to encourage and lend greater emphasis to those.

JC: Just as this interesting question of usefulness led to practical implications later on, the same might be said of the work that was done to look into other process-related questions. For example, how many people should you include in a code review? Is there a number beyond which it becomes counterproductive? We all intuitively feel that smaller reviews are better, but where exactly to draw that line? And what's the optimal amount of time to allow for a review?

MG: Another interesting thing we found is that, while the popular notion is that code reviews are mostly about finding bugs, only a very small percentage of the code-review comments we studied actually had anything to do with bugs at all. In fact, most of the comments were about structural issues and style problems. Sometimes they were even about really minor issues, like spelling. Basically, what we found was that many reviewers were using their commenting platform to *discuss* these issues and share their knowledge.

We found it very enlightening to categorize these comments and do



LUCAS PANJER

Are you saying that after you have created these tools for research purposes, other teams will go on to use them to reflect on their own processes?



some mappings to determine which ones were thought to be the most interesting or useful. It turns out that generally proved to be comments that identified functional issues, pointed out missing validation checks, or offered suggestions related to API usage or best practices.

LP: Just for context, can you also speak to the scale of this research—the size of the codebase you were working with, the number of code reviews you analyzed, or the number of developers who were involved?

CB: We did a number of different studies, many of which were more quantitative than observational. In one case, we did an initial study where it became clear that the depth of knowledge someone has of a certain piece of code will definitely show up in the quality of feedback they are able to offer as a reviewer. Which is to say, to get higher-quality comments, you need reviews from people who have some experience with that particular piece of software. Then, to check out that conclusion, we spoke with and observed some engineers who had submitted reviews for code already familiar to them. We also observed some engineers who had been asked to review code they had no prior experience with. That was a small study, but it left us with some definite impressions.

There also were those studies Michaela just mentioned, where we considered comment usefulness. That was based on data gathered from across all of Microsoft and then fed

into a machine-learning classifier we had built to categorize code reviews. We ended up using that to classify three million reviews of code that had been written by tens of thousands of developers and drawn from every codebase across the whole of Microsoft—meaning we are easily talking about hundreds of millions of lines of code. Obviously, the quantitative data analysis we were able to perform there was based on a substantial amount of data. The qualitative observational studies, on the other hand, were typically much smaller.

MG: We definitely had a tremendous amount of data available—essentially all the code written for Office, Windows, Windows Phone, Azure, and Visual Studio, as well as many smaller projects.

JC: We also enjoy an advantage here at Microsoft in that we have so many different product types. We look at the work people do on operating systems, as well as apps and large-scale services and small-scale services and everything in between. We are very aware of the different demands in each of these areas, and we make a point of keeping that in mind as we do our studies.

LP: In those cases where you could derive data from the use of CodeFlow, were you also able to further instrument the tool to augment your studies?

JC: One of the most interesting things to surface from instrumenting CodeFlow was just how much time people were actively spending in the review tool. That's because



TERRY COATTA

With an eye to the people outside of Microsoft that don't have your tooling, do you have any recommendations from your experience that might prove relevant?



we have found that people will often open multiple instances of the tool and then, as they get a bit of free time, do a small review here and then another small review there. So, just because you can see the tool has been open for a certain amount of time doesn't mean you can assume there has been activity for that whole time. We have the telemetry to determine just how long you were navigating around within the app. That has allowed us to determine that people, on average, spend about 20 minutes per day actively working in CodeFlow—which amounts to a significant amount of time once you multiply that by 40,000 people.

CB: From all that, we have been able to make a number of general observations we're always happy to pass along as recommendations. In fact, one suggestion I would offer to anyone looking to do something similar to what we've done in analyzing their own organization's code-review process is that, in considering what data to collect, stay as close as possible to the actual object model employed by the application itself. For example, there is almost a 1:1 correspondence between the tables in our database and the classes in the application. As a result, we didn't have to think very hard about whether to collect something or not. We just grabbed everything.

So, we ended up collecting all this raw data, and one advantage of that is, even if you don't see an immedi-

ate need for some of that data, you might find a use for it later as new studies come up. Which means you won't be faced with needing to go back and update your data-collection system to provide for that. The downside is that you will also have all this raw information on your hands that hasn't been processed for use, which means some engineer is going to have to come along later to build a metrics layer on top of all that. That will leave you with two levels of data—the analytics layer and another layer containing the raw object model data—which people can dive into later if they are looking to get their hands really dirty.

That sort of layering turned out to be a really smart move for us since we now can cater not only to the casual user who simply wants to look at metrics and reviews but also to someone who wants to dive into things.

LP: Are you saying that after you have created these tools for your research purposes, other teams will go on to use them to reflect on their own processes?

CB: Yes. In fact, we did a study a few years ago where we contacted some of the teams that were using our data to discover exactly what they were doing with it, as well as to see whether they had managed to improve the process in any way. We thought that this might be a way to find out where we needed to take our own research.

We found that some teams were using the data to generate scorecards, whereas some were using it to

discover where people were having problems understanding the code-base and then using those insights to drive their training programs. We ended up talking with at least another dozen teams, and it was interesting and surprising to learn about the different ways some of those teams had used our data.

LP: What were some of the bigger surprises?

CB: The biggest surprise for me was learning that some teams would use our tools to identify code reviews that took too long or contained only a few comments. Then they would open the code reviews based on that data, and the reviews would tell them what code had been used and what part of the code was being reviewed. They would dig into that and quickly determine, “Oh, it looks like people are having a tough time reviewing code that uses this particular API.” That’s how they would determine that their next training session ought to be devoted to that API.

TC: Have you developed any metrics for essentially grading the quality of code reviews?

CB: Not as such, but I know some teams have built live dashboards around this data. Some development teams have mounted a massive TV monitor right on the wall where metrics like “Time since last bug” or “Time to delivery of next release” can be displayed. One team told us they also put code-review data up on their scoreboard so people could see how many code reviews are on backlog or how much time on average is required to complete a code review. From what they told us, it seems that having that data up on a real-time dashboard, mission-control style, has proved to be quite motivating.

Delivering a new set of capabilities for managing and improving Microsoft’s code-review process was the primary goal right from the start. In the course of accomplishing that, much was also learned about certain general code-review principles—guidelines that might also be applied to beneficial effect elsewhere. In fact, subsequent research has offered surprising evidence of just how similar the impact can be when many of these principles

are followed at companies other than Microsoft—or, for that matter, by open source projects.

LP: Looking back to when you first started this project, what would you say came up most whenever you questioned people about their primary motives for doing code reviews?

MG: We did a survey where we asked people to rank their reasons. What came out of that tended to be fairly obvious: improving the code, finding defects, knowledge transfer ... that sort of thing. But then, when we launched this other study to categorize the comments that had been left in the actual code, we found they only rarely aligned with those stated motivations.

LP: Interesting. What did those comments chiefly focus on?

MG: There were a lot of comments about the documentation, of course. And you would see some remarks having to do with alternative solutions. There also were comments about validation, which admittedly leaned in the direction of bug resolution since people would say, “You know, if this particular corner case went away, you would be able to eliminate some of these problems.” People also had things to say about API usage—and best practices as well. On the whole, I’d say these sorts of comments far outweighed any that focused on specific defects.

JC: To Michaela’s point regarding this mismatch between expectations and reality, despite the fact that people consistently said their primary reason for doing code reviews was to discover bugs in code, only 15% of the comments we found in code actually related to bugs. For example, we would find comments about control-flow issues or use of the wrong API—or even use of the right API but in the wrong way. On the other hand, at least half of the comments were about maintainability. So, it would seem that for the reviewers themselves, identifying maintainability issues proves to be more of a priority than uncovering bugs.

LP: Now that your work has been out there for a number of years, what sort of impact have you seen on code-review policies and practices across all the different development teams?

JC: One of our top goals was to reduce the amount of time required to do a code review on average. We looked to discover where it was that people seemed to be spending an inordinate amount of time, and that is what led to the creation of a reviewer recommender. It’s such a simple thing, really, but it can be hard to find people with the right experience if you are part of a large team. Having an automated system to identify those engineers who have some familiarity with the file where some changes have been made can help cut down on the time required to get those changes reviewed.

Something else we’ve done, quite recently, is to give the developers a way to explain what it was they were trying to accomplish. This is because a complaint we commonly hear from reviewers is that it can be quite challenging to understand the reasoning behind a code change. Which is to say they would like some way to get into the mindset of the person who made that change so they can better understand whether it actually makes any sense or not.

One way of dealing with this is to show more than just the isolated section of code where a change has been made. Instead, we show entire files so reviewers can get a better sense of the code around each change. We also wanted to provide some means for the author of a change to offer additional information so reviewers could better understand their reasoning. Toward that end, our system now lets authors put tags on files and regions to indicate which files are at the heart of a change and so should probably be given particular attention. For example, the tags can be used to quickly indicate which changes have been made to test cases as opposed to the product codes. Or they can be used to call out certain files or changes with potential security implications.

LP: Do you have any other new capabilities in the works?

JC: The fundamental underlying factor we’re trying to address is the size of code reviews since that affects both the time required to produce a review and the usefulness of the comments that come out of it. It’s a difficult problem to address because some of the issues are cultural in nature, and some relate

to workflow. Still, there are times when two unrelated concerns end up getting crammed into a single review, so we are hoping we will be able to untangle some reviews by automatically splitting those concerns into two smaller reviews. On average, that ought to lead to better turnaround times, as well as better outcomes.

LP: Have you taken any steps to get development teams to focus their code-review time on correctness and content versus style? Have any tool changes or process changes been implemented toward that end?

JC: We haven't done a proper study of that, but there is a team here that's done something along those lines. This is something that had to do with some factoring changes they considered to be low-risk—such as the renaming of methods or local variables. For example, this might involve putting a special tag on a review to say, “We don't really need to have two people look at this. One is enough since it's very unlikely we'll have any functionality issues here.” Modest as that might seem, it can also prove profound since it turns out there are many changes like this floating through a legacy system—*clogging* the system.

The thing to remember is that it's not just about making one change go faster, since what you're dealing with here is a pipeline of changes—meaning that any change you can redirect to a lighter-weight path is going to lower the load on your key people and get it out of the way of other changes waiting to be reviewed. That's just the sort of thing that makes for a more efficient system all the way around.

TC: With an eye to the people outside of Microsoft that don't have your tooling, do you have any recommendations from your experience that might prove relevant?

JC: I would say the one thing to recognize is that comments related to maintainability are primarily what you are going to get out of the code-review process. Contrary to popular opinion, locating bugs is not the primary outcome. The other important thing to bear in mind is that the smaller a review is, the better it's going to be. In our case, we've found that if a review contains more than 20 files, it's too big already. In fact, from our

study of all the data at our disposal, we've concluded that for more than 20 files the density and usefulness of comments degrades significantly. This is actually more a rule of thumb than a precise limit, but it is useful to keep in mind.

Also, if your organization has data from past reviews, I would suggest investing in a recommender system that can help make some of the administrative steps a little less tedious. You can even use these systems to automatically address some of your maintainability issues, which is something we're starting to get into these days. That is, you can imagine that some of these maintainability issues are essentially things that might be autodiscovered and flagged, which means you then don't have to expend any human resources to get this accomplished.

Another thing, as we just discussed, is the idea that two signoffs on every change might be too many. If you look at the distribution of comments made by either the first or the second reviewer, you'll find that your first reviewer typically discovers the most egregious problems. In many cases, waiting for a second reviewer to corroborate those findings before allowing the commit into the main source tree might be less efficient.

MG: My biggest takeaway from the survey is to always make the burden of code reviews just as small as you possibly can. Part of that comes down to having a good code-review process that enables and encourages comments that can be easily reviewed.

Another important consideration has to do with supporting the reviewers themselves by giving them advance notice about any reviews that might be coming up and giving them enough context so they will be able to dive right into a review without having to figure all that out for themselves. Doing what you can to reduce the size of reviews can also be helpful. But I think what is really important is to make the reviews just as uncomplicated as possible, since, otherwise, you may end up with reviewers who have no clue about where even to start.

Also, organizations need to show they recognize the value of code reviews since there's no question that they take away from the time developers could

otherwise be using to create code. If developers are rewarded only for adding functionality, that's going to end up crippling the code-review process, which in turn will almost certainly have an adverse effect on the maintainability of the code that's generated.

CB: One thing I would like to add is that the code-review process we now have at Microsoft has more or less grown organically—through experimentation—from the grassroots. I mention this only because I think it might also work well for smaller companies, instead of having some process that's mandated from the top down.

Also, each product group at Microsoft does code reviews a little differently, with each group using its own set of policies that have essentially come together organically. While this probably won't come as a groundbreaking revelation, it can definitely be said that there is no one-size-fits-all solution for code reviews. This only serves to reinforce the importance of being willing to let your approach evolve organically such that it ends up fitting in with your work processes with the least amount of friction while putting the lightest burden possible on your developers.

Another important point is something Michaela talked about earlier, which is that treating code review as a first-class citizen—just as many companies are likely to treat testing—is probably the best way to get the most bang for your buck. If, instead, it becomes something you are just expected to do, like flossing your teeth daily, then you'll find people aren't going to embrace it. But if you say this is important and so will be tracked and evaluated, then people are likely to respond to that. Certainly, that's how it has worked out here.

And then the other thing I would add is that it's instructive to think in some depth about what it is you're really looking to get out of code reviews. Then, of course, you should also think about how you can go about measuring that. To the degree that you can track those metrics and set targets, you're always going to achieve more. ▣



**You have to finish strong,
every time.**

BY KATE MATSUDAIRA

The Importance of a Great Finish

HAVE YOU EVER felt super excited about the start of a project, but as time went on your excitement (and motivation) started to wane?

Unfortunately, not all work is created equal. It is often the work through the bulk of a project that is not remembered or recognized.

The work that tends to be remembered from any given project is the work that happened last. It is the final step that most people will think of, because it happened most recently. This is especially true of the people who have the most power over your promotions and future opportunities, who don't see what you accomplish day to day. They just see the results.

I have worked with hundreds of engineers during my career, and I have seen this happen over and over again. Projects start with a bang and end with a whimper, and the people on the team are surprised when their hard work isn't viewed as positively as they think it should be.

How can you make sure you are recognized as a valuable member of your team, whose work is seen as critical to the team's success?

You have to finish strong, every time. Here is how to keep your momentum up and make the right moves to be a visible contributor to the final success of every project.

The Psychology of a Strong Finish

Humans tend to remember the ending of something far more clearly than any other part—even if other parts were more significant or important. Why is that?

Essentially, our brains can process only so much. We take in so much in-



formation every day that it is impossible to remember everything completely. As a result, our brains have to give priority to certain pieces of information over others.

This means we usually have the clearest recall for things that were associated with strong emotions and things that happened most recently. This is known as the Peak-End Rule (https://en.wikipedia.org/wiki/Peak-end_rule).

This applies to all areas of our lives. It's why you should always stay at the nicest hotel at the very end of your vacation—it's the one you will remember most when you think about that trip.

At work, your performance reviews are usually weighted toward the work you did most recently. Why? Because it is freshest in your manager's mind.

So, when you are working on a project, think about how it will be perceived by your leadership, keeping in mind the importance of the end result.

While you might remember the long hours you worked to build a new feature one night, your boss may have a different perspective. If, for example, that feature you built had bugs that had to be fixed at the last minute, or operational problems that generated negative attention, that's what the boss will remember more than the many hours you put in.

Therefore, if you want to make a big impact at work, you need to take advantage of the Peak-End Rule by ensuring

every project you work on has a successful, strong conclusion. If you are in a position to present the project to your leadership, make sure they see how your hard work applies to their goals and the things that are most important to them.

Why So Few People Finish Strong

Starting work on a new project or goal is usually an exciting time. In the beginning, there is a lot of momentum. You are excited to tackle a big problem, and energy is high in meetings. The first 80% of a project is all about building up; there is a thrill in creating something new.

By the end, though, energy is low. You push to get things done by a deadline, and you procrastinate on the boring stuff that still has to get done, like extra testing, polishing, documentation, and boundary cases you missed earlier.

The less elegant work is not as much fun to work on, so people don't really work on it. Plus, there is very little recognition for this kind of work.

Our brains are resistant to working on tasks that don't seem to offer some kind of reward. They seem too small, or too tedious. It can be mentally, and even physically, taxing to spend time on a job that you do not want to do or know you will not be directly rewarded for.

These mundane tasks, when done correctly, make the problems they are solving invisible. You would only ever notice if that work *had not* been

done because you would have a buggy, clunky product. When the details are done right, it looks seamless and you forget about how much work went into finishing.

Unfortunately, letting these boring details go is akin to undoing all the exciting work you already put in on the project. If you want the beautiful thing you built to stay standing, you have to finish it out right.

How To Make a Great Finish a Priority

The next time you are hard at work on a big project, make sure you allot time and energy for a strong finish. Set aside time in your project plans for the boring detail work; that way, it won't catch you by surprise. Make it seem just as important as all the rest of the work you do—because it is.

As you get to work on your next big goal, keep in mind these three ways to make sure you finish strong and make the biggest possible impact with your work.

1. Think Big Picture

When you are working on a project, always keep the bigger-picture goals in mind. What is the overall impact of this project on your company? What does your manager see as your team's biggest goal?

You may remember an amazing solution you came up with early in the project, but your manager or executive team—who were not in the trenches with you every day, and who instead are

making judgments based on limited information channeled up to them—have only so many details to go on. One of the biggest factors they use to determine success or failure is how a project wrapped up.

Did the project miss the deadline? Were tons of bugs reported right after the launch? Did your team have to explain to the boss why x, y, z didn't work?

Whenever you are choosing what to work on or where to apply your best efforts, take a moment to step back. Zoom out from your own preferences and remind yourself what the bigger-picture goals are. Where will your work mean the most to the people in charge?

If you are not sure, ask. Go to your manager and say, "I am thinking about working on A or B next. Which is most important? Or is there another place I should be focusing?"

It may seem counterintuitive—you might worry that asking about priorities might make you look stupid—but checking in with your manager is actually really smart. Not only do you ensure you are working on the right priorities, but it is also a great way to keep your manager up to date about your contributions and show that you are focused on the big-picture goals that matter most to managers.

2. Make the Unglamorous a Priority

When people lose momentum on a project, it is usually right around the time the shiniest, most interesting work gets completed. Don't let this happen to you.

One way to approach the boring details of a project—bug fixes, use cases, among others—is to reframe them in your mind. Tell yourself that this is actually some of the most important work you'll do because you will be helping the outcome to be as perfect as it can be.

Look for opportunities to make these tasks more challenging or interesting. Instead of slogging through boring details, try to bring new energy to them.

Although this work may not be all that visible, it is still important. Remember that a rising tide lifts all boats. Even if you do not get the glory for fixing small final details, your work will make the overall project more successful in the end, and you will have been part of



When people lose momentum on a project, it is usually right around the time the shiniest, most interesting work gets completed. Don't let this happen to you.



a team that executed well. In time, you will become known for always being on the team that succeeds.

3. Channel Your Ability To Keep Going

Have you ever heard a story about a mother who lifted a car to save her child? What about marathon runners who talk about having "nothing left" but go on to finish the race?

We all have extreme strength within us; we just don't usually see it because it comes out only in extreme circumstances.

In normal life, your brain communicates with your body about what you can and cannot do. Your brain says, "Hey, that will hurt," and your body slows down. In most situations, this serves you well. You cannot actually lift a car every day, and you would not want to try.

However, the ability to power through challenges that you normally don't face is in your toolkit. Remember that the next time you are nearing the end of a long, exhausting project. You can do it. You might feel like you have nothing left, but the end is the most important part—so, draw on your resources and make the last steps count.

If you work hard on a project, your hours will not be worth as much if you are not seen delivering a strong finish. So, make all that work worth it, and follow through on every single step. Dot your i's, cross your t's, and deliver amazing results that will take you far in your career. 

Related articles on queue.acm.org

The Small Batches Principle

Thomas A. Limoncelli

<https://queue.acm.org/detail.cfm?id=2945077>

Kode Vicious Unleashed

George Neville-Neil

<https://queue.acm.org/detail.cfm?id=1046939>

Culture Surprises in Remote Software Development Teams

Judith S. Olson, Gary M. Olson

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

Kate Matsudaira (katemats.com) is an experienced technology leader. She has worked at Microsoft and Amazon and successful startups before starting her own company, Popforms, which was acquired by Safari Books.

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

Innovations like domain-specific hardware, enhanced security, open instruction sets, and agile chip development will lead the way.

BY JOHN L. HENNESSY AND DAVID A. PATTERSON

A New Golden Age for Computer Architecture

WE BEGAN OUR Turing Lecture June 4, 2018¹¹ with a review of computer architecture since the 1960s. In addition to that review, here, we highlight current challenges and identify future opportunities, projecting another golden age for the field of computer architecture in the next decade, much like the 1980s when we did the research that led to our award, delivering gains in cost, energy, and security, as well as performance.

“Those who cannot remember the past are condemned to repeat it.” —George Santayana, 1905

Software talks to hardware through a vocabulary called an instruction set architecture (ISA). By the early 1960s, IBM had four incompatible lines of computers, each with its own ISA, software stack, I/O system, and market niche—targeting small business, large business, scientific, and real time, respectively. IBM

engineers, including ACM A.M. Turing Award laureate Fred Brooks, Jr., thought they could create a single ISA that would efficiently unify all four of these ISA bases.

They needed a technical solution for how computers as inexpensive as

» key insights

- Software advances can inspire architecture innovation.
- Elevating the hardware/software interface creates opportunities for architecture innovation.
- The marketplace ultimately settles architecture debates.





those with 8-bit data paths and as fast as those with 64-bit data paths could share a single ISA. The data paths are the “brawn” of the processor in that they perform the arithmetic but are relatively easy to “widen” or “narrow.” The greatest challenge for computer designers then and now is the “brains” of the processor—the control hardware. Inspired by software programming, computing pioneer and Turing laureate Maurice Wilkes proposed how to simplify control. Control was specified as a two-dimensional array he called a “control store.” Each column of the array corresponded to

one control line, each row was a microinstruction, and writing microinstructions was called microprogramming.³⁹ A control store contains an ISA interpreter written using microinstructions, so execution of a conventional instruction takes several microinstructions. The control store was implemented through memory, which was much less costly than logic gates.

The table here lists four models of the new System/360 ISA IBM announced April 7, 1964. The data paths vary by a factor of 8, memory capacity by a factor of 16, clock rate by nearly 4, performance by 50, and cost by near-

ly 6. The most expensive computers had the widest control stores because more complicated data paths used more control lines. The least-costly computers had narrower control stores due to simpler hardware but needed more microinstructions since they took more clock cycles to execute a System/360 instruction.

Facilitated by microprogramming, IBM bet the future of the company that the new ISA would revolutionize the computing industry and won the bet. IBM dominated its markets, and IBM mainframe descendants of the computer family announced 55 years

ago still bring in \$10 billion in revenue per year.

As seen repeatedly, although the marketplace is an imperfect judge of technological issues, given the close ties between architecture and commercial computers, it eventually determines the success of architecture innovations that often require significant engineering investment.

Integrated circuits, CISC, 432, 8086, IBM PC. When computers began using integrated circuits, Moore’s Law meant control stores could become much larger. Larger memories in turn allowed much more complicated ISAs. Consider that the control store of the VAX-11/780 from Digital Equipment Corp. in 1977 was 5,120 words × 96 bits, while its predecessor used only 256 words × 56 bits.

Some manufacturers chose to make microprogramming available by letting select customers add custom features they called “writable control store” (WCS). The most famous WCS computer was the Alto³⁶ Turing laureates Chuck Thacker and Butler Lampson, together with their colleagues, cre-

ated for the Xerox Palo Alto Research Center in 1973. It was indeed the first personal computer, sporting the first bit-mapped display and first Ethernet local-area network. The device controllers for the novel display and network were microprograms stored in a 4,096-word × 32-bit WCS.

Microprocessors were still in the 8-bit era in the 1970s (such as the Intel 8080) and programmed primarily in assembly language. Rival designers would add novel instructions to outdo one another, showing their advantages through assembly language examples.

Gordon Moore believed Intel’s next ISA would last the lifetime of Intel, so he hired many clever computer science Ph.D.’s and sent them to a new facility in Portland to invent the next great ISA. The 8800, as Intel originally named it, was an ambitious computer architecture project for any era, certainly the most aggressive of the 1980s. It had 32-bit capability-based addressing, object-oriented architecture, variable-bit-length instructions, and its own

operating system written in the then-new programming language Ada.

This ambitious project was alas several years late, forcing Intel to start an emergency replacement effort in Santa Clara to deliver a 16-bit microprocessor in 1979. Intel gave the new team 52 weeks to develop the new “8086” ISA and design and build the chip. Given the tight schedule, designing the ISA took only 10 person-weeks over three regular calendar weeks, essentially by extending the 8-bit registers and instruction set of the 8080 to 16 bits. The team completed the 8086 on schedule but to little fanfare when announced.

To Intel’s great fortune, IBM was developing a personal computer to compete with the Apple II and needed a 16-bit microprocessor. IBM was interested in the Motorola 68000, which had an ISA similar to the IBM 360, but it was behind IBM’s aggressive schedule. IBM switched instead to an 8-bit bus version of the 8086. When IBM announced the PC on August 12, 1981, the hope was to sell 250,000 PCs by 1986. The company instead sold 100 million worldwide, bestowing a very bright future on the emergency replacement Intel ISA.

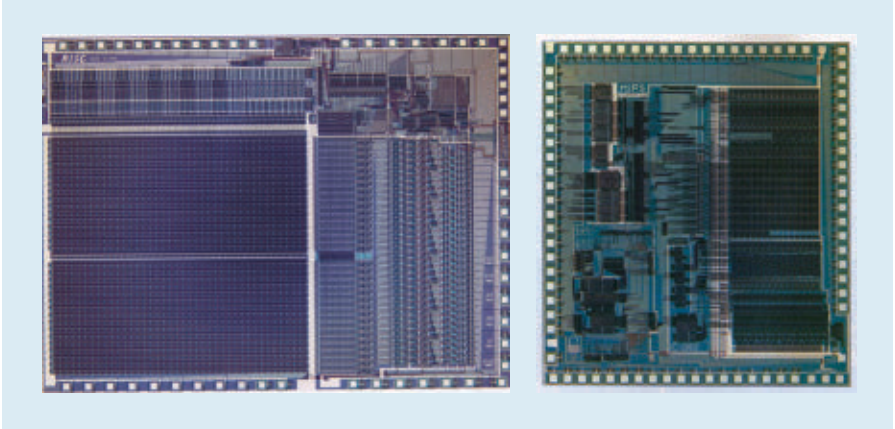
Intel’s original 8800 project was renamed iAPX-432 and finally announced in 1981, but it required several chips and had severe performance problems. It was discontinued in 1986, the year after Intel extended the 16-bit 8086 ISA in the 80386 by expanding its registers from 16 bits to 32 bits. Moore’s prediction was thus correct that the next ISA would last as long as Intel did, but the marketplace chose the emergency replacement 8086 rather than the anointed 432. As the architects of the Motorola 68000 and iAPX-432 both learned, the marketplace is rarely patient.

From complex to reduced instruction set computers. The early 1980s saw several investigations into complex instruction set computers (CISC) enabled by the big microprograms in the larger control stores. With Unix demonstrating that even operating systems could use high-level languages, the critical question became: “What instructions would compilers generate?” instead of “What assembly language would programmers use?” Significantly raising the hardware/software inter-

Features of four models of the IBM System/360 family; IPS is instructions per second.

Model	M30	M40	M50	M65
Datapath width	8 bits	16 bits	32 bits	64 bits
Control store size	4k x 50	4k x 52	2.75k x 85	2.75k x 87
Clock rate (ROM cycle time)	1.3 MHz (750 ns)	1.6 MHz (625 ns)	2 MHz (500 ns)	5 MHz (200 ns)
Memory capacity	8–64 KiB	16–256 KiB	64–512 KiB	128–1,024 KiB
Performance (commercial)	29,000 IPS	75,000 IPS	169,000 IPS	567,000 IPS
Performance (scientific)	10,200 IPS	40,000 IPS	133,000 IPS	563,000 IPS
Price (1964 \$)	\$192,000	\$216,000	\$460,000	\$1,080,000
Price (2018 \$)	\$1,560,000	\$1,760,000	\$3,720,000	\$8,720,000

Figure 1. University of California, Berkeley, RISC-I and Stanford University MIPS microprocessors.



face created an opportunity for architecture innovation.

Turing laureate John Cocke and his colleagues developed simpler ISAs and compilers for minicomputers. As an experiment, they retargeted their research compilers to use only the simple register-register operations and load-store data transfers of the IBM 360 ISA, avoiding the more complicated instructions. They found that programs ran up to three times faster using the simple subset. Emer and Clark⁶ found 20% of the VAX instructions needed 60% of the microcode and represented only 0.2% of the execution time. One author (Patterson) spent a sabbatical at DEC to help reduce bugs in VAX microcode. If microprocessor manufacturers were going to follow the CISC ISA designs of the larger computers, he thought they would need a way to repair the microcode bugs. He wrote such a paper,³¹ but the journal *Computer* rejected it. Reviewers opined that it was a terrible idea to build microprocessors with ISAs so complicated that they needed to be repaired in the field. That rejection called into question the value of CISC ISAs for microprocessors. Ironically, modern CISC microprocessors do indeed include microcode repair mechanisms, but the main result of his paper rejection was to inspire him to work on less-complex ISAs for microprocessors—reduced instruction set computers (RISC).

These observations and the shift to high-level languages led to the opportunity to switch from CISC to RISC. First, the RISC instructions were simplified so there was no need for a microcoded interpreter. The RISC instructions were typically as simple as microinstructions and could be executed directly by the hardware. Second, the fast memory, formerly used for the microcode interpreter of a CISC ISA, was repurposed to be a cache of RISC instructions. (A cache is a small, fast memory that buffers recently executed instructions, as such instructions are likely to be reused soon.) Third, register allocators based on Gregory Chaitin's graph-coloring scheme made it much easier for compilers to efficiently use registers, which benefited these register-register ISAs.³ Finally, Moore's Law meant there were enough transistors in the 1980s to include a full 32-bit



In today's post-PC era, x86 shipments have fallen almost 10% per year since the peak in 2011, while chips with RISC processors have skyrocketed to 20 billion.



datapath, along with instruction and data caches, in a single chip.

For example, Figure 1 shows the RISC-I⁸ and MIPS¹² microprocessors developed at the University of California, Berkeley, and Stanford University in 1982 and 1983, respectively, that demonstrated the benefits of RISC. These chips were eventually presented at the leading circuit conference, the IEEE International Solid-State Circuits Conference, in 1984.^{33,35} It was a remarkable moment when a few graduate students at Berkeley and Stanford could build microprocessors that were arguably superior to what industry could build.

These academic chips inspired many companies to build RISC microprocessors, which were the fastest for the next 15 years. The explanation is due to the following formula for processor performance:

$$\begin{aligned} \text{Time/Program} &= \text{Instructions} / \\ &\quad \text{Program} \times (\text{Clock cycles}) / \\ &\quad \text{Instruction} \times \text{Time} / (\text{Clock cycle}) \end{aligned}$$

DEC engineers later showed² that the more complicated CISC ISA executed about 75% of the number instructions per program as RISC (the first term), but in a similar technology CISC executed about five to six more clock cycles per instruction (the second term), making RISC microprocessors approximately 4× faster.

Such formulas were not part of computer architecture books in the 1980s, leading us to write *Computer Architecture: A Quantitative Approach*¹³ in 1989. The subtitle suggested the theme of the book: Use measurements and benchmarks to evaluate trade-offs quantitatively instead of relying more on the architect's intuition and experience, as in the past. The quantitative approach we used was also inspired by what Turing laureate Donald Knuth's book had done for algorithms.²⁰

VLIW, EPIC, Itanium. The next ISA innovation was supposed to succeed both RISC and CISC. Very long instruction word (VLIW)⁷ and its cousin, the explicitly parallel instruction computer (EPIC), the name Intel and Hewlett Packard gave to the approach, used wide instructions with multiple independent operations bundled together in each instruction. VLIW and EPIC advocates at the time believed if a single instruction could specify, say, six independent

Figure 2. Transistors per chip of Intel microprocessors vs. Moore's Law.

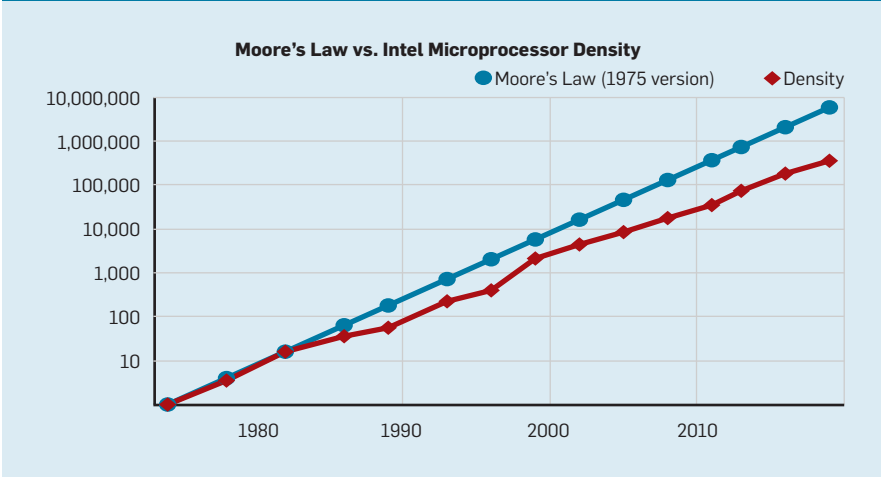
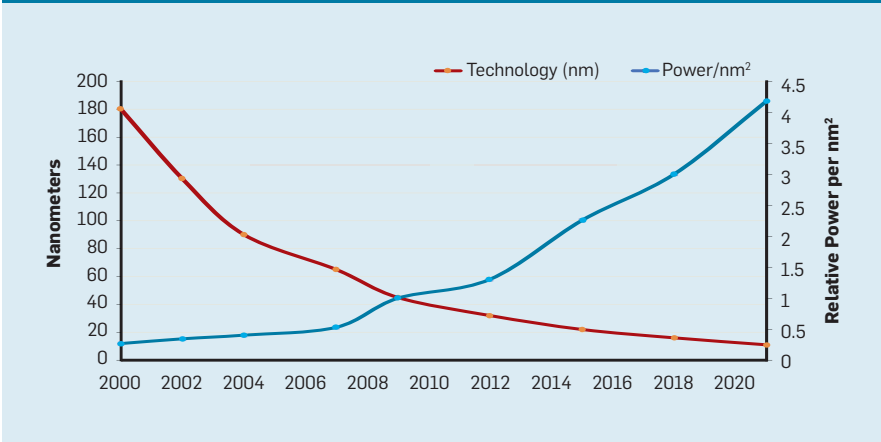


Figure 3. Transistors per chip and power per mm².



operations—two data transfers, two integer operations, and two floating point operations—and compiler technology could efficiently assign operations into the six instruction slots, the hardware could be made simpler. Like the RISC approach, VLIW and EPIC shifted work from the hardware to the compiler.

Working together, Intel and Hewlett Packard designed a 64-bit processor based on EPIC ideas to replace the 32-bit x86. High expectations were set for the first EPIC processor, called Itanium by Intel and Hewlett Packard, but the reality did not match its developers' early claims. Although the EPIC approach worked well for highly structured floating-point programs, it struggled to achieve high performance for integer programs that had less predictable cache misses or less-predictable branches. As Donald Knuth later noted:²¹ "The Itanium approach ... was supposed to be so terrific—until it turned out that the wished-for compilers were basically impossible

to write." Pundits noted delays and underperformance of Itanium and rechristened it "Itanic" after the ill-fated Titanic passenger ship. The marketplace again eventually ran out of patience, leading to a 64-bit version of the x86 as the successor to the 32-bit x86, and not Itanium.

The good news is VLIW still matches narrower applications with small programs and simpler branches and omit caches, including digital-signal processing.

RISC vs. CISC in the PC and Post-PC Eras

AMD and Intel used 500-person design teams and superior semiconductor technology to close the performance gap between x86 and RISC. Again inspired by the performance advantages of pipelining simple vs. complex instructions, the instruction decoder translated the complex x86 instructions into internal RISC-like microinstructions on the fly. AMD and Intel then pipelined the execu-

tion of the RISC microinstructions. Any ideas RISC designers were using for performance—separate instruction and data caches, second-level caches on chip, deep pipelines, and fetching and executing several instructions simultaneously—could then be incorporated into the x86. AMD and Intel shipped roughly 350 million x86 microprocessors annually at the peak of the PC era in 2011. The high volumes and low margins of the PC industry also meant lower prices than RISC computers.

Given the hundreds of millions of PCs sold worldwide each year, PC software became a giant market. Whereas software providers for the Unix marketplace would offer different software versions for the different commercial RISC ISAs—Alpha, HP-PA, MIPS, Power, and SPARC—the PC market enjoyed a single ISA, so software developers shipped "shrink wrap" software that was binary compatible with only the x86 ISA. A much larger software base, similar performance, and lower prices led the x86 to dominate both desktop computers and small-server markets by 2000.

Apple helped launch the post-PC era with the iPhone in 2007. Instead of buying microprocessors, smartphone companies built their own systems on a chip (SoC) using designs from other companies, including RISC processors from ARM. Mobile-device designers valued die area and energy efficiency as much as performance, disadvantaging CISC ISAs. Moreover, arrival of the Internet of Things vastly increased both the number of processors and the required trade-offs in die size, power, cost, and performance. This trend increased the importance of design time and cost, further disadvantaging CISC processors. In today's post-PC era, x86 shipments have fallen almost 10% per year since the peak in 2011, while chips with RISC processors have skyrocketed to 20 billion. Today, 99% of 32-bit and 64-bit processors are RISC.

Concluding this historical review, we can say the marketplace settled the RISC-CISC debate; CISC won the later stages of the PC era, but RISC is winning the post-PC era. There have been no new CISC ISAs in decades. To our surprise, the consensus on the best

ISA principles for general-purpose processors today is still RISC, 35 years after their introduction.

Current Challenges for Processor Architecture

"If a problem has no solution, it may not be a problem, but a fact—not to be solved, but to be coped with over time."

—Shimon Peres

While the previous section focused on the design of the instruction set architecture (ISA), most computer architects do not design new ISAs but implement existing ISAs in the prevailing implementation technology. Since the late 1970s, the technology of choice has been metal oxide semiconductor (MOS)-based integrated circuits, first n-type metal-oxide semiconductor (nMOS) and then complementary metal-oxide semiconductor (CMOS). The stunning rate of improvement in MOS technology—captured in Gordon Moore's predictions—has been the driving factor enabling architects to design more-aggressive methods for achieving performance for a given ISA. Moore's original prediction in 1965²⁶ called for a doubling in transistor density yearly; in 1975, he revised it, projecting a doubling every two years.²⁸ It eventually became called Moore's Law. Because transistor density grows quadratically while speed grows linearly, architects used more transistors to improve performance.

End of Moore's Law and Dennard Scaling

Although Moore's Law held for many decades (see Figure 2), it began to slow sometime around 2000 and by 2018 showed a roughly 15-fold gap between Moore's prediction and current capability, an observation Moore made in 2003 that was inevitable.²⁷ The current expectation is that the gap will continue to grow as CMOS technology approaches fundamental limits.

Accompanying Moore's Law was a projection made by Robert Dennard called "Dennard scaling,"²⁵ stating that as transistor density increased, power consumption per transistor would drop, so the power per mm² of silicon would be near constant. Since the computational capability of a mm² of silicon was increasing with each new

generation of technology, computers would become more energy efficient. Dennard scaling began to slow significantly in 2007 and faded to almost nothing by 2012 (see Figure 3).

Between 1986 and about 2002, the exploitation of instruction level parallelism (ILP) was the primary architectural method for gaining performance and, along with improvements in speed of transistors, led to an annual performance increase of approximately 50%. The end of Dennard scaling meant architects had to find more efficient ways to exploit parallelism.

To understand why increasing ILP caused greater inefficiency, consider a modern processor core like those from ARM, Intel, and AMD. Assume it has a 15-stage pipeline and can issue four instructions every clock cycle. It thus has up to 60 instructions in the pipeline at any moment in time, in-

cluding approximately 15 branches, as they represent approximately 25% of executed instructions. To keep the pipeline full, branches are predicted and code is speculatively placed into the pipeline for execution. The use of speculation is both the source of ILP performance and of inefficiency. When branch prediction is perfect, speculation improves performance yet involves little added energy cost—it can even save energy—but when it "mispredicts" branches, the processor must throw away the incorrectly speculated instructions, and their computational work and energy are wasted. The internal state of the processor must also be restored to the state that existed before the mispredicted branch, expending additional time and energy.

To see how challenging such a design is, consider the difficulty of correctly

Figure 4. Wasted instructions as a percentage of all instructions completed on an Intel Core i7 for a variety of SPEC integer benchmarks.

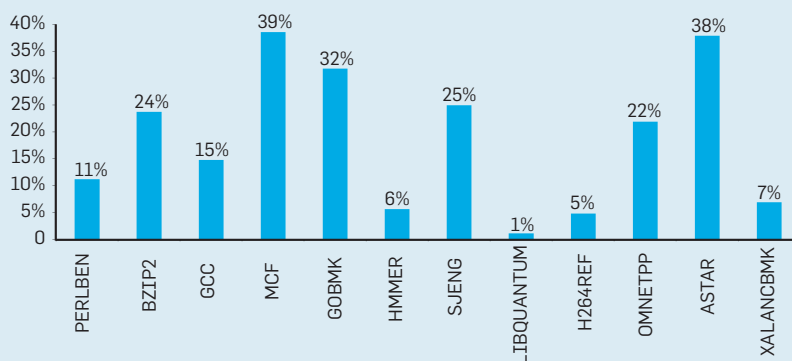
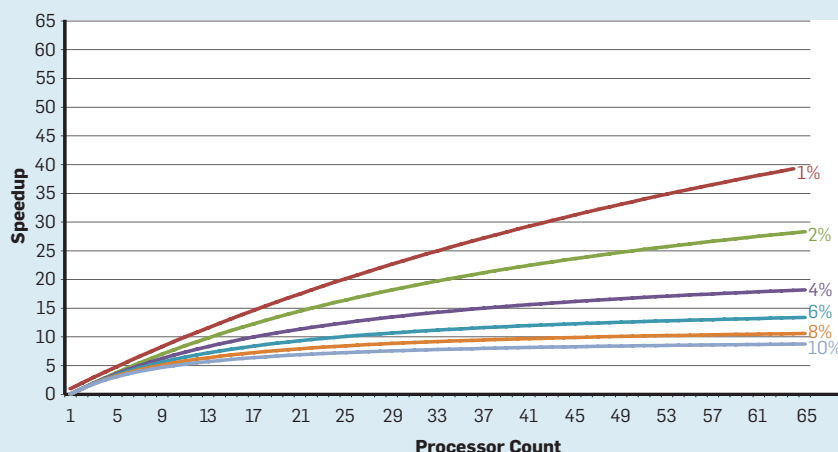


Figure 5. Effect of Amdahl's Law on speedup as a fraction of clock cycle time in serial mode.



predicting the outcome of 15 branches. If a processor architect wants to limit wasted work to only 10% of the time, the processor must predict each branch correctly 99.3% of the time. Few general-purpose programs have branches that can be predicted so accurately.

To appreciate how this wasted work adds up, consider the data in Figure 4, showing the fraction of instructions that are effectively executed but turn out to be wasted because the processor speculated incorrectly. On average, 19% of the instructions are wasted for these benchmarks on an Intel Core i7. The amount of wasted energy is greater, however, since the processor must use additional energy to restore the state when it speculates incorrectly. Measurements like these led many to conclude architects needed a differ-

ent approach to achieve performance improvements. The multicore era was thus born.

Multicore shifted responsibility for identifying parallelism and deciding how to exploit it to the programmer and to the language system. Multicore does not resolve the challenge of energy-efficient computation that was exacerbated by the end of Dennard scaling. Each active core burns power whether or not it contributes effectively to the computation. A primary hurdle is an old observation, called Amdahl's Law, stating that the speedup from a parallel computer is limited by the portion of a computation that is sequential. To appreciate the importance of this observation, consider Figure 5, showing how much faster an application runs with up to 64 cores compared to

a single core, assuming different portions of serial execution, where only one processor is active. For example, when only 1% of the time is serial, the speedup for a 64-processor configuration is about 35. Unfortunately, the power needed is proportional to 64 processors, so approximately 45% of the energy is wasted.

Real programs have more complex structures of course, with portions that allow varying numbers of processors to be used at any given moment in time. Nonetheless, the need to communicate and synchronize periodically means most applications have some portions that can effectively use only a fraction of the processors. Although Amdahl's Law is more than 50 years old, it remains a difficult hurdle.

With the end of Dennard scaling, increasing the number of cores on a chip meant power is also increasing at nearly the same rate. Unfortunately, the power that goes into a processor must also be removed as heat. Multicore processors are thus limited by the thermal dissipation power (TDP), or average amount of power the package and cooling system can remove. Although some high-end data centers may use more advanced packages and cooling technology, no computer users would want to put a small heat exchanger on their desks or wear a radiator on their backs to cool their cell-phones. The limit of TDP led directly to the era of "dark silicon," whereby processors would slow on the clock rate and turn off idle cores to prevent overheating. Another way to view this approach is that some chips can reallocate their precious power from the idle cores to the active ones.

An era without Dennard scaling, along with reduced Moore's Law and Amdahl's Law in full effect means inefficiency limits improvement in performance to only a few percent per year (see Figure 6). Achieving higher rates of performance improvement—as was seen in the 1980s and 1990s—will require new architectural approaches that use the integrated-circuit capability much more efficiently. We will return to what approaches might work after discussing another major shortcoming of modern computers—their support, or lack thereof, for computer security.

Figure 6. Growth of computer performance using integer programs (SPECintCPU).

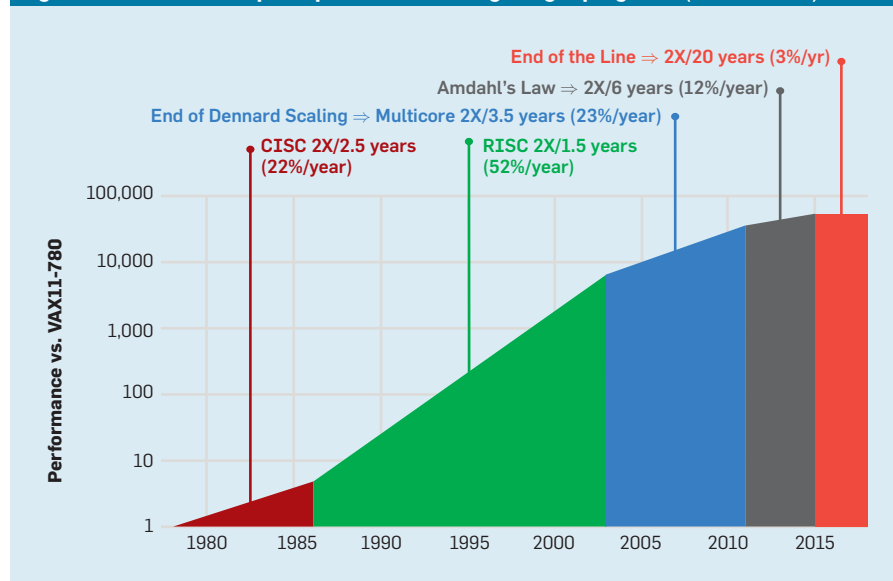
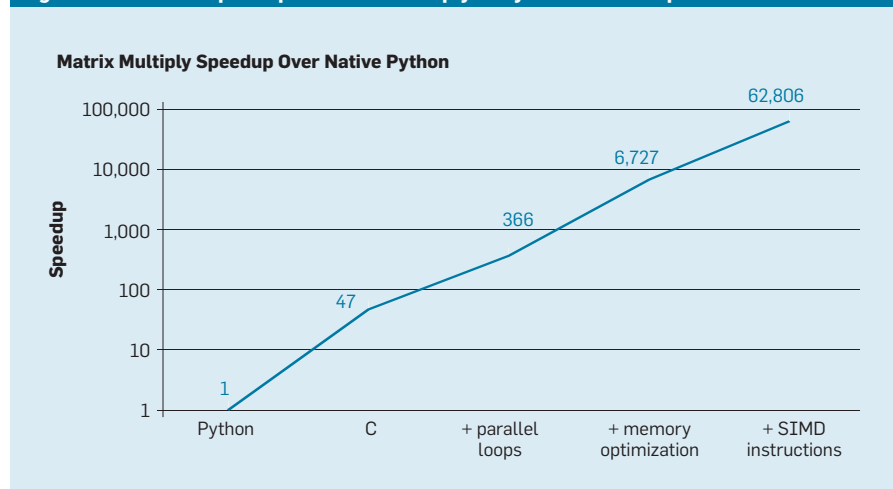


Figure 7. Potential speedup of matrix multiply in Python for four optimizations.



Overlooked Security

In the 1970s, processor architects focused significant attention on enhancing computer security with concepts ranging from protection rings to capabilities. It was well understood by these architects that most bugs would be in software, but they believed architectural support could help. These features were largely unused by operating systems that were deliberately focused on supposedly benign environments (such as personal computers), and the features involved significant overhead then, so were eliminated. In the software community, many thought formal verification and techniques like microkernels would provide effective mechanisms for building highly secure software. Unfortunately, the scale of our collective software systems and the drive for performance meant such techniques could not keep up with processor performance. The result is large software systems continue to have many security flaws, with the effect amplified due to the vast and increasing amount of personal information online and the use of cloud-based computing, which shares physical hardware among potential adversaries.

Although computer architects and others were perhaps slow to realize the growing importance of security, they began to include hardware support for virtual machines and encryption. Unfortunately, speculation introduced an unknown but significant security flaw into many processors. In particular, the Meltdown and Spectre security flaws led to new vulnerabilities that exploit vulnerabilities in the microarchitecture, allowing leakage of protected information at a high rate.¹⁴ Both Meltdown and Spectre use so-called side-channel attacks whereby information is leaked by observing the time taken for a task and converting information invisible at the ISA level into a timing visible attribute. In 2018, researchers showed how to exploit one of the Spectre variants to leak information over a network without the attacker loading code onto the target processor.³⁴ Although this attack, called NetSpectre, leaks information slowly, the fact that it allows any machine on the same local-area network (or within the same cluster in a cloud)



The end of Dennard scaling meant architects had to find more efficient ways to exploit parallelism.



to be attacked creates many new vulnerabilities. Two more vulnerabilities in the virtual-machine architecture were subsequently reported.^{37,38} One of them, called Foreshadow, allows penetration of the Intel SGX security mechanisms designed to protect the highest risk data (such as encryption keys). New vulnerabilities are being discovered monthly.

Side-channel attacks are not new, but in most earlier cases, a software flaw allowed the attack to succeed. In the Meltdown, Spectre, and other attacks, it is a flaw in the hardware implementation that exposes protected information. There is a fundamental difficulty in how processor architects define what is a correct implementation of an ISA because the standard definition says nothing about the performance effects of executing an instruction sequence, only about the ISA-visible architectural state of the execution. Architects need to rethink their definition of a correct implementation of an ISA to prevent such security flaws. At the same time, they should be rethinking the attention they pay computer security and how architects can work with software designers to implement more-secure systems. Architects (and everyone else) depend too much on more information systems to willingly allow security to be treated as anything less than a first-class design concern.

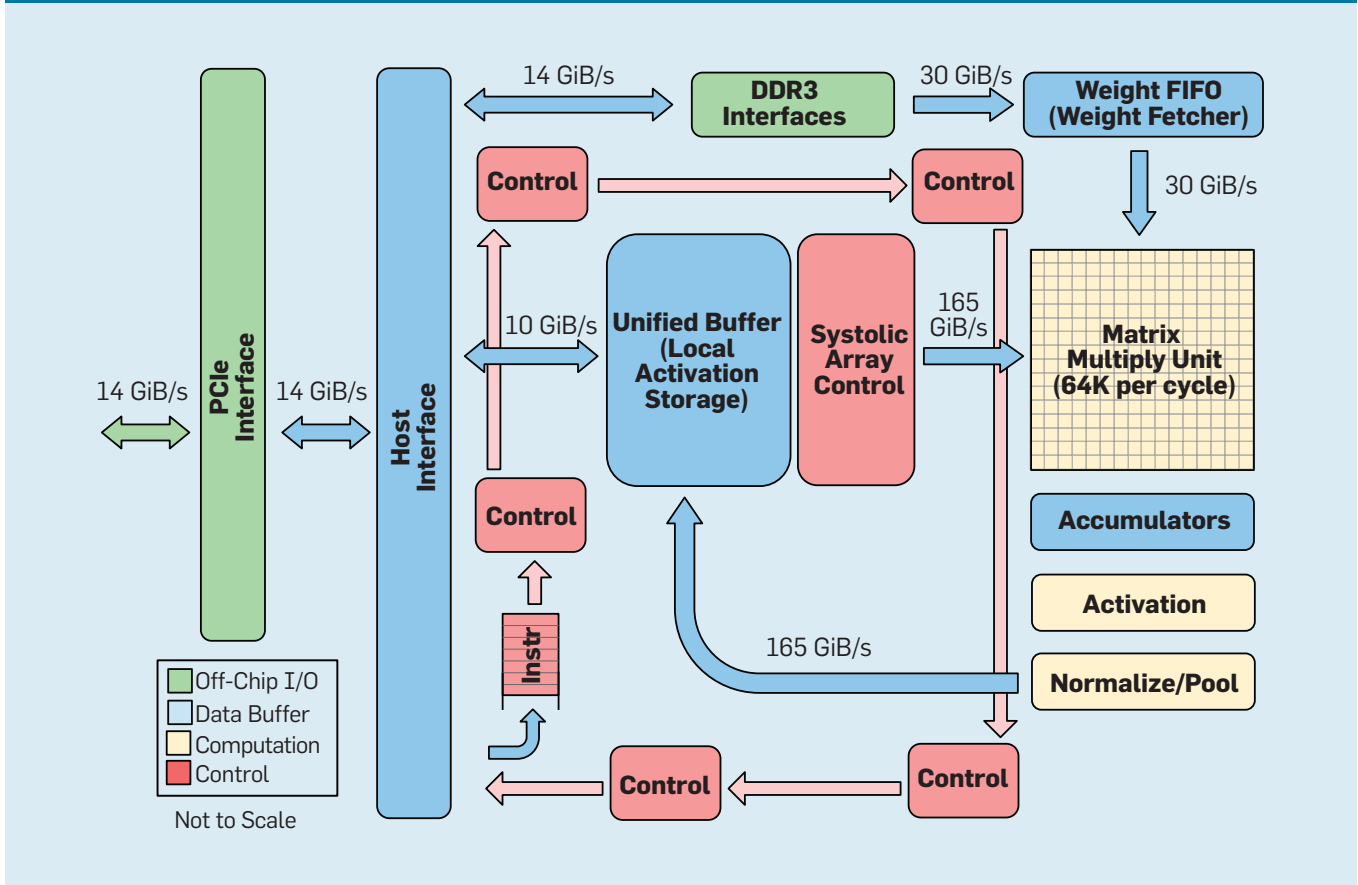
Future Opportunities in Computer Architecture

“What we have before us are some breathtaking opportunities disguised as insoluble problems.” —John Gardner, 1965

Inherent inefficiencies in general-purpose processors, whether from ILP techniques or multicore, combined with the end of Dennard scaling and Moore’s Law, make it highly unlikely, in our view, that processor architects and designers can sustain significant rates of performance improvements in general-purpose processors. Given the importance of improving performance to enable new software capabilities, we must ask: What other approaches might be promising?

There are two clear opportunities, as well as a third created by combining the two. First, existing techniques for building software make extensive use of high-

Figure 8. Functional organization of Google Tensor Processing Unit (TPU v1).



level languages with dynamic typing and storage management. Unfortunately, such languages are typically interpreted and execute very inefficiently. Leiserson et al.²⁴ used a small example—performing matrix multiply—to illustrate this inefficiency. As in Figure 7, simply rewriting the code in C from Python—a typical high-level, dynamically typed language—increases performance 47-fold. Using parallel loops running on many cores yields a factor of approximately 7. Optimizing the memory layout to exploit caches yields a factor of 20, and a final factor of 9 comes from using the hardware extensions for doing single instruction multiple data (SIMD) parallelism operations that are able to perform 16 32-bit operations per instruction. All told, the final, highly optimized version runs more than 62,000× faster on a multicore Intel processor compared to the original Python version. This is of course a small example, one might expect programmers to use an optimized library for. Although it exaggerates the usual performance gap, there are likely many programs for which factors of 100 to 1,000 could be achieved.

An interesting research direction concerns whether some of the performance gap can be closed with new compiler technology, possibly assisted by architectural enhancements. Although the challenges in efficiently translating and implementing high-level scripting languages like Python are difficult, the potential gain is enormous. Achieving even 25% of the potential gain could result in Python programs running tens to hundreds of times faster. This simple example illustrates how great the gap is between modern languages emphasizing programmer productivity and traditional approaches emphasizing performance.

Domain-specific architectures. A more hardware-centric approach is to design architectures tailored to a specific problem domain and offer significant performance (and efficiency) gains for that domain, hence, the name “domain-specific architectures” (DSAs), a class of processors tailored for a specific domain—programmable and often Turing-complete but tailored to a specific class of applications. In this sense, they differ from

application-specific integrated circuits (ASICs) that are often used for a single function with code that rarely changes. DSAs are often called accelerators, since they accelerate some of an application when compared to executing the entire application on a general-purpose CPU. Moreover, DSAs can achieve better performance because they are more closely tailored to the needs of the application; examples of DSAs include graphics processing units (GPUs), neural network processors used for deep learning, and processors for software-defined networks (SDNs). DSAs can achieve higher performance and greater energy efficiency for four main reasons:

First and most important, DSAs exploit a more efficient form of parallelism for the specific domain. For example, single-instruction multiple data parallelism (SIMD), is more efficient than multiple instruction multiple data (MIMD) because it needs to fetch only one instruction stream and processing units operate in lockstep.⁹ Although SIMD is less flexible than MIMD, it is a good match for many

DSAs. DSAs may also use VLIW approaches to ILP rather than speculative out-of-order mechanisms. As mentioned earlier, VLIW processors are a poor match for general-purpose code¹⁵ but for limited domains can be much more efficient, since the control mechanisms are simpler. In particular, most high-end general-purpose processors are out-of-order superscalars that require complex control logic for both instruction initiation and instruction completion. In contrast, VLIWs perform the necessary analysis and scheduling at compile-time, which can work well for an explicitly parallel program.

Second, DSAs can make more effective use of the memory hierarchy. Memory accesses have become much more costly than arithmetic computations, as noted by Horowitz.¹⁶ For example, accessing a block in a 32-kilobyte cache involves an energy cost approximately $200\times$ higher than a 32-bit integer add. This enormous differential makes optimizing memory accesses critical to achieving high-energy efficiency. General-purpose processors run code in which memory accesses typically exhibit spatial and temporal locality but are otherwise not very predictable at compile time. CPUs thus use multilevel caches to increase bandwidth and hide the latency in relatively slow, off-chip DRAMs. These multilevel caches often consume approximately half the energy of the processor but avoid almost all accesses to the off-chip DRAMs that require approximately $10\times$ the energy of a last-level cache access.

Caches have two notable disadvantages:

When datasets are very large. Caches simply do not work well when datasets are very large and also have low temporal or spatial locality; and

When caches work well. When caches work well, the locality is very high, meaning, by definition, most of the cache is idle most of the time.

In applications where the memory-access patterns are well defined and discoverable at compile time, which is true of typical DSLs, programmers and compilers can optimize the use of the memory better than can dynamically allocated caches. DSAs thus usually use a hierarchy of memories with movement controlled explicitly by the software, similar to how vector pro-

cessors operate. For suitable applications, user-controlled memories can use much less energy than caches.

Third, DSAs can use less precision when it is adequate. General-purpose CPUs usually support 32- and 64-bit integer and floating-point (FP) data. For many applications in machine learning and graphics, this is more accuracy than is needed. For example, in deep neural networks (DNNs), inference regularly uses 4-, 8-, or 16-bit integers, improving both data and computational throughput. Likewise, for DNN training applications, FP is useful, but 32 bits is enough and 16 bits often works.

Finally, DSAs benefit from targeting programs written in domain-specific languages (DSLs) that expose more parallelism, improve the structure and representation of memory access, and make it easier to map the application efficiently to a domain-specific processor.

Domain-Specific Languages

DSAs require targeting of high-level operations to the architecture, but trying to extract such structure and information from a general-purpose language like Python, Java, C, or Fortran is simply too difficult. Domain specific languages (DSLs) enable this process and make it possible to program DSAs efficiently. For example, DSLs can make vector, dense matrix, and sparse matrix operations explicit, enabling the DSL compiler to map the operations

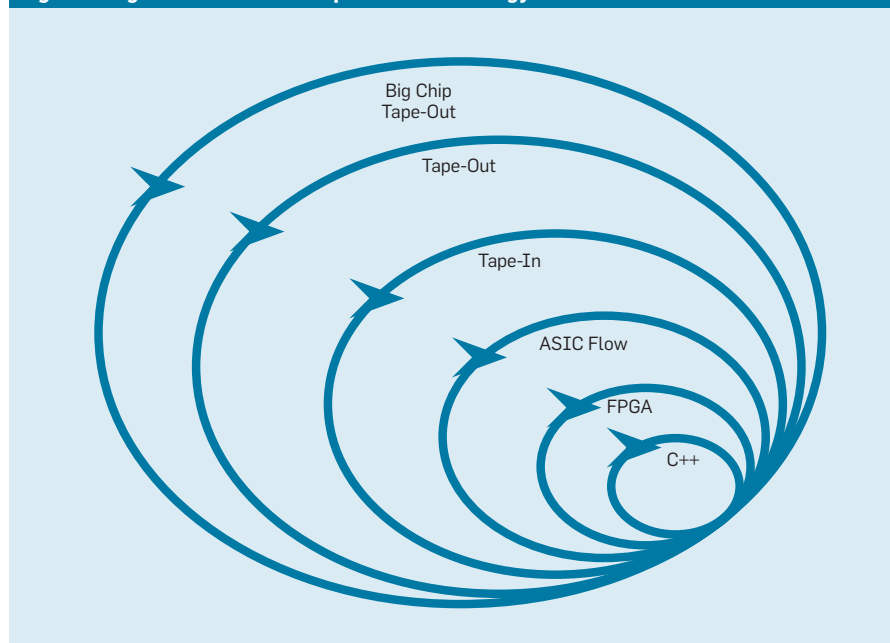
to the processor efficiently. Examples of DSLs include Matlab, a language for operating on matrices, TensorFlow, a dataflow language used for programming DNNs, P4, a language for programming SDNs, and Halide, a language for image processing specifying high-level transformations.

The challenge when using DSLs is how to retain enough architecture independence that software written in a DSL can be ported to different architectures while also achieving high efficiency in mapping the software to the underlying DSA. For example, the XLA system translates Tensorflow to heterogeneous processors that use Nvidia GPUs or Tensor Processor Units (TPUs).⁴⁰ Balancing portability among DSAs along with efficiency is an interesting research challenge for language designers, compiler creators, and DSA architects.

Example DSA: TPU v1. As an example DSA, consider the Google TPU v1, which was designed to accelerate neural net inference.^{17,18} The TPU has been in production since 2015 and powers applications ranging from search queries to language translation to image recognition to AlphaGo and AlphaZero, the DeepMind programs for playing Go and Chess. The goal was to improve the performance and energy efficiency of deep neural net inference by a factor of 10.

As shown in Figure 8, the TPU organization is radically different from a

Figure 9. Agile hardware development methodology.



general-purpose processor. The main computational unit is a matrix unit, a systolic array²² structure that provides 256×256 multiply-accumulates every clock cycle. The combination of 8-bit precision, highly efficient systolic structure, SIMD control, and dedication of significant chip area to this function means the number of multiply-accumulates per clock cycle is approximately $100\times$ what a general-purpose single-core CPU can sustain. Rather than caches, the TPU uses a local memory of 24 megabytes, approximately double a 2015 general-purpose CPU with the same power dissipation. Finally, both the activation memory and the weight memory (including a FIFO structure that holds weights) are linked through user-controlled high-bandwidth memory channels. Using a weighted arithmetic mean based on six common inference problems in Google data centers, the TPU is $29\times$ faster than a general-purpose CPU. Since the TPU requires less than half the power, it has an energy efficiency for this workload that is more than $80\times$ better than a general-purpose CPU.

Summary

We have considered two different approaches to improve program performance by improving efficiency in the use of hardware technology: First, by improving the performance of modern high-level languages that are typically interpreted; and second, by building domain-specific architectures that greatly improve performance and efficiency compared to general-purpose CPUs. DSLs are another example of how to improve the hardware/software interface that enables architecture innovations like DSAs. Achieving significant gains through such approaches will require a vertically integrated design team that understands applications, domain-specific languages and related compiler technology, computer architecture and organization, and the underlying implementation technology. The need to vertically integrate and make design decisions across levels of abstraction was characteristic of much of the early work in computing before the industry became horizontally structured. In this new era, vertical integration has become more important, and teams that can ex-

amine and make complex trade-offs and optimizations will be advantaged.

This opportunity has already led to a surge of architecture innovation, attracting many competing architectural philosophies:

GPUs. Nvidia GPUs use many cores, each with large register files, many hardware threads, and caches;⁴

TPUs. Google TPUs rely on large two-dimensional systolic multipliers and software-controlled on-chip memories;¹⁷

FPGAs. Microsoft deploys field programmable gate arrays (FPGAs) in its data centers it tailors to neural network applications;¹⁰ and

CPUs. Intel offers CPUs with many cores enhanced by large multi-level caches and one-dimensional SIMD instructions, the kind of FPGAs used by Microsoft, and a new neural network processor that is closer to a TPU than to a CPU.¹⁹

In addition to these large players, dozens of startups are pursuing their own proposals.²⁵ To meet growing demand, architects are interconnecting hundreds to thousands of such chips to form neural-network supercomputers.

This avalanche of DNN architectures makes for interesting times in computer architecture. It is difficult to predict in 2019 which (or even if any) of these many directions will win, but the marketplace will surely settle the competition just as it settled the architectural debates of the past.

Open Architectures

Inspired by the success of open source software, the second opportunity in computer architecture is open ISAs. To create a “Linux for processors” the field needs industry-standard open ISAs so the community can create open source cores, in addition to individual companies owning proprietary ones. If many organizations design processors using the same ISA, the greater competition may drive even quicker innovation. The goal is to provide processors for chips that cost from a few cents to \$100.

The first example is RISC-V (called “RISC Five”), the fifth RISC architecture developed at the University of California, Berkeley.³² RISC-V’s has a community that maintains the architecture under the stewardship of the RISC-V

Foundation (<http://riscv.org/>). Being open allows the ISA evolution to occur in public, with hardware and software experts collaborating before decisions are finalized. An added benefit of an open foundation is the ISA is unlikely to expand primarily for marketing reasons, sometimes the only explanation for extensions of proprietary instruction sets.

RISC-V is a modular instruction set. A small base of instructions run the full open source software stack, followed by optional standard extensions designers can include or omit depending on their needs. This base includes 32-bit address and 64-bit address versions. RISC-V can grow only through optional extensions; the software stack still runs fine even if architects do not embrace new extensions. Proprietary architectures generally require upward binary compatibility, meaning when a processor company adds new feature, all future processors must also include it. Not so for RISC-V, whereby all enhancements are optional and can be deleted if not needed by an application. Here are the standard extensions so far, using initials that stand for their full names:

M. Integer multiply/divide;

A. Atomic memory operations;

F/D. Single/double-precision floating-point; and

C. Compressed instructions.

A third distinguishing feature of RISC-V is the simplicity of the ISA. While not readily quantifiable, here are two comparisons to the ARMv8 architecture, as developed by the ARM company contemporaneously:

Fewer instructions. RISC-V has many fewer instructions. There are 50 in the base that are surprisingly similar in number and nature to the original RISC-I.³⁰ The remaining standard extensions—M, A, F, and D—add 53 instructions, plus C added another 34, totaling 137. ARMv8 has more than 500; and

Fewer instruction formats. RISC-V has many fewer instruction formats, six, while ARMv8 has at least 14.

Simplicity reduces the effort to both design processors and verify hardware correctness. As the RISC-V targets range from data-center chips to IoT devices, design verification can be a significant part of the cost of development.

Fourth, RISC-V is a clean-slate design, starting 25 years later, letting its

architects learn from mistakes of its predecessors. Unlike first-generation RISC architectures, it avoids microarchitecture or technology-dependent features (such as delayed branches and delayed loads) or innovations (such as register windows) that were superseded by advances in compiler technology.

Finally, RISC-V supports DSAs by reserving a vast opcode space for custom accelerators.

Beyond RISC-V, Nvidia also announced (in 2017) a free and open architecture²⁹ it calls Nvidia Deep Learning Accelerator (NVDLA), a scalable, configurable DSA for machine-learning inference. Configuration options include data type (int8, int16, or fp16) and the size of the two-dimensional multiply matrix. Die size scales from 0.5 mm² to 3 mm² and power from 20 milliWatts to 300 milliWatts. The ISA, software stack, and implementation are all open.

Open simple architectures are synergistic with security. First, security experts do not believe in security through obscurity, so open implementations are attractive, and open implementations require an open architecture. Equally important is increasing the number of people and organizations who can innovate around secure architectures. Proprietary architectures limit participation to employees, but open architectures allow all the best minds in academia and industry to help with security. Finally, the simplicity of RISC-V makes its implementations easier to check. Moreover, the open architectures, implementations, and software stacks, plus the plasticity of FPGAs, mean architects can deploy and evaluate novel solutions online and iterate them weekly instead of annually. While FPGAs are 10× slower than custom chips, such performance is still fast enough to support online users and thus subject security innovations to real attackers. We expect open architectures to become the exemplar for hardware/software co-design by architects and security experts.

Agile Hardware Development

The Manifesto for Agile Software Development (2001) by Beck et al.¹ revolutionized software development, overcoming the frequent failure of the traditional elaborate planning and documenta-



Security experts do not believe in security through obscurity, so open implementations are attractive, and open implementations require an open architecture.



tion in waterfall development. Small programming teams quickly developed working-but-incomplete prototypes and got customer feedback before starting the next iteration. The scrum version of agile development assembles teams of five to 10 programmers doing sprints of two to four weeks per iteration.

Once again inspired by a software success, the third opportunity is agile hardware development. The good news for architects is that modern electronic computer aided design (ECAD) tools raise the level of abstraction, enabling agile development, and this higher level of abstraction increases reuse across designs.

It seems implausible to claim sprints of four weeks to apply to hardware, given the months between when a design is “taped out” and a chip is returned. Figure 9 outlines how an agile development method can work by changing the prototype at the appropriate level.²³ The innermost level is a software simulator, the easiest and quickest place to make changes if a simulator could satisfy an iteration. The next level is FPGAs that can run hundreds of times faster than a detailed software simulator. FPGAs can run operating systems and full benchmarks like those from the Standard Performance Evaluation Corporation (SPEC), allowing much more precise evaluation of prototypes. Amazon Web Services offers FPGAs in the cloud, so architects can use FPGAs without needing to first buy hardware and set up a lab. To have documented numbers for die area and power, the next outer level uses the ECAD tools to generate a chip’s layout. Even after the tools are run, some manual steps are required to refine the results before a new processor is ready to be manufactured. Processor designers call this next level a “tape in.” These first four levels all support four-week sprints.

For research purposes, we could stop at tape in, as area, energy, and performance estimates are highly accurate. However, it would be like running a long race and stopping 100 yards before the finish line because the runner can accurately predict the final time. Despite all the hard work in race preparation, the runner would miss the thrill and satisfaction of actually crossing the finish line. One advantage hardware engineers have over software engineers is they build physical things. Getting chips

back to measure, run real programs, and show to their friends and family is a great joy of hardware design.

Many researchers assume they must stop short because fabricating chips is unaffordable. When designs are small, they are surprisingly inexpensive. Architects can order 100 1-mm² chips for only \$14,000. In 28 nm, 1 mm² holds millions of transistors, enough area for both a RISC-V processor and an NVLDA accelerator. The outermost level is expensive if the designer aims to build a large chip, but an architect can demonstrate many novel ideas with small chips.

Conclusion

“The darkest hour is just before the dawn.” —Thomas Fuller, 1650

To benefit from the lessons of history, architects must appreciate that software innovations can also inspire architects, that raising the abstraction level of the hardware/software interface yields opportunities for innovation, and that the marketplace ultimately settles computer architecture debates. The iAPX-432 and Itanium illustrate how architecture investment can exceed returns, while the S/360, 8086, and ARM deliver high annual returns lasting decades with no end in sight.

The end of Dennard scaling and Moore’s Law and the deceleration of performance gains for standard microprocessors are not problems that must be solved but facts that, recognized, offer breathtaking opportunities. High-level, domain-specific languages and architectures, freeing architects from the chains of proprietary instruction sets, along with demand from the public for improved security, will usher in a new golden age for computer architects. Aided by open source ecosystems, agilely developed chips will convincingly demonstrate advances and thereby accelerate commercial adoption. The ISA philosophy of the general-purpose processors in these chips will likely be RISC, which has stood the test of time. Expect the same rapid improvement as in the last golden age, but this time in terms of cost, energy, and security, as well as in performance.

The next decade will see a Cambrian explosion of novel computer architectures, meaning exciting times for computer architects in academia and in industry. 

References

1. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... and Kern, J. *Manifesto for Agile Software Development*, 2001; <https://agilemanifesto.org/>
2. Bhandarkar, D. and Clark, D.W. Performance from architecture: Comparing a RISC and a CISC with similar hardware organization. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Santa Clara, CA, Apr. 8–11). ACM Press, New York, 1991, 310–319.
3. Chaitin, G. et al. Register allocation via coloring. *Computer Languages* 6, 1 (Jan. 1981), 47–57.
4. Dally, W. et al. Hardware-enabled artificial intelligence. In *Proceedings of the Symposium on VLSI Technology and Circuits* (Honolulu, HI, June 18–22). IEEE Press, 2018, 3–6.
5. Dennard, R. et al. Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE Journal of Solid State Circuits* 9, 5 (Oct. 1974), 256–268.
6. Emer, J. and Clark, D. A characterization of processor performance in the VAX-11/780. In *Proceedings of the 11th International Symposium on Computer Architecture* (Ann Arbor, MI, June). ACM Press, New York, 1984, 301–310.
7. Fisher, J. The VLIW machine: A multiprocessor for compiling scientific code. *Computer* 17, 7 (July 1984), 45–53.
8. Fitzpatrick, D.T., Foderaro, J.K., Katevenis, M.G., Landman, H.A., Patterson, D.A., Peek, J.B., Peshkess, Z., Sequin, C.H., Sherburne, R.W., and Van Dyke, K.S. A RISCy approach to VLSI. *ACM SIGARCH Computer Architecture News* 10, 1 (Jan. 1982), 28–32.
9. Flynn, M. Some computer organizations and their effectiveness. *IEEE Transactions on Computers* 21, 9 (Sept. 1972), 948–960.
10. Fowers, J. et al. A configurable cloud-scale DNN processor for real-time AI. In *Proceedings of the 45th ACM/IEEE Annual International Symposium on Computer Architecture* (Los Angeles, CA, June 2–6). IEEE, 2018, 1–14.
11. Hennessy, J. and Patterson, D. A New Golden Age for Computer Architecture. Turing Lecture delivered at the 45th ACM/IEEE Annual International Symposium on Computer Architecture (Los Angeles, CA, June 4, 2018); http://iscaconf.org/isca2018/turing_lecture.html; <https://www.youtube.com/watch?v=3LVEjsn8Ts>
12. Hennessy, J., Jouppi, N., Przybylski, S., Rowen, C., Gross, T., Baskett, F., and Gill, J. MIPS: A microprocessor architecture. *ACM SIGMICRO Newsletter* 13, 4 (Oct. 5, 1982), 17–22.
13. Hennessy, J. and Patterson, D. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, San Francisco, CA, 1989.
14. Hill, M. A primer on the meltdown and Spectre hardware security design flaws and their important implications. *Computer Architecture Today* blog (Feb. 15, 2018); <https://www.sigarch.org/a-primer-on-the-meltdown-spectre-hardware-security-design-flaws-and-their-important-implications/>
15. Hopkins, M. A critical look at IA-64: Massive resources, massive ILP, but can it deliver? *Microprocessor Report* 14, 2 (Feb. 7, 2000), 1–5.
16. Horowitz M. Computing’s energy problem (and what we can do about it). In *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers* (San Francisco, CA, Feb. 9–13). IEEE Press, 2014, 10–14.
17. Jouppi, N., Young, C., Patil, N., and Patterson, D. A domain-specific architecture for deep neural networks. *Commun. ACM* 61, 9 (Sept. 2018), 50–58.
18. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., and Boyle, R. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th ACM/IEEE Annual International Symposium on Computer Architecture* (Toronto, ON, Canada, June 24–28). IEEE Computer Society, 2017, 1–12.
19. Kloss, C. *Nervana Engine Delivers Deep Learning at Ludicrous Speed*. Intel blog, May 18, 2016; <https://ai.intel.com/nervana-engine-delivers-deep-learning-at-ludicrous-speed/>
20. Knuth, D. *The Art of Computer Programming: Fundamental Algorithms, First Edition*. Addison Wesley, Reading, MA, 1968.
21. Knuth, D. and Binstock, A. Interview with Donald Knuth. InformIT, Hoboken, NJ, 2010; <http://www.informit.com/articles/article.aspx>
22. Kung, H. and Leiserson, C. Systolic arrays (for VLSI). Chapter in *Sparse Matrix Proceedings Vol. 1*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979, 256–282.
23. Lee, Y., Waterman, A., Cook, H., Zimmer, B., Keller, B., Puggelli, A., ... and Chiu, P. An agile approach to building RISC-V microprocessors. *IEEE Micro* 36, 2 (Feb. 2016), 8–20.
24. Leiserson, C. et al. There’s plenty of room at the top. To appear.
25. Metz, C. Big bets on A.I. open a new frontier for chip start-ups, too. *The New York Times* (Jan. 14, 2018).
26. Moore, G. Cramming more components onto integrated circuits. *Electronics* 38, 8 (Apr. 19, 1965), 56–59.
27. Moore, G. No exponential is forever: But ‘forever’ can be delayed! [semiconductor industry]. In *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers* (San Francisco, CA, Feb. 13). IEEE, 2003, 20–23.
28. Moore, G. Progress in digital integrated electronics. In *Proceedings of the International Electronic Devices Meeting* (Washington, D.C., Dec.). IEEE, New York, 1975, 11–13.
29. Nvidia. *Nvidia Deep Learning Accelerator (NVIDIA)*, 2017; <http://nvidia.org/>
30. Patterson, D. *How Close is RISC-V to RISC-I?* ASPIRE blog, June 19, 2017; <https://aspire.eecs.berkeley.edu/2017/06/how-close-is-risc-v-to-risc-i/>
31. Patterson, D. RISCy history. *Computer Architecture Today* blog, May 30, 2018; <https://www.sigarch.org/riscy-history/>
32. Patterson, D. and Waterman, A. *The RISC-V Reader: An Open Architecture Atlas*. Strawberry Canyon LLC, San Francisco, CA, 2017.
33. Rowen, C., Przybylski, S., Jouppi, N., Gross, T., Shott, J., and Hennessy, J. A pipelined 32b NMOS microprocessor. In *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers* (San Francisco, CA, Feb. 22–24). IEEE, 1984, 180–181.
34. Schwarz, M., Schwarzl, M., Lipp, M., and Gruss, D. Netspectre: Read arbitrary memory over network. arXiv preprint, 2018; <https://arxiv.org/pdf/1807.10535.pdf>
35. Sherburne, R., Katevenis, M., Patterson, D., and Sequin, C. A 32b NMOS microprocessor with a large register file. In *Proceedings of the IEEE International Solid-State Circuits Conference* (San Francisco, CA, Feb. 22–24). IEEE Press, 1984, 168–169.
36. Thacker, C., MacCreight, E., and Lampson, B. *Alto: A Personal Computer*. CSL-79-11, Xerox Palo Alto Research Center, Palo Alto, CA, Aug. 7, 1979; <http://people.scs.carleton.ca/~soma/dists/fall2008/alto.pdf>
37. Turner, P., Parseghian, P., and Linton, M. Protecting against the new ‘LITF’ speculative vulnerabilities. Google blog, Aug. 14, 2018; <https://cloud.google.com/blog/products/gcp/protecting-against-the-new-litf-speculative-vulnerabilities>
38. Van Bulck, J. et al. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *Proceedings of the 27th USENIX Security Symposium* (Baltimore, MD, Aug. 15–17). USENIX Association, Berkeley, CA, 2018.
39. Wilkes, M. and Stringer, J. Micro-programming and the design of the control circuits in an electronic digital computer. *Mathematical Proceedings of the Cambridge Philosophical Society* 49, 2 (Apr. 1953), 230–238.
40. XLA Team. XLA – TensorFlow. Mar. 6, 2017; <https://developers.googleblog.com/2017/03/xlatensorflow-compiled.html>

John L. Hennessy (hennessy@stanford.edu) is Past-President of Stanford University, Stanford, CA, USA, and is Chairman of Alphabet Inc., Mountain View, CA, USA.

David A. Patterson (pattsrn@berkeley.edu) is the Pardee Professor of Computer Science, Emeritus at the University of California, Berkeley, CA, USA, and a Distinguished Engineer at Google, Mountain View, CA, USA.

© 2019 ACM 0001-0782/19/2 \$15.00



To watch Hennessy and Patterson’s full Turing Lecture, see <https://www.acm.org/hennessy-patterson-turing-lecture>

Diffusion speed and scale depend on all kinds of information, not just which users have the most or fewest connections.

BY CHAO GAO, ZHEN SU, JIMING LIU, AND JÜRGEN KURTHS

Even Central Users Do Not Always Drive Information Diffusion

COMMUNITY STRUCTURE, a significant and useful statistical characteristic, is ubiquitous in social networks.¹⁷ Based on it, a network can be viewed as consisting of multiple units. The nodes (users) are highly connected to each other inside a unit, while the connections between units are sparse.^{4,17} For example, people with similar

interests or backgrounds might join together to form a community or web-pages with related topics might cluster together. Different types of information, including rumors,⁵ virus attacks,¹⁰ and even cyber epidemics diffuse through social networks,⁸ possibly leading to unexpected social effects. A typical example is the worldwide cyberattack by WannaCry ransomware, as first reported May 12, 2017, that resulted in the infections of more than 200,000

organizations worldwide.¹⁵ The underlying attack reflects a malicious diffusion in the presence of communities; that is, the homogeneous feature of individuals leads to the community's vulnerability. It is against this backdrop that understanding the potential dynamics could help network administrators gain insight into controlling unwanted information diffusion. Much research today involves networks with community structure (such as to detect

potential communities,²¹ model diffusion dynamics,⁶ and control information dissemination and sharing¹⁹). In particular, the influence of each node in the diffusion process must be taken into consideration. In simulation experiments, the source nodes that trigger diffusion are selected by researchers at random from a network or based on predefined measures of centrality.

In recent decades, multiple centrality measures have been proposed to statistically evaluate the importance or influence of a node (such as degree,² betweenness,¹¹ coreness,¹⁴ and eigenvector³). Degree is used mainly for characterizing the partial influence of a node.² Betweenness reflects the potential power of a node in controlling information flow.¹¹ Coreness implies that if a node lies in the core part of a network, the node is more important.¹⁴ And eigenvector accounts for two factors: a node's connections and its neighbors' influences.³ State-of-the-art studies have looked into nodes with relatively greater centrality in information diffusion. However, the influence of nodes with relatively less centrality on the diffusion process has never been completely addressed. In this article, we aim to explain the importance of two kinds of nodes in the information-diffusion process in a community-based network. Our findings can help network administrators better understand the diversity of communities and associated complexity of the diffusion process.

Potential for a Crossover

Centrality characterizes the influence of a node or user in a network. Intuitively, nodes with relatively greater centrality should be more important than those with relatively less centrality, as they can lead to fast, large-scale diffusion. However, we often find diffusion breaks out from a group of nodes with relatively less centrality in the real world.

Figure 1 outlines two diffusion processes as triggered by different initial states in a community-based network. The dotted circles represent different communities in a network. With a strong community structure, the density of the intracommunity links is much greater than that of the intercommunity; for instance, only one

» key insights

- **Central users do not always contribute to information diffusion due to a crossover point of two diffusion processes triggered by source users with most and fewest connections, respectively.**
- **A strong community structure decreases the stability of the crossover point in terms of influence of two diffusion processes.**
- **Compared to the influence of community structure on the diffusion process, the increment of source nodes leads the diffusion scale to the appearance of an earlier crossover point.**

link connects different communities in the figure. Two diffusion processes are triggered by the maximum degree nodes, as in Figure 1a, and minimum degree nodes, as in Figure 1b, respectively. Theoretically, the requirement of simultaneous diffusion of source nodes is not necessary due to the independent infection process between each infected node and its susceptible neighbors. However, to ensure a clear, quick observation of a diffusion phenomenon, as in, say, letting source nodes (represented by two red nodes in Figure 1) initiate a diffusion process simultaneously. Based on the propagation rules defined in a typical “two-state” diffusion model,²² each infected node tries to infect all its susceptible neighbors with a certain probability at each time step, bringing uncertainty during the propagation process. For example, as shown in Figure 1a, “A” is infected by “E” at $t = 3$, rather than by its source node neighbor at $t = 1$ or $t = 2$. Meanwhile, at $t = 8$, “D” remains susceptible until infected by “C” at $t = 10$. In Figure 1a, information diffuses quickly at the initial stage. The speed of diffusion could be enhanced by increasing the initial number of source nodes. Note also two factors concerning the effect of network structure on the potential diffusion process:

Effective diffusion links. The effective diffusion links represent the connections that make a key contribution to the diffusion process;²² for example, the link between the two source nodes in Figure 1a does not benefit subsequent diffusion. With the increment of source nodes, there is a strong likelihood that some might cluster together, as outlined in Figure 1a, thus decreasing the effective diffusion links. But

such a negative effect is unlikely to show up in Figure 1b unless there are more initial source nodes.

Community structure. Global diffusion in a network with community structure is restricted by intercommunity links;¹⁶ that is, global diffusion is facilitated only when the nodes on the intercommunity links (also called “bridge nodes” by network architects) are infected. In Figure 1a, four nodes—“A,” “B,” “C,” and “D”—are bridge nodes. The global diffusion is suppressed temporarily because “D” remains susceptible at $t = 8$. Although “D” is not infected by “C” in Figure 1b, the other source node initiates new propagation in other communities, enhancing global diffusion.

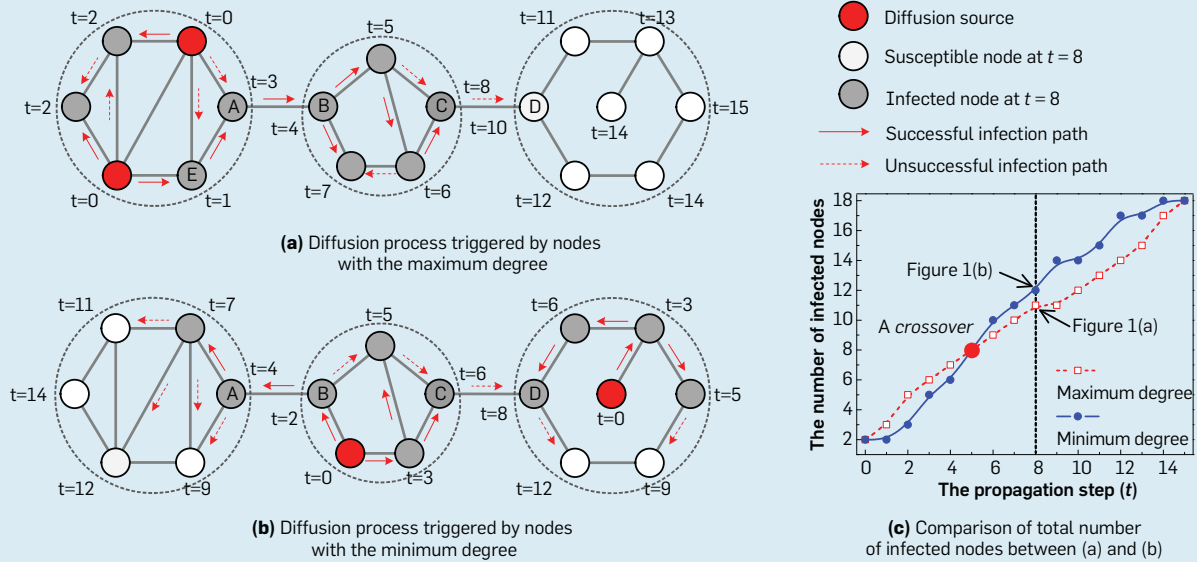
Based on two factors—effective diffusion links and community structure—the two diffusion processes—maximum-degree-based and minimum-degree-based—in Figure 1a and Figure 1b might result in a crossover in terms of diffusion scale. The diffusion scale of Figure 1b would be greater than the diffusion scale of Figure 1a. Differing diffusion scales involve several questions: For example, do the most connected users always drive information diffusion in social networks? If not, what kind of influence would the community structure have on the diffusion process? To answer, we simulated information diffusion in both real-world and synthetic networks with community structure to investigate the potential crossover points of two diffusion processes.

Crossover in Terms of Propagation Scale

Many real-world systems can be described as networks; examples are email, social, and technological. Here, we select a benchmark university email dataset and construct an interaction network to demonstrate the influence of two diffusion processes—maximum-degree-based and minimum-degree-based—triggered by two kinds of initial source nodes with greatest- and least-degree centrality. The network includes 1,133 nodes and 5,451 links.⁹ The average degree and clustering coefficient in the network are 9.62 and 0.25, respectively. Specifically, the clustering coefficient is used to denote the degree to which the neighbors of a user know

Figure 1. Schematic of two diffusion processes: maximum-degree-based and minimum-degree-based.

During diffusion, all nodes in a network are divided into three categories: source nodes, infected nodes, and susceptible nodes. The source nodes receive information first and trigger the overall diffusion process. The infected and susceptible nodes represent nodes that have or have not received information. At each time step, each infected node tries to infect all susceptible neighbors with a certain probability. The final infected time of each node is labeled; for example, at time step $t = 8$, two snapshots are used to present two diffusion processes: (a) the process is triggered by two highly connected source nodes; and (b) the process starts from the relatively least-connected source nodes. In particular, solid and dashed arrows associated with links denote successful and unsuccessful infection paths. Section (c) reports the dynamic changes of infected nodes in (a) and (b) at each time step. The crossover of the two propagation scales in (a) and (b) is plotted in (c).



each other.² A greater value means a network's greater inherent tendency to cluster of a network.

Simulation model. Each user is essentially represented by two states in the scenario of information diffusion—"received" a message or "not received" a message. We adopt a typical "two-state" diffusion model—the interactive email model proposed by Zou et al.²² and implemented by Gao et al.⁷—as a testbed for characterizing various kinds of information-diffusion processes.^{6,7} Each node in the model reflects one of two corresponding states—"susceptible" or "infected"—and the transition cannot be reversed; that is, a user who receives a message is denoted as an "infected" node, and others are denoted as "susceptible." In a diffusion process, a basic step that benefits the subsequent process is a user must change state from "susceptible" to "infected." The diffusion process is triggered by user behavior—the email-checking time interval and the email-clicking probability. The diffusion rate is thus different for different users. By

assuming the behavior of each user is independent, we used a Gaussian distribution to depict the features of two behaviors when the sample size is large.⁷ In this article, we use two normal distribution functions— $N(40, 20^2)$ and $N(0.5, 0.3^2)$ —to represent the features of checking intervals and clicking probability.^{7,22}

Experimental settings. We set the percentage of initial source nodes at 20%. We simulated two diffusion processes triggered by maximum-degree nodes and minimum-degree nodes in the email network simultaneously and independently. We averaged simulation results by following 100 runs for wiping off the computational fluctuation. In each run, we terminated the propagation process after 2,000 time steps to ensure the whole system is and would remain stable.

Experimental results. In general, we used the proportion or total number of infected nodes to evaluate a propagation process. Here, we adopt the total number of infected nodes at time t as the propagation scale for the pur-

pose of giving an intuitive demonstration of a crossover, as plotted in Figure 2a. We also investigated the dynamic changes of two propagation scales by calculating the numerical difference of two propagation processes at each time t , as plotted in Figure 2b. Three critical points are labeled t_1 , t_c , and t_2 .

When $t < t_c$, the difference between propagation scales is positive, as shown in Figure 2b. That difference corresponds to the stage (see $t < t_c$ in Figure 2a) when the propagation process, triggered by the maximum degree nodes, diffuses more quickly than the other process. The maximum difference is found the moment $t = t_1$ in Figure 2b. However, as the propagation continues (see $t_1 < t < t_2$), the numerical difference decreases sharply, as plotted in Figure 2b. This unexpected change implies the propagation process, triggered by the minimum-degree nodes, represents relatively greater propagation ability. The shift coincides with the dynamic change of the propagation scale in Figure 2a. When $t > t_c$, the shift is completely reversed. The propaga-

tion process, triggered by minimum-degree nodes, leads to a larger scale of diffusion until the whole propagation system is stable. The maximum difference is reached numerically at time t_2 , even exceeding that of time t_1 . The time t_c is the exact crossover point of the two propagation processes in Figure 2.

During the propagation process, the most important period is between t_1 and t_2 when the two potential propagation processes undergo different transitions. The phenomenon in Figure 2 shows that, compared to nodes with relatively greater centrality, those with relatively less centrality could ensure the stability of propagation, reflecting its vital role in long-term diffusion. Such an interesting phenomenon also implies that in some cases, even central users may not always drive information diffusion. To validate this assumption,

we conducted more simulations, as we explore in two real-world networks in the next section.

Nonlinear Crossover Phenomenon

To obtain a deeper understanding of such a phenomenon, we simulated propagations in real-world networks:

Datasets. We included two real-world networks with a potential community structure—a U.S. political weblog network (PolBlogs)¹ and a scientific collaboration network (Arxiv).¹⁸ The PolBlogs network includes 1,490 nodes and 19,025 links. Its average degree and clustering coefficient were 22.44 and 0.36, respectively. Two political communities represent liberal blogs and conservative blogs, respectively.¹ Mark Newman of the University of Michigan analyzed the Arxiv network, with 56,276 nodes and

631,632 links,¹⁸ looking to identify community-based features on co-authorship patterns. The average degree and clustering coefficient of the Arxiv networks were 11.23 and 0.69, respectively, and the overall Arxiv network included 42 communities.

Experimental settings. The initial proportion of source nodes we denote as i_0 varied from 0.01 to 0.5 and was divided into two parts. When the initial proportion is between 0.01 and 0.05, the rate of increase increases by 1%, after which the rate of increase increases to 5%. We selected the initial source nodes based on four kinds of centrality measures: degree,² betweenness,¹¹ k-core,¹⁴ and eigenvector.³

Experimental results. Under the same experimental conditions as outlined in the previous section, two propagation processes are triggered by source nodes with relatively greatest and relatively least centrality. Our focus is still on the critical crossover points. Since they are relevant to the time steps of each propagation, we recorded the time each crossover point emerged and normalized them based on $t_c/2000$, where 2,000 was the total time steps, as is plotted in Figure 3. Despite different centrality measures and networks, the figure reveals several similarities:

Propagation scale. The crossover in terms of propagation scale emerges when the initial proportion of source nodes is low (such as 1%). Experiments on different kinds of networks show that a stable state, when the crossover phenomenon can be triggered, is indeed possible when i_0 increases;

Crossover points. The time of different crossover points is generally a decreasing function of the initial proportion of source nodes i_0 ; that is, the crossover points come earlier with the increment of the initial source nodes; and

Strength of community structure. The different crossover points under the same degree of centrality reveal the strength of influence a community structure exerts on the propagation process.

Experimental results in real-world networks demonstrate our assumption that central users (or nodes with relatively greatest centrality) do not always drive information diffusion. Specifically, the crossover phenomenon pre-

Figure 2. Crossover of two propagation processes in terms of propagation scale in a university email network.

Time t_c represents the critical moment the crossover begins, indicating nodes with relatively greater centrality do not always drive diffusion.

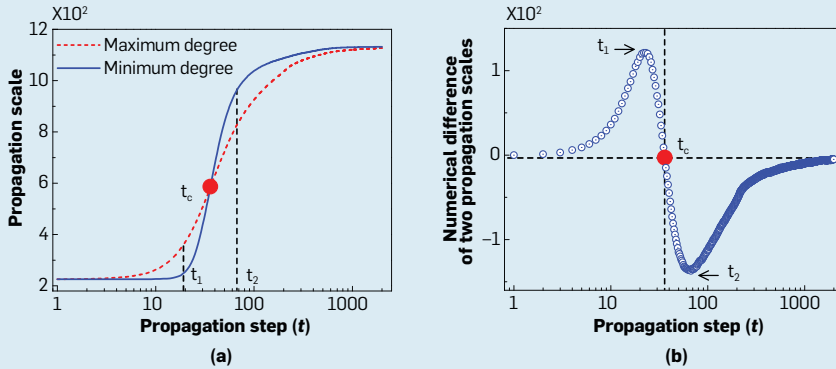
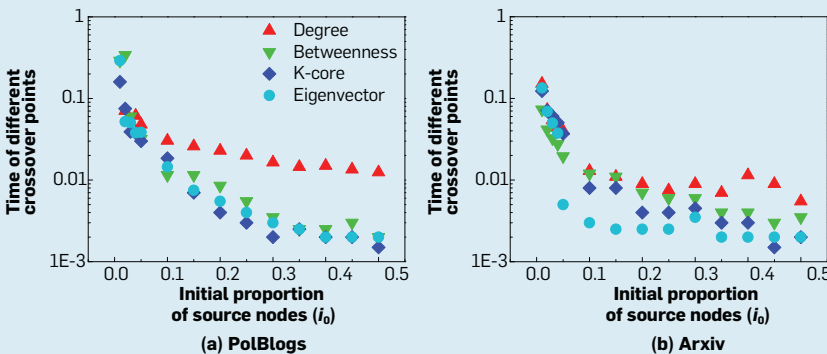


Figure 3. Nonlinear crossover phenomenon in networks with community structure.

Each point represents the potential crossover point in terms of propagation scale. The results indicate both community structure and initial proportion of source nodes have influence on such phenomena.



vails and will intensify when the initial proportion of source nodes increases. We also investigated the influence of different initial states on effective diffusion links to verify our hypothesis, as proposed in Figure 1.

Diffusion links analysis. Taking the email network as an example, we evaluated two opposite initial states under four kinds of centrality measures by calculating the average distance of source nodes. This distance can reveal the degree to which source nodes are close to each other. A shorter average distance refers to a relatively greater probability of being clustered together. Diffusion links between source nodes could thus be decreased. As outlined in Figure 4, under the condition of nodes with relatively greatest centrality functioning as source nodes, the average distance of these sources is much shorter than the distance under nodes with relatively least centrality being treated as source nodes. The reason for the shorter distance is that nodes with relatively least centrality are located at the boundary of a network, and vice versa. Hence, when nodes with relatively greatest centrality are selected as sources, the increasing proportion of source nodes can lead to a relative decrease in effective diffusion links. Moreover, the subsequent propagation process would be suppressed. How nodes with relatively greatest centrality might enhance information diffusion depends on the number of initial source nodes clustering together. In particular, when there are few initial source nodes (such as less than 1%), the propagation ability of nodes with relatively greatest centrality can take full effect.

Behind the crossover phenomenon, this shift is derived by taking into account two propagation processes—greatest-centrality-based and least-centrality-based—as triggered by different initial states. In the domain of social networks, analysis of a diffusion process is associated with a selected propagation model and the topology of an underlying network. In our experiments, we simulated two propagation processes simultaneously based on the same model, indicating the crossover phenomenon is independent of the selected simulation models. The only factor that should be relevant to this observed phenomenon is thus the structure of

The underlying attack reflects a malicious diffusion in the presence of communities; that is, the homogeneous feature of individuals leads to the community's vulnerability.

the diffusion network. Specifically, although the initial source nodes in two propagation processes share the same proportion, the potential processes can be different in light of the diversity of the underlying diffusion network.

Since both the initial proportion of source nodes and the strength of community structure influence potential crossover points, we explored more simulations in synthetic networks to identify the influence of these factors.

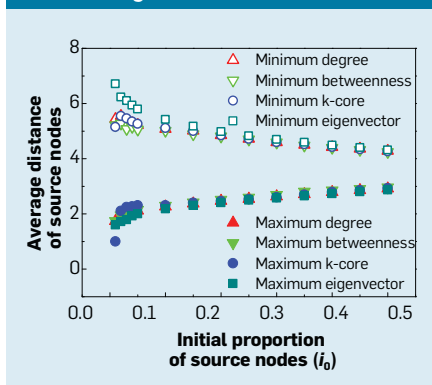
Influence Comparison

To help us understand the influence of the strength of community structure on the diffusion process, we adopted a community-network generator¹² with tunable parameters:

Datasets. We built two synthetic networks by varying the mix parameter $\mu = 0.05$ and 0.5 . This parameter controls the strength of a community structure, indicating that with a smaller μ , the community structure of a synthetic network is stronger. The generator includes two kinds of parameters—specified and default settings. We assigned the specified settings as follows: total number of nodes = 1,000; average degree = 15; maximum degree in the network = 50; and maximum and minimum community sizes = 50 and 20, respectively. We kept the default settings, with the exponent for the degree distribution at 2; the exponent for the community-size distribution at 1; and the number of overlapping nodes and number of memberships of the overlapping nodes both at 0.

Experimental results. Following the same experimental scenario, we per-

Figure 4. Average distance of source nodes in the email network. Statistical results indicate source nodes with relatively greater centrality tend to be clustered together.



formed extensive simulations in two such synthetic networks.¹² Comparing the influence of the initial proportion of source nodes and the strength of community structure, Figure 5 in-

cludes average time and standard deviation of different crossover points with respect to four kinds of centrality measures: degree,² betweenness,¹¹ k-core,¹⁴ and eigenvector.³ Figure 5 includes fur-

ther detail, in addition to the crossover phenomenon:

Crossover points. Comparing the statistical results in the phase II segment of the figure, although the increment of the mix parameter μ triggers the crossover points slightly earlier, it is still far less than the influence resulting from increasing the initial source nodes; and

Deviation. The deviation of different crossover points tends to be stable in the wake of a weaker community structure, or greater value for μ .

On the basis of the simulation results in synthetic networks, we found two types of non-centrality-related network influence:

Strength of community structure. The stability of crossover points is inversely related to the strength of a community structure, demonstrating the strong (though indirect) influence of community structure on the diffusion process; and

Increment of initial source nodes. The increment of the initial source nodes is the primary factor resulting in an earlier crossover phenomenon.

We likewise analyzed the influence of community structure on two diffusion processes—maximum-degree-based and minimum-degree-based—to verify our hypothesis, as proposed in Figure 1.

Influence of community structure. Taking the synthetic network with $\mu = 0.05$ (Figure 5a) as an example, the moment the crossover phenomenon begins to emerge was visualized to show the states of all nodes in two propagation processes being initialized based on degree of centrality. Figure 6 highlights the detailed states of nodes in each community in various colors. Moreover, we extracted five communities we labeled as “ C_0 ,” “ C_1 ,” “ C_2 ,” “ C_3 ,” and “ C_4 ” that include only two kinds of nodes.

Figure 6 outlines that a strong community structure does not benefit a subsequent propagation process. When nodes with relatively greater centrality are treated as sources, source nodes tend to be clustered together, decreasing (to some extent) the effective diffusion links. In a network with a strong community structure, global diffusion can be enhanced only when the nodes on the intercommunity links become infected. In the worst case, all

Figure 5. Average time of crossover points in synthetic networks with different community structures.

The mix parameter μ controls the strength of community structure of the synthetic networks. Each subgraph includes the average crossover point of four measures of centrality and the standard deviation. The statistical results indicate the increment of the initial source nodes is the main factor causing earlier crossover points, while the stronger a community structure a network has, the less stable are the crossover points.

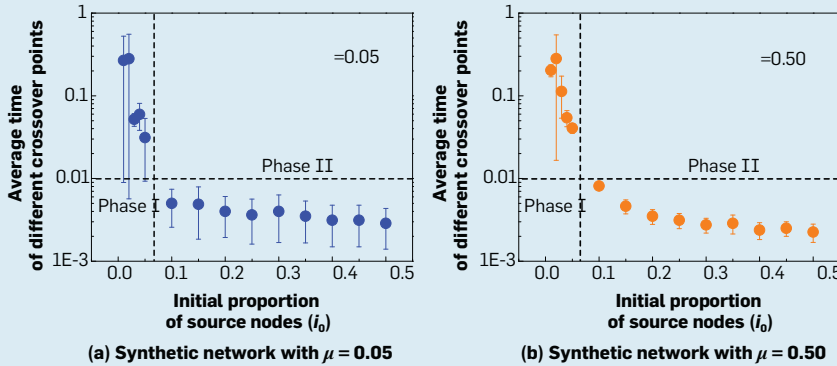
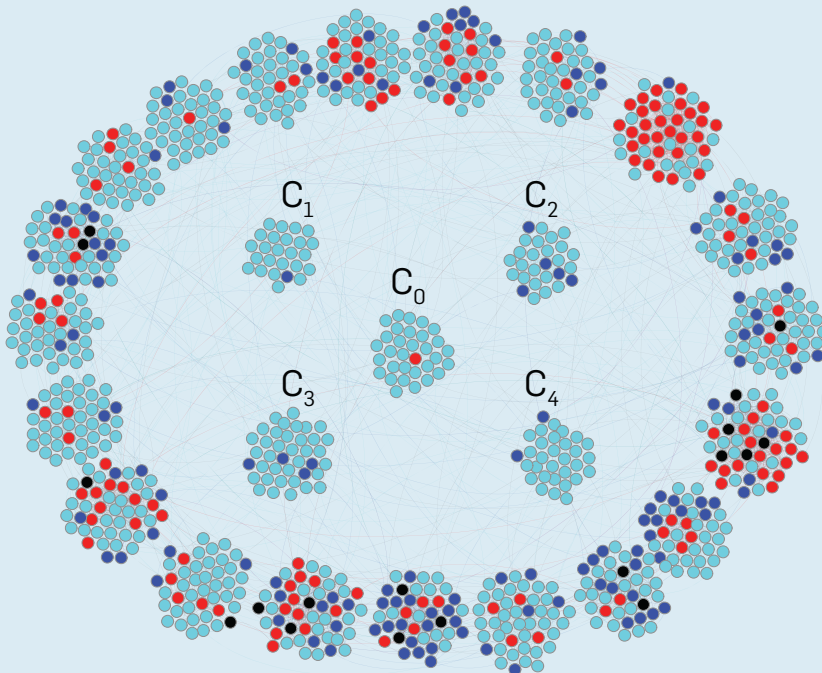


Figure 6. Visualization of two propagation processes—maximum-degree-based and minimum-degree-based—in the synthetic network with $\mu = 0.05$ when the crossover phenomenon emerges; the susceptible nodes are marked in cyan.

The infected nodes, highlighted in red or blue, belong solely to the maximum-degree-based process or the minimum-degree-based process, respectively. The black ones represent the infected nodes in both processes. Five communities—“ C_0 ,” “ C_1 ,” “ C_2 ,” “ C_3 ,” and “ C_4 ”—include two kinds of nodes, demonstrating that a strong community structure could hinder or even prevent global diffusion.



source nodes are distributed over only one community, thereby suppressing global diffusion.

However, diffusion is quite different when nodes characterized by relatively least centrality are viewed as sources. Source nodes under such conditions are distributed over more communities and more likely to facilitate global diffusion. Moreover, the worst case is unlikely to appear due to the relatively greater proportion of low-degree nodes in a network. That is why there are fewer red nodes in “ C_0 ,” “ C_1 ,” “ C_2 ,” “ C_3 ,” and “ C_4 ” than blue nodes. As the two propagation processes in Figure 6—maximum-degree-based and minimum-degree-based—proceed, such phenomenon will intensify. Finally, the various diffusion scenarios we have addressed also increase the fluctuation of crossover points.

For networks with weak community structures, the increasing proportion of intracommunity links makes global diffusion more likely, making crossover points relatively stable.

Conclusion

We have explored the nonlinear crossover of two diffusion processes—central-user-based and boundary-user-based—triggered by two opposite initial states in networks with community structure. We first considered the universality of the crossover phenomenon, then offered a detailed comparison with respect to the influence of community structure and initial proportion of source nodes on the diffusion process. The results were twofold: Networks with weak community structure could increase the stability of crossover points; and compared to the influence of community structure, the increment of the initial source nodes is the primary factor leading to an earlier crossover phenomenon.

The crossover phenomenon shows the topology of a network is a major factor affecting the diffusion process. A deep understanding of diffusion dynamics requires consideration of both network topology and dynamical correlations. Many popular theoretical approaches (such as mean field, dynamical message passing, and pairwise approximation) are used to study the dynamics of different kinds of information diffusion, but the difficulty

of capturing both network topology and dynamical correlations remains an open topic.²⁰ Even with the continuous-time Markov approach, the complicated master equations lead to yet another challenge—that the approach is unlikely to directly yield analytical or numerical results for large-scale networks. Studies investigating the balance between potential diffusion dynamics and solving computational complexity are still being challenged.

This article has offered insight into the dynamics of information diffusion in community-based networks. For instance, compared with the ability of nodes with relatively greater centrality to dramatically enhance diffusion speed at the initial stage, nodes with relatively least centrality could in fact have a greater propagation effect in the long term, especially when a network includes more initial source nodes. However, we are not saying nodes with relatively least centrality are *critically* important. It is the topological structure that establishes an explicit and complex connection between the two kinds of nodes. In some cases, such connections suggest users with relatively least centrality should be taken into consideration, as they could still significantly influence global diffusion.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant No. 61402379), Hong Kong Research Grants Council (No. HKBU12202415), CQ CSTC (grant No. cstc2018jcyjAX0274), the Fundamental Research Funds for the Central Universities (grant No. XDJK2016A008), and Chongqing Graduate Student Research Innovation Project (grant No. CYS17075).

References

1. Adamic, L.A. and Glance, N. The political blogosphere and the 2004 U.S. election: Divided they blog. In *Proceedings of the Third International Workshop on Link Discovery* (Chicago, IL, Aug. 21–25). ACM Press, New York, 2005, 36–43.
2. Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of Modern Physics* 74, 1 (Jan. 2002), 47–97.
3. Borgatti, S.P. Centrality and network flow. *Social Networks* 27, 1 (Jan. 2005), 55–71.
4. De Meo, P., Ferrara, E., Fiumara, G., and Provetti, A. On Facebook, most ties are weak. *Commun. ACM* 57, 11 (Oct. 2014), 78–84.
5. Doerr, B., Fouz, M., and Friedrich, T. Why rumors spread so quickly in social networks. *Commun. ACM* 55, 6

(June 2012), 70–75.

6. Gao, C. and Liu, J.M. Network-based modeling for characterizing human collective behaviors during extreme events. *IEEE Transactions on System, Man, and Cybernetics: Systems* 47, 1 (Jan. 2017), 171–183.
7. Gao, C., and Liu, J.M. Modeling and restraining mobile virus propagation. *IEEE Transactions on Mobile Computing* 12, 3 (Mar. 2013), 529–541.
8. Goel, S., Watts, D.J., and Goldstein, D.G. The structure of online diffusion networks. In *Proceedings of the 13th ACM Conference on Electronic Commerce* (Valencia, Spain, June 4–8). ACM Press, New York, 2012, 623–638.
9. Guimerà, R., Danon, L., Díaz-Guilera, A., Giralt, F., and Arenas, A. Self-similar community structure in a network of human interactions. *Physical Review E* 68, 6 (Dec. 2003), 065103.
10. Howard, B. Analyzing online social networks. *Commun. ACM* 51, 11 (Nov. 2008), 14–16.
11. Kitsak, M., Gallos, L.K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H.E., and Makse, H.A. Identification of influential spreaders in complex networks. *Nature Physics* 6, 11 (Aug. 2010), 888–893.
12. Lancichinetti, A., Fortunato, S., and Radicchi, F. Benchmark graphs for testing community detection algorithms. *Physical Review E* 78, 4 (Oct. 2008).
13. Leskovec, J., Kleinberg, J., and Faloutsos, C. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data* 1, 1 (Mar. 2007).
14. Liu, Y.Y., Slotine, J.J., and Barabási, A.-L. Controllability of complex networks. *Nature* 473, 7346 (May 2011), 167–173.
15. McGoogan, C. What is WannaCry and how does ransomware work? *The Telegraph* (May 18, 2017); <http://www.telegraph.co.uk/technology/0/ransomware-does-work/>
16. Nematzadeh, A., Ferrara, E., Flammini, A., and Ahn, Y.-Y. Optimal network modularity for information diffusion. *Physical Review Letters* 113, 8 (Aug. 2014), 088701.
17. Newman, M.E.J. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* 103, 23 (June 2006), 8577–8582.
18. Newman, M.E.J. Co-authorship networks and patterns of scientific collaboration. *Proceedings of the National Academy of Sciences* 101, Supplement 1 (Apr. 2004), 5200–5205.
19. Ranjbar, A. and Maheswaran, M. Using community structure to control information sharing in online social networks. *Computer Communications* 41 (Jan. 2014), 11–21.
20. Wang, W., Tang, M., Stanley, H.E., and Braunstein, L.A. Unification of theoretical approaches for epidemic spreading on complex networks. *Reports on Progress in Physics* 80, 3 (Feb. 2017), 036603.
21. Xie, J.R., Kelley, S., and Szymanski, B.K. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys* 45, 4 (Aug. 2013), 43:1–43:35.
22. Zou, C.C., Towsley D., and Gong W. Modeling and simulation study of the propagation and defense of Internet e-mail worms. *IEEE Transactions on Dependable and Secure Computing* 4, 2 (Apr. 2007), 105–118.

Chao Gao (cgao@swu.edu.cn) is a professor in the College of Computer and Information Science, Southwest University, Chongqing, China, and a visiting scholar in the Humboldt University of Berlin, Germany.

Zhen Su (zsstar@outlook.com) is pursuing a master's degree in the College of Computer and Information Science, Southwest University, Chongqing, China.

Jiming Liu (jiming@comp.hkbu.edu.hk) (corresponding author) is a professor of computer science and associate vice president (research) at Hong Kong Baptist University, Hong Kong, China.

Jürgen Kurths (Juergen.Kurths@pik-potsdam.de) is a professor of nonlinear dynamics in the Humboldt University of Berlin, Germany, and Chair of the Research Domain Transdisciplinary Concepts in the Potsdam Institute for Climate Impact Research, Potsdam, Germany.

DOI:10.1145/3224204

SONYC integrates sensors, machine listening, data analytics, and citizen science to address noise pollution in New York City.

BY JUAN P. BELLO, CLAUDIO SILVA, ODED NOV, R. LUKE DUBOIS, ANISH ARORA, JUSTIN SALAMON, CHARLES MYDLARZ, AND HARISH DORAISWAMY

SONYC: A System for Monitoring, Analyzing, and Mitigating Urban Noise Pollution

NOISE IS UNWANTED or harmful sound from environmental sources, including traffic, construction, industrial, and social activity. Noise pollution is one of the topmost quality-of-life concerns for urban residents in the U.S., with more than 70 million people nationwide exposed to noise levels beyond the limit the U.S. Environmental Protection Agency (EPA) considers harmful.¹² Such levels have proven effects on health, including sleep disruption, hypertension, heart disease, and hearing loss.^{5,11,12} In addition, there is evidence of harmful effects on educational performance, with studies showing noise pollution causing learning and cognitive impairment in children, resulting in

decreased memory capacity, reading skills, and test scores.^{2,5}

The economic impact of noise is also significant. The World Health Organization estimates that, as of 2012, one million healthy life-years in Western Europe were being lost annually to environmental noise.¹¹ Other estimates put the external cost of noise-related health issues in the E.U. between 0.3%–0.4% of GDP¹⁴ and 0.2% of GDP in Japan.¹⁶ Studies in the U.S. and Europe also demonstrate the relationship between environmental noise and real estate markets, with housing prices falling as much as 2% per decibel (dB) of noise increase.^{21,30} Noise pollution is not merely an annoyance but an important problem with broad societal effects that apply to a significant portion of the population. It is clear that effective noise mitigation is in the public interest, with the promise of health, economic, and quality-of-life benefits.

Mitigation

Noise can be mitigated at the receiver's end by, say, wearing earplugs or along the transmission path by, say, erecting sound barriers along major roads. These strategies do not, however, reduce noise emissions but instead put the burden of mitigation on the receiver.¹² Alternatively, noise can be mitigated at the source (such as by designing aircraft with quieter engines, acoustically treating night clubs, muffling jackhammers for roadwork, and stopping unnecessary

>> key insights

- **Public exposure to noise is a growing concern in cities, leading to substantial health, educational and economic costs, but noise is ephemeral and invisible, making it difficult for city agencies to monitor it effectively.**
- **An interdisciplinary effort explores new ways to use both fixed and mobile sensors, with output annotated by citizen scientists, for training novel machine-listening models and analyzing spatiotemporal noise patterns.**
- **The resulting fine-grain and aggregate analytics layers help public agencies monitor the local environment and intervene to mitigate noise pollution.**



honking). These actions are commonly encouraged and incentivized through a regulatory framework that uses fines and other penalties to raise the cost of emitting noise.²⁰ However, enforcing noise codes in large urban areas, to the point where they effectively deter noise emissions, is far from trivial.

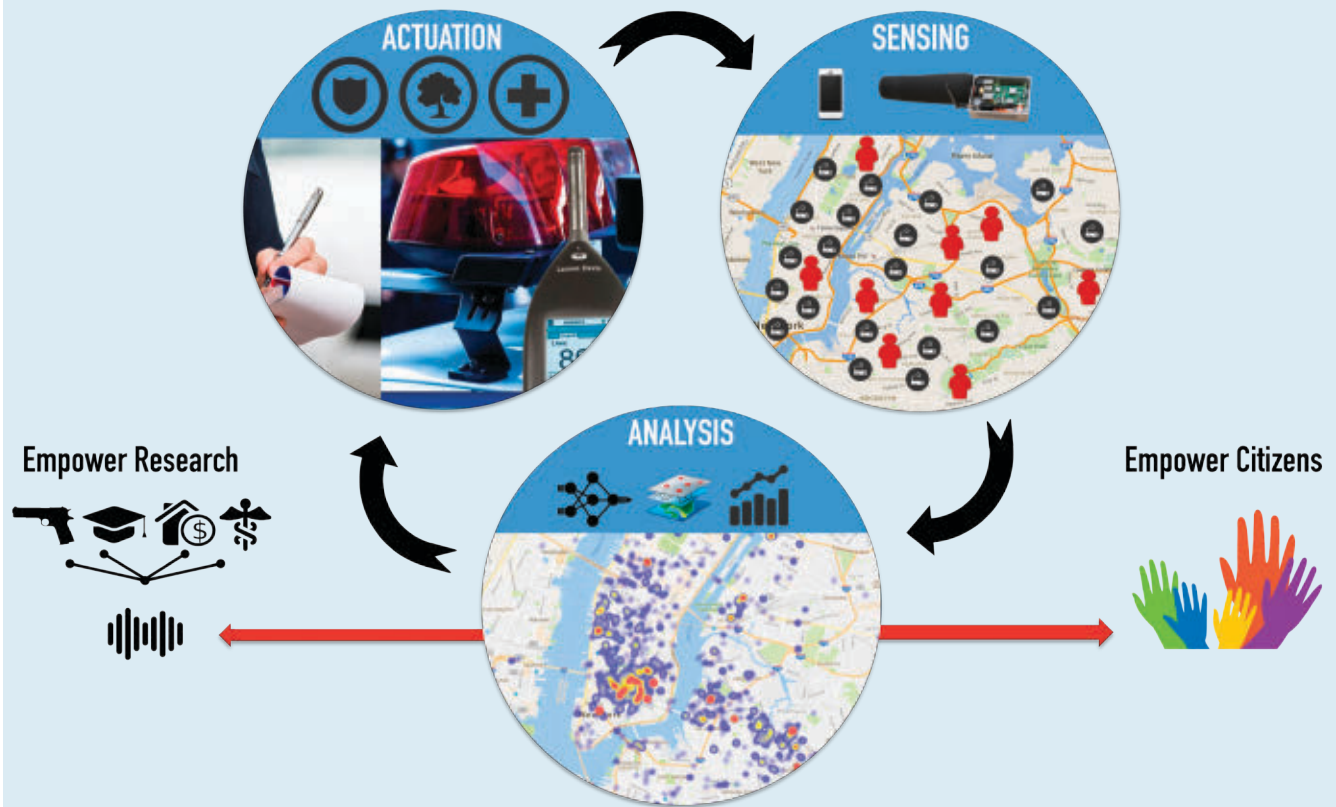
Consider New York City. Beyond the occasional physical inspection, the city government monitors noise through its 311 service for civil complaints. Since 2010, 311 has logged more than 2.7 million noise-related complaints, significantly more than

for any other type of complaint.^a This averages approximately 834 complaints a day, the most comprehensive citizen noise-reporting system in the world. However, research by New York City's Department of Health and Mental Hygiene (DOHMH) found 311 data does not accurately capture information about all noise exposure in the city.²² It identified the top sources of disruptive noise to be traffic, sirens, and construction; the effect to be similar in the boroughs of Manhat-

a <http://www1.nyc.gov/311>

tan, Brooklyn, and the Bronx; and low-income and unemployed New Yorkers among the most frequently exposed. In contrast, 311 noise-complaint data collected for the same period emphasized social noise (such as parties, car alarms, loud talking, music, and TV), with fewer complaints citing traffic or construction. Notably, residents of Manhattan, home to many affluent New Yorkers, are more than twice as likely to file 311 complaints than those in the other boroughs. This pattern clearly highlights the need to collect objective noise measurements across

Figure 1. The SONYC cyber-physical system loop, including intelligent sensing, noise analysis at city-scale, and data-driven mitigation. SONYC supports new research in the social sciences and public health while providing the data citizens need to improve their communities.



the city, along with citizen reporting, to fully characterize the phenomenon.

A closely related challenge involves how to respond to potential violations of the noise code. In New York, the subset of noise complaints pertaining to static, systemic sources (such as construction, animals, traffic, air conditioning, and ventilation units) are routed to the city’s Department of Environmental Protection (DEP), which employs approximately 50 highly qualified inspectors to measure sound levels and issue a notice of violation as needed. Unfortunately, the limited human resources and high number of complaints result in average response times of more than five days. Given the ephemeral nature of sound, a very small proportion of inspections actually result in a violation observed, let alone penalized.

To complicate matters, even when noise sources are active during inspections, isolating their individual effect is difficult. Noise is commonly measured in overall sound pressure levels (SPL) expressed in so-called A-

weighted decibels (dBA)²⁰ that aggregate all sound energy in an acoustic scene. Existing technologies are unable to isolate the effect of offending sources, especially in urban environments flooded with multiple sounds. As a result, inspectors resort to long, complicated measurement strategies that often require help from the people responsible for the violation in the first place, an additional factor contributing to the difficulty and reduced efficiency of the enforcement process.

Here, we outline the opportunities and challenges associated with SONYC, our cyber-physical systems approach to the monitoring, analysis, and mitigation of urban noise pollution. Connecting various subfields of computing, including wireless sensor networks, machine learning, collaborative and social computing, and computer graphics, it creates a potentially transformative solution to this important quality-of-life issue affecting millions of people worldwide. To illustrate this potential, we present findings from an initial study we con-

ducted in 2017 showing how SONYC can help understand and address important gaps in the process of urban noise mitigation.

SONYC

Multiple research projects have sought to create technological solutions to improve the cycle of urban noise pollution. For example, some have used mobile devices to crowdsource instantaneous SPL measurements, noise labels, and subjective responses^{3,24,28} but generally lag well behind the coverage in space-time of civic complaint systems like 311, while the reliability of their objective measurements suffers from a lack of adequate calibration. Others have deployed static-sensing solutions that are often too costly to scale up or go beyond the capabilities of standard noise meters.^{4,23,29} On the analytical side, a significant amount of work has focused on noise maps generated from sound propagation models for major urban noise sources (such as industrial activity and road, rail, and air traffic).^{13,17} However, these maps

lack temporal dynamics and make modeling assumptions that often render them too inaccurate to support mitigation or action planning.¹ Few of these initiatives involve acting on the sensed or modeled data to affect noise emissions, and even fewer have included participation from local governments.¹⁵

SONYC (Sounds of New York City), our novel solution, as outlined in Figure 1, aims to address these limitations through an integrated cyber-physical systems' approach to noise pollution.

First, it includes a low-cost, intelligent sensing platform capable of continuous, real-time, accurate, source-specific noise monitoring. It is scalable in terms of coverage and power consumption, does not suffer from the same biases as 311-style reporting, and goes well beyond SPL-based measurements of the acoustic environment. Second, SONYC adds new layers of cutting-edge data-science methods for large-scale noise analysis, including predictive noise modeling in off-network locations using spatial statistics and physical modeling, development of interactive 3D visualizations of noise activity across time and space to enable better understanding of noise patterns, and novel information-retrieval tools that exploit the topology of noise events to facilitate search and discovery. And third, it uses this sensing and analysis framework to improve mitigation in two ways—first by enabling optimized, data-driven planning and scheduling of inspections by the local government, thus making it more likely code violations will be detected and enforced; and second, by increasing the flow of information to those in a position to control emissions (such as building and construction-site managers, drivers, and neighbors) thus providing credible incentives for self-regulation. Because the system is constantly monitoring and analyzing noise pollution, it generates information that can be used to validate, and iteratively refine, any noise-mitigating strategy.

Consider a scenario in which a system integrates information from the sensor network and 311 to identify a pattern of after-hours jackhammer activity around a construction site. This information triggers targeted inspections by the DEP that results in

an inspector issuing a violation. Statistical analysis can then be used by researchers or city officials to validate whether the action is short-lived in time or whether its effect propagates to neighboring construction sites or distant ones by the same company. By systematically monitoring interventions, inspectors can understand how often penalties need to be issued before the effect becomes long term. The overarching goal is to understand how to minimize the cost of interventions while maximizing noise mitigation, a classic resource-allocation problem that motivates much research in smart-cities initiatives.

All this is made possible by formulating our solution in terms of a cyber-physical system. However, unlike most cyber-physical systems covered in the literature, the distributed and decentralized nature of the noise-pollution problem requires multiple socioeconomic incentives (such as fines and peer comparisons) to exercise indirect control over tens of thousands of subsystems contributing noise emissions. It also calls for developing and implementing a set of novel mechanisms for integrating humans in the cyber-physical system loop at scale and at multiple levels of the system's management hierarchy, including extensive use of human-computer interaction (HCI) research in, say, citizen science and data visualization, to fa-

cilitate seamless interaction between humans and cyber-infrastructure. Worth emphasizing is that this line of work is fundamentally different from current research on human-in-the-loop cyber-physical systems that often focuses on applications in which control is centralized and fully or mostly automated while usually only a single human is involved (such as in assistive robots and intelligent prosthetics). The synthesis of approaches from social computing, citizen science, and data science to advance integration, management, and control of large and variable numbers of human agents in cyber-physical systems is potentially transformative, addressing a crucial bottleneck for the widespread adoption of similar methods in all kinds of socio-technical systems, including transportation networks, power grids, smart buildings, environmental control, and smart cities.

Finally, SONYC uses New York City, the largest, densest, noisiest city in North America, as its test site. The city has long been at the forefront of discussions about noise pollution, has an exemplary noise code^b and, in 311, the most comprehensive citizen noise-reporting system. Beyond noise, the city collects vast amounts of data about everything from public

^b <http://www.nyc.gov/html/dep/html/noise/index.shtml>

Figure 2. Acoustic sensing unit deployed on a New York City street.



safety, traffic, and taxi activity to construction, making much of it publicly available.^c Our work involves close collaboration with city agencies, including DEP, DOHMH, various business improvement districts, and private initiatives (such as LinkNYC) that provide access to existing infrastructure. As a powerful sensing-and-analysis infrastructure, SONYC thus holds the potential to empower new research in environmental psychology, public health, and public policy, as well as empower citizens seeking to improve their own communities. We next describe the technology and methods underpinning the project, presenting some of our early findings and future challenges.

Acoustic Sensor Network

As mentioned earlier, SONYC's intelligent sensing platform should be scalable and capable of source identification and high-quality, round-the-clock noise monitoring. To that end we have developed an acoustic sensor¹⁸ (see Figure 2) based on the popular Raspberry Pi single-board computer outfitted with a custom microelectromechanical systems (MEMS) microphone module. We chose MEMS microphones for their low cost and consistency across units and size, which can be 10x smaller than conventional microphones. Our custom standalone microphone module includes additional circuitry, including in-house analog-to-digital converters and pre-amp stages, as well as an on-board microcontroller that enables preprocessing of the incoming audio signal to compensate for the microphone's frequency response. The digital MEMS microphone features a wide dynamic range of 32dBA–120dBA, ensuring all urban sound pressure levels are monitored effectively. We calibrated it using a precision-grade sound-level meter as reference under low-noise anechoic conditions and was empirically shown to produce sound-pressure-level data at an accuracy level compliant with the ANSI Type-2 standard²⁰ required by most local and national noise codes.

The sensor's computing core is

housed in an aluminum casing we chose to reduce RFI interference and solar heat gain. The microphone module is mounted externally via a flexible metal gooseneck attachment, making it possible to reconfigure the sensor node for deployment in varying locations, including sides of buildings, light poles, and building ledges. Apart from continuous SPL measurements, we designed the nodes to sample 10-second audio snippets at random intervals over a limited period of time, collecting data to train and benchmark our machine-listening solutions. SONYC compresses the audio using the lossless FLAC audio coding format, using 4,096-bit AES encryption and the RSA public/private key-pair encryption algorithm. Sensor nodes communicate with the server via a virtual private network, uploading audio and SPL data at one-minute intervals.

As of December 2018, the parts of each sensor cost approximately \$80 using mostly off-the-shelf components. We fully expect to reduce the unit cost significantly through custom redesign for high-volume, third-party assembly. However, even at the current price, SONYC sensors are significantly more affordable, and thus amenable to large-scale deployment, than existing noise-monitoring solutions. Moreover, this reduced cost does not come at the expense of measurement accuracy, with our sensors' performance comparable to high-quality devices that are orders of magnitude more costly while outperforming solutions in the same price range. Finally, the dedicated computing core opens the possibility for edge computing, particularly for in-situ machine listening intended to automatically and robustly identify the presence of common sound sources. This unique feature of SONYC goes well beyond the capabilities of existing noise-monitoring solutions.

Machine Listening at the Edge

Machine listening is the auditory counterpart to computer vision, combining techniques from signal processing and machine learning to develop systems able to extract meaningful information from sound. In the context of SONYC, we focus on developing computational

methods to automatically detect specific types of sound sources (such as jackhammers, idling engines, car horns, and police sirens) from environmental audio. Detection is a challenge, given the complexity and diversity of sources, auditory scenes, and background conditions routinely found in noisy urban acoustic environments.

We thus created an urban sound taxonomy, annotated datasets, and various cutting-edge methods for urban sound-source identification.^{25,26} Our research shows that feature learning, using even simple dictionary-based methods (such as spherical k-means) makes for significant improvement in performance over the traditional approach of feature engineering. Moreover, we have found that temporal-shift invariance, whether through modulation spectra or deep convolutional networks, is crucial not only for overall accuracy but also to increase robustness in low signal-to-noise-ratio (SNR) conditions, as when sources of interest are in the background of acoustic scenes. Shift invariance also results in more compact machines that can be trained with less data, thus adding greater value for edge-computing solutions. More recent results highlight the benefits of using convolutional recurrent architectures, as well as ensembles of various models via late fusion.

Deep-learning models necessitate large volumes of labeled data traditionally unavailable for environmental sound. Addressing this lack of data, we have developed an audio data augmentation framework that systematically deforms the data using well-known audio transformations (such as time stretching, pitch shifting, dynamic range compression, and addition of background noise at different SNRs), significantly increasing the amount of data available for model training. We also developed an open source tool for soundscape synthesis.²⁷ Given a collection of isolated sound events, it functions as a high-level sequencer that can generate multiple soundscapes from a single probabilistically defined "specification." We generated large datasets of perfectly annotated data in order to assess algorithmic performance as a function of, say, maximum polyphony and SNR ratio,

c <https://nycopendata.socrata.com>

studies that would be prohibitive at this scale and precision using manually annotated data.

The combination of an augmented training set and increased capacity and representational power of deep-learning models yields state-of-the-art performance. Our current machine-listening models can perform robust multi-label classification for 10 common classes of urban sound sources in real time running on a laptop. We will soon adapt them to run under the computational constraints of the Raspberry Pi.

However, despite the advantages of data augmentation and synthesis, the lack of a significant amount of annotated data for supervised learning remains the main bottleneck in the development of machine-listening solutions that can detect more sources of noise. To address this need, we developed a framework for Web-based human audio annotation and conducted a large-scale, experimental study on how visualization aids and acoustic conditions affect the annotation process and its effectiveness.⁶ We aimed to quantify the reliability/redundancy trade-off in crowdsourced soundscape annotation, investigate how visualizations affect accuracy and efficiency, and characterize how performance varies as a function of audio characteristics. Our study followed a between-subjects factorial experimental design in which we tested 18 different experimental conditions with 540 participants we recruited through Amazon's Mechanical Turk.

We found more complex audio scenes result in lower annotator agreement and that spectrogram visualizations are superior at producing higher-quality annotations at lower cost in terms of time and human labor. Given enough time, all tested visualization aids enable annotators to identify sound events with similar recall, but the spectrogram visualization enables annotators to identify sounds more quickly. We speculate this may be because annotators are able to more easily identify visual patterns in the spectrogram, in turn enabling them to identify sound events and their boundaries more precisely and efficiently. We also found participants learn to use each interface more ef-



It is scalable in terms of coverage and power consumption, does not suffer from the same biases as 311-style reporting, and goes well beyond SPL-based measurements of the acoustic environment.



fectively over time, suggesting we can expect higher-quality annotations with only a small amount of additional training.

We found the value of additional annotators decreased after five to 10 annotators and that having 16 annotators was sufficient for capturing 90% of the gain in annotation quality. However, when resources are limited and cost is a concern, our findings suggest five annotators may be a reasonable choice for reliable annotation with respect to the trade-off between cost and quality. These findings are valuable for the design of audio-annotation interfaces and the use of crowdsourcing and citizen science strategies for audio annotation at scale.

Noise Analytics

One main SONYC promise is its future ability to analyze and understand noise pollution at city-scale in an interactive and efficient manner. As of December 2018, we had deployed 56 sensors, primarily in the city's Greenwich Village neighborhood, as well as in other locations in Manhattan, Brooklyn, and Queens. Collectively, the sensors have gathered the equivalent of 30 years of audio data and more than 60 years of sound-pressure levels and telemetry. These numbers are a clear indication of the magnitude of the challenge from a data-analytics perspective.

We are currently developing a flexible, powerful visual-analytics framework that enables visualization of noise levels in the context of the city, together with other related urban data streams. Working with urban data poses further research challenges. Although much work has focused on scaling databases for big data, existing data-management technologies do not meet the requirements needed to interactively explore massive or even reasonable-size datasets.⁸

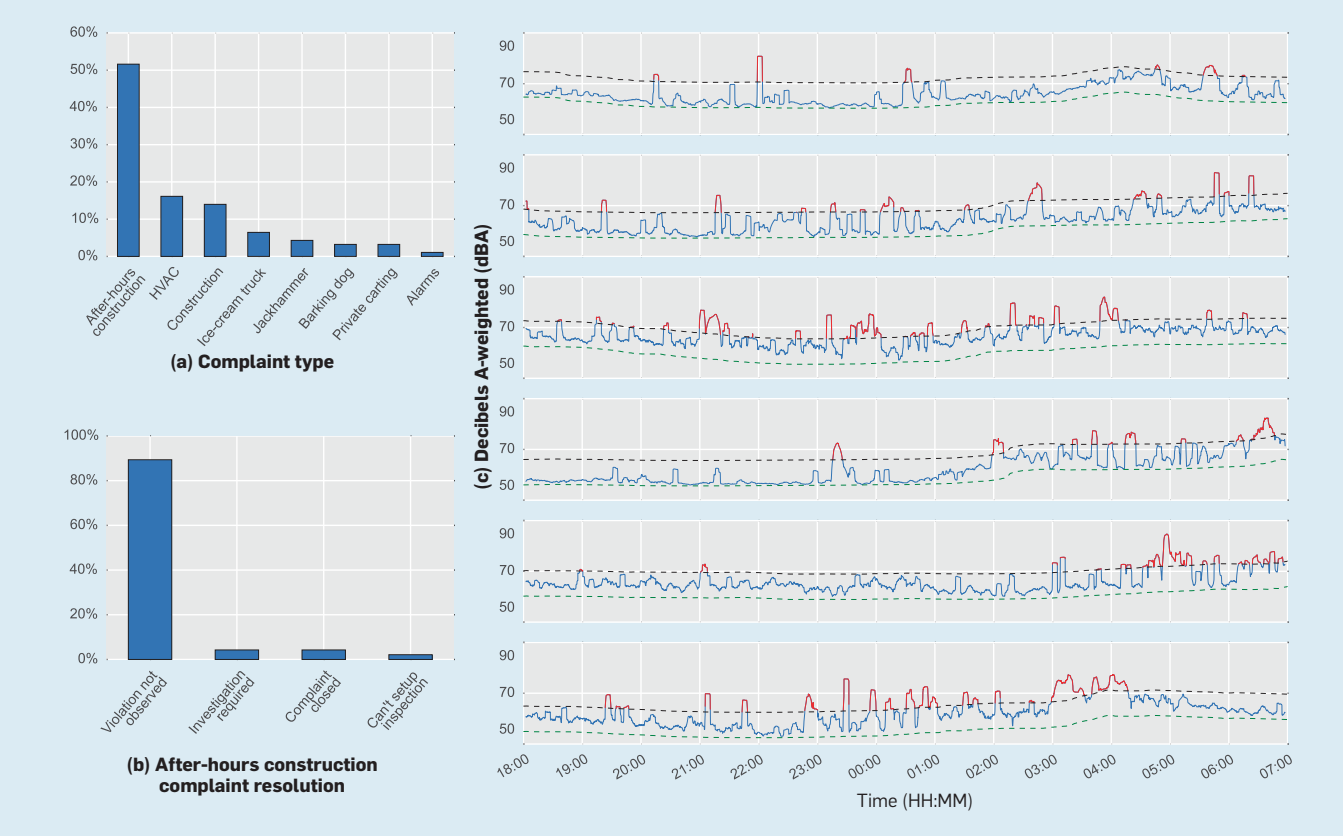
Accomplishing interactivity requires not only efficient techniques for data and query management but for scalable visualization techniques capable of rendering large amounts of information.

In addition, visualizations and interfaces must be rendered in a form that is easily understood by domain experts and non-expert users alike, in-

Figure 3. (left) Interactive 3D visualization of a New York neighborhood using Urbane. By selecting specific sensors (red pins) and buildings (purple) researchers can retrieve and visualize multiple data streams associated with these locations. (right) SPL data at various resolutions and time scales retrieved using the time lattice. Each sub-figure reflects different individual (gray) and aggregated (red) sensor data for the three sensor units highlighted in the left plot.



Figure 4. Case study involving the area around Washington Square Park: (a) Distribution of 311 outdoor noise complaints in the focus area during the study period; the bar graph shows clear predominance of after-hours construction noise. (b) Distribution of complaint resolution for after-hours construction complaints; almost all complaints result in “violation not observed” status. (c) Sensor data for the after-hours period corresponding to six complaints: continuous SPL data (blue), background level (green), event-detection threshold at 10dB above background level (black), and potential noise code violation events (red).



cluding crowdsourcing workers and volunteers, and bear meaningful relationship to the properties of the data in the physical world that, in the case of sound, implies the need for three-dimensional visualization.

We have been working on a three-dimensional, urban geographic information system (GIS) framework called Urbane⁹ (see Figure 3), an interactive tool, including a novel three-dimensional map layer, we developed from the ground up to take advantage of the GPU capabilities of modern computing systems. It allows for fast, potentially real-time computation, as well as integration and visualization of multiple data streams commonly found in major cities like New York City. In the context of SONYC, we have expanded Urbane's capabilities to include efficient management of high-resolution temporal data. We achieve this efficiency through a novel data structure we call the "time lattice" that allows for fast retrieval, visualization, and analysis of individual and aggregate sensor data at multiple time scales (such as hours, days, weeks, and months). An example of data retrieved through this capability can be seen in Figure 3, right plot. We have since used Urbane and the time lattice to support the preliminary noise analysis we cover in the next section, but their applicability goes well beyond audio.

We are currently expanding Urbane to support visual spatiotemporal queries over noise data, including computational-topology methods for pattern detection and retrieval. Similar tools have proved useful in smart-cities research projects, including prior collaborations between team members and the New York City Department of Transportation and Taxi and Limousine Commission.^{7,10}

Data-Driven Mitigation

We conducted a preliminary study in 2017 on the validity and response of noise complaints around the Washington Square Park area of Manhattan using SONYC's sensing and analytics infrastructure.¹⁹ The study combined information mined from the log of civic complaints made to the city over the study period through the 311 system,

the analysis of a subset of our own sensor data during the same period, and information gathered through interactions and site visits with inspectors from the DEP tasked with enforcing the city's noise code.

For the study we chose an area in Greenwich Village with a relatively dense deployment of 17 nodes. We established a 100-meter boundary around each node and merged them to form the focus area. From 311, we collected all non-duplicate noise complaints occurring within this area that had been routed to the DEP while neighboring sensors were active. Note this criterion discards complaints about noise from residents that are routed to the police department and tend to dominate the 311 log; see Figure 4a for a breakdown of selected complaint types.

Over an 11-month period—May 2016 to April 2017—51% of all noise complaints in the focus area were related to after-hours construction activity (6 P.M.–7 A.M.), three times the amount in the next category. Note combining all construction-related complaints adds up to 70% of this sample, highlighting how disruptive to the lives of ordinary citizens this particular category of noise can be.

Figure 4c includes SPL values (blue line) at a five-minute resolution for the after-hours period during or immediately preceding a subset of the complaints. Dotted green lines correspond to background levels, computed as the moving average of SPL measurements within a two-hour window. Dotted black lines correspond to SPL values 10dB above the background, the threshold defined by the city's noise code to indicate potential violations. Finally, we were able to identify events (in red) in which instantaneous SPL measurements were above the threshold. Our analysis resulted in detection of 324 such events we classified by noise source and determined 76% (246) were related to construction as follows: jackhammering (223), compressor engines (16), metallic banging/scraping (7), and the remainder to non-construction sources, mainly sirens and other traffic noise. Our analysis found for 94% of all after-hours construction complaints quantitative evidence in our

sensor data of a potential violation.

How does this evidence stack up against the enforcement record for the complaints? Citizen complaints submitted via 311 and routed to the DEP trigger an inspection, and public-record repositories made available by the city include information about how each complaint was resolved. Examining the records, we found that, for all complaints in this study, 78% resulted in a "No violation could be observed" status and only 2% in a violation ticket being issued. Figure 4b shows, in the specific case of after-hours construction noise, no violation could be observed in 89% of all cases, and none of the inspections resulted in a violation ticket being issued.

There are multiple possible explanations for the significant gap between the evidence collected by the sensor network and the results of the inspections. For example, we speculate it is due in part to the delay in the city's response to complaints, four to five days on average, which is too great for phenomena that are both transient and traceless. Another factor is the conspicuousness of the inspection crew that alone modifies the behavior of potentially offending sources, as we observed during our site visits with the DEP. Moreover, under some circumstances the city government grants special, after-hours construction permits under the assumption of minimal noise impact, as defined by the noise code. It is thus possible that some after-hours activity results from such permits. We are currently mining after-hours-construction-permit data to understand this relationship better.

In all cases, the SONYC sensing and analytical framework is able to address the shortcomings of current monitoring and enforcement mechanisms by providing hard data to: quantify the actual impact of after-hours construction permits on the acoustic environment, and thus nearby residents; provide historical data that can validate complaints and thus support inspection efforts on an inconspicuous and continuous basis; and develop novel, data-driven strategies for the efficient allocation of inspection crews in space and

time using the same tools from operations research that optimize routes for delivery trucks and taxis. Worth noting is that, even though our preliminary study focused on validating 311 complaints, SONYC can be used to gain insight beyond complaint data, allowing researchers and city officials to understand the extent and type of unreported noise events, identify biases in complaint behavior, and accurately measure the level of noise pollution in the local environment.

Looking Forward

The SONYC project is currently in the third of five years of its research and development agenda. Its initial focus was on developing and deploying intelligent sensing infrastructure but has progressively shifted toward analytics and mitigation in collaboration with city agencies and other stakeholders. Here are some areas we intend to address in future work:

Low-power mesh sensor network. To support deployment of sensors at significant distances from Wi-Fi or other communication infrastructure and at locations lacking ready access to electrical power, we are developing a second generation of the sensor node to be mesh-enabled and battery/solar powered. Each sensor node will serve as a router in a low-power multi-hop wireless network in the 915MHz band, using FCC-compatible cognitive radio techniques over relatively long links and energy-efficient multi-channel routing for communicating to and from infrastructure-connected base stations. The sensor design will further reduce power consumption for multi-label noise classification by leveraging heterogeneous processors for duty-cycled/event-driven hierarchical computing. Specifically, the design of the sensor node will be based on a low-power system-on-chip—the Ineda i7^d—for which we are redesigning “mote-scale” computation techniques originally developed for single microcontroller devices to support heterogeneous processor-specific operating systems via hardware virtualization.

Modeling. The combination of noise data collected by sensors and



The dedicated computing core opens the possibility for edge computing, particularly for in-situ machine listening intended to automatically and robustly identify the presence of common sound sources.



citizens will necessarily be sparse in space and time. In order to perform meaningful analyses and help inform decisions by city agencies, it is essential for the system to compensate for this sparseness. Several open datasets are available that could, directly or indirectly, provide information on the noise levels in the city; for example, locations of restaurants, night clubs, and tourist attractions indicate areas where sources of social noise are likely, while social media data streams can be used to understand the temporal dynamics of crowd behavior. Likewise, multiple data streams associated with taxi, bus, and aircraft traffic can provide indirect information on traffic-based noise levels. We plan to develop noise models that use spatiotemporal covariance to predict unseen acoustic responses through a combination of sensor and open data. We will also explore combinations of data-driven modeling, applying physical models that exploit the three-dimensional geometry of the city, sound type and localization cues from sensors and 311, and basic principles of sound propagation. We expect that through a combination of techniques from data mining, statistics, and acoustics, as well as our own expertise developing models suitable for GPU implementation using ray-casting queries in the context of computer graphics, we will be able to create accurate, dynamic, three-dimensional urban noise maps in real time.

Citizen science and civic participation. The role of humans in SONYC is not limited to annotating sound. In addition to the fixed sensors located in various parts of the city, we will be designing a SONYC mobile platform aimed at enabling ordinary citizens to record and annotate sounds in situ, view existing data contributed and analyzed by others, and contact city authorities about noise-related concerns. A mobile platform will allow them to leverage slices taken from this rich dataset to describe and support these concerns with evidence as they approach city authorities, regulators, and policymakers. Citizens will not only be more informed and engaged with their envi-

^d <http://inedasystems.com/wearables.php>

ronment, they will be better equipped to voice their concerns when interacting with city authorities.

Conclusion

SONYC is a smart-cities, next-generation application of a cyber-physical system. Its development calls for innovation in various fields of computing and engineering, including sensor networks, machine learning, human-computer interaction, citizen science, and data science. The technology will be able to support novel scholarly work on the effects of noise pollution on public health, public policy, environmental psychology, and economics. But the project is far from purely scholarly. By seeking to improve urban-noise mitigation, a critical quality-of-life issue, SONYC promises to benefit urban citizens worldwide. Our agenda calls for the system to be deployed, tested, and used in real-world urban conditions, potentially resulting in a model that can be scaled and replicated throughout the U.S. and beyond.

Acknowledgments

This work is supported in part by the National Science Foundation (Award # 1544753), NYU's Center for Urban Science and Progress, NYU's Tandon School of Engineering, and the Translational Data Analytics Institute at The Ohio State University. 

References

- Ausejo, M., Recuero, M., Asensio, C., Pavón, I., and Pagan, R. Study of uncertainty in noise mapping. In *Proceedings of 39th International Congress on Noise Control Engineering, Internoise* (Lisbon, Portugal, June 13–16). Portuguese Acoustical Society, Lisbon, 2010, 6210–6219.
- Basner, M., Babisch, W., Davis, A., Brink, M., Clark, C., Janssen, S., and Stansfeld, S. Auditory and non-auditory effects of noise on health. *The Lancet* 383, 9925 (Apr. 2014), 1325–1332.
- Becker, M., Caminiti, S., Fiorella, D., Francis, L., Gravino, P., Haklay, M. M., Hotho, A., Loreto, V., Mueller, J., Ricchiuti, F. et al. Awareness and learning in participatory noise sensing. *PLoS One* 8, 12 (Dec. 2013), 1–12.
- Bell, M.C. and Galatioto, F. Novel wireless pervasive sensor network to improve the understanding of noise in street canyons. *Applied Acoustics* 74, 1 (Jan. 2013), 169–180.
- Bronzaft, A. and Van Ryzin, G. *Neighborhood Noise and Its Consequences: Implications for Tracking Effectiveness of NYC Revised Noise Code. Special Report #14*. Survey Research Unit, School of Public Affairs, Baruch College, CUNY, New York, Apr. 2007; <http://www.noiseoff.org/document/cenyc.noise.report.14.pdf>
- Cartwright, M., Seals, A., Salamon, J., Williams, A., Mikloska, S., McConnell, D., Law, E., Bello, J., and Nov, O. Seeing sound: Investigating the effects of visualizations and complexity on crowdsourced audio annotations. In *Proceedings of the 21st ACM Conference on Computer-Supported Cooperative Work and Social Computing* (Jersey City, NJ, Nov. 3–7). ACM Press, New York, 2018, 29:1–29:21.
- Doraiswamy, H., Ferreira, N., Damoulas, T., Freire, J., and Silva, C. T. Using topological analysis to support event-guided exploration in urban data. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (Dec. 2014), 2634–2643.
- Fekete, J.-D. and Silva, C. Managing data for visual analytics: Opportunities and challenges. *IEEE Data Engineering Bulletin* 35, 3 (Sept. 2012), 27–36.
- Ferreira, N., Lage, M., Doraiswamy, H., Vo, H., Wilson, L., Werner, H., Park, M.C., and Silva, C. Urbane: A 3D framework to support data-driven decision making in urban development. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology* (Chicago, IL, Oct. 25–30), 2015, 97–104.
- Ferreira, N., Poco, J., Vo, H.T., Freire, J., and Silva, C.T. Visual exploration of big spatiotemporal urban data: A study of New York City taxi trips. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (Dec. 2013), 2149–2158.
- Fritsch, L., Brown, L., Kim, R., Schwela, D., and Kephapoulos, S. Burden of disease from environmental noise: Quantification of healthy years life lost in Europe. World Health Organization, Bonn, Germany, 2012; <http://www.euro.who.int/en/publications/abstracts/burden-of-disease-from-environmental-noise-quantification-of-healthy-life-years-lost-in-europe>
- Hammer, M.S., Swinburn, T.K., and Neitzel, R.L. Environmental noise pollution in the United States: Developing an effective public health response. *Environmental Health Perspectives* 122, 2 (Feb. 2014), 115–119.
- Kaliski, K., Duncan, E., and Cowan, J. Community and regional noise mapping in the United States. *Sound and Vibration* 41, 9 (Sept. 2007), 12.
- Maibach, M., Schreyer, C., Sutter, D., Van Essen, H., Boon, B., Smokers, R., Schrotten, A., Doll, C., Pawlowska, B., and Bak, M. *Handbook on estimation of external costs in the transport sector*. CE Delft, Feb. 2008; https://ec.europa.eu/transport/sites/transport/files/themes/sustainable/doc/2008_costs_handbook.pdf
- Manvell, D., Marcos, L.B., Stapelfeldt, H., and Sanz, R. SADMAM—Combining measurements and calculations to map noise in Madrid. In *Proceedings of the 33rd Congress and Exposition on Noise Control Engineering (Internoise)* (Prague, Czech Republic, Aug. 22–25). Institute of Noise Control Engineering, Reston, VA, 2004.
- Mizutani, F., Suzuki, Y., and Sakai, H. Estimation of social costs of transport in Japan. *Urban Studies* 48, 16 (Apr. 2011), 3537–3559.
- Murphy, E. and King, E. Strategic environmental noise mapping: Methodological issues concerning the implementation of the EU Environmental Noise Directive and their policy implications. *Environment International* 36, 3 (Apr. 2010), 290–298.
- Mydlarz, C., Salamon, J., and Bello, J. The implementation of low-cost urban acoustic monitoring devices. *Applied Acoustics, Special Issue on Acoustics for Smart Cities* 117, B (Feb. 2017), 207–218.
- Mydlarz, C., Shamoon, C., and Bello, J. Noise monitoring and enforcement in New York City using a remote acoustic sensor network. In *Proceedings of the INTER-NOISE and NOISE CON Congress and Conference* (Hong Kong, China, Aug. 27–30). Institute of Noise Control Engineering, Reston, VA, 2017.
- National Academy of Engineering. *Technology for a Quieter America: NAEPR-06-01-A. Technical Report*. The National Academies Press, Washington, D.C., Sept. 2010; <https://www.nap.edu/catalog/12928/Technology-for-a-quieter-america>
- Nelson, J. P. Highway noise and property values: A survey of recent evidence. *Journal of Transport Economics and Policy* 16, 2 (May 1982), 117–138.
- New York City Department of Health and Mental Hygiene. *Ambient Noise Disruption in New York City, Data Brief 45*. New York City Department of Health and Mental Hygiene, Apr. 2014; <https://www1.nyc.gov/assets/doh/downloads/pdf/epi/databrief45.pdf>
- Pham, C. and Cousin, P. Streaming the sound of smart cities: Experimentations on the SmartSantander test-bed. In *Proceedings of IEEE International Conference on Green Computing and Communications, IEEE Internet of Things, and IEEE Cyber, Physical and Social Computing* (Beijing, China, Aug. 20–23). IEEE, Piscataway, NJ, 2013, 611–618.
- Ruge, L., Altakroui, B., and Schrader, A. Sound of the city: Continuous noise monitoring for a healthy city. In *Proceedings of the IEEE International Conference on*

Pervasive Computing and Communications Workshops (San Diego, CA, Mar. 18–22). IEEE, Piscataway, NJ, 670–675.

- Salamon, J. and Bello, J. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters* 24, 3 (Mar. 2017), 279–283.
- Salamon, J., Jacoby, C., and Bello, J.P. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM International Conference on Multimedia* (Orlando, FL, Nov. 3–7). ACM Press, New York, 2014.
- Salamon, J., McConnell, D., Cartwright, M., Li, P., and Bello, J. SCAPER: A library for soundscape synthesis and augmentation. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (Mohonk, New Paltz, NY, Oct. 15–18). IEEE, Piscataway, NJ, 2017.
- Schweizer, I., Meurisch, C., Gedeon, J., Bärthel, R., and Mühlhäuser, M. Noisemap: Multi-tier incentive mechanisms for participative urban sensing. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones* (Toronto, ON, Canada, Nov. 6–9). ACM Press, New York, 2012, 9.
- Steele, D., Krijnders, D., and Guastavino, C. The Sensor City Initiative: Cognitive sensors for soundscape transformations. In *Proceedings of GIS Ostrava 2013: Geoinformatics for City Transformation* (Ostrava, Czech Republic, Jan. 21–23). Technical University of Ostrava, 2013.
- Theebe, M.A. Planes, trains, and automobiles: The impact of traffic noise on house prices. *The Journal of Real Estate Finance and Economics* 28, 2–3 (Mar. 2004), 209–234.

Juan Pablo Bello (jpbello@nyu.edu) is a professor of music technology and computer science and engineering at New York University, New York, USA, and director of the Center for Urban Science of Progress and of the Music and Audio Research Laboratory.

Claudio Silva (csilva@nyu.edu) is a professor of computer science and engineering and data science at New York University, New York, USA.

Oded Nov (onov@nyu.edu) is an associate professor of technology management and innovation at New York University, New York, USA.

R. Luke DuBois (dubois@nyu.edu) is co-director and an associate professor of integrated digital media at New York University, New York, USA.

Anish Arora (arora.9@osu.edu) is a professor of computer science and engineering at The Ohio State University, Columbus, OH, USA.

Justin Salamon (justin.salamon@nyu.edu) is a senior research scientist at the Music and Audio Research Laboratory and the Center for Urban Science and Progress at New York University, New York, USA.

Charles Mydlarz (cmydlarz@nyu.edu) is a senior research scientist at the Music and Audio Research Laboratory and the Center for Urban Science and Progress at New York University, New York, USA.

Harish Doraiswamy (harishd@nyu.edu) is a research assistant professor of computer science and engineering and a research scientist at the Center for Data Science at New York University, New York, USA.

Copyright held by authors.



Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/sonyc>

The roots of blockchain technologies are deeply interwoven in distributed computing.

BY MAURICE HERLIHY

Blockchains from a Distributed Computing Perspective

BITCOIN FIRST APPEARED in a 2008 white paper authored by someone called Satoshi Nakamoto,¹⁸ the mysterious *deus absconditus* of the blockchain world. Today, cryptocurrencies and blockchains are very much in the news. Much of this coverage is lurid, sensationalistic, and irresistible: roller-coaster prices and instant riches, vast sums of money stolen or inexplicably lost, underground markets for drugs and weapons, and promises of libertarian utopias just around the corner.

This article is a tutorial on the basic notions and mechanisms underlying blockchains, colored by the perspective that much of the blockchain world is a disguised, sometimes distorted, mirror image of the distributed^a computing world.

^a In this article, “distributed computing” is used to encompass both message passing and shared-memory models of concurrent computation.

This article is not a technical manual, nor is it a broad survey of the literature (both widely available elsewhere). Instead, it attempts to explain blockchain research in terms of the many similarities, parallels, semi-reinventions, and lessons not learned from distributed computing.

This article is intended mostly to appeal to blockchain novices, but perhaps it will provide some insights to those familiar with blockchain research but less familiar with its precursors.

The Ledger Abstraction

The abstraction at the heart of blockchain systems is the notion of a *ledger*, an invention of the Italian Renaissance originally developed to support double-entry bookkeeping, a distant precursor of modern cryptocurrencies. For our purposes, a ledger is just an indelible, append-only log of transactions that take place between various parties. A ledger establishes which transactions happened (“Alice transferred 10 coins to Bob”), and the order in which those transactions happened (“Alice transferred 10 coins to Bob, and then Bob transferred title to his car to Alice”). Ledgers are public, accessible to all parties, and they must be tamper-proof: no party can add, delete, or modify ledger entries once they have been recorded. In short, the algorithms that maintain ledgers must be immune to attack, ensuring the ledger remains secure even

» key insights

- The long-term scientific value of blockchain algorithms and systems is independent of the fates of today's coins.
- Many of the basic algorithms and techniques used in blockchains are best understood as variations on familiar algorithms and techniques from classic distributed computing.
- A smart contract language should have an explicit concurrency model to make programmers aware of well-known concurrency-related pitfalls and hazards.
- The blockchain world encompasses both “permissioned” and “permissionless” chains, and a number of promising application areas beyond just coins.



if some parties misbehave, whether accidentally or maliciously.

Blockchain ledger precursors. It is helpful to start by reviewing a blockchain precursor, the *so-called universal construction* for lock-free data structures.¹³

Alice runs an online news service. Articles that arrive concurrently on multiple channels are placed in an in-memory table where they are indexed for retrieval. At first, Alice used a lock to synchronize concurrent access to the table, but every now and then, the thread holding the lock would take a page fault or a scheduling interrupt, leaving the articles inaccessible for too long. Despite the availability of excellent textbooks on the subject,¹⁴ Alice was uninterested in customized lock-free algorithms, so she was in need of a simple way to eliminate lock-based vulnerabilities.

She decided to implement her data structure in two parts. To record articles as they arrive, she created a ledger implemented as a simple linked list, where each list entry includes the article and a link to the entry before it. When an article arrives, it is placed in a shared pool, and a set of dedicated threads, called *miners* (for reasons to be explained later), collectively and repeatedly run a protocol, called consensus, to select which article to append to the ledger. Here, Alice's consensus protocol can be simple: each thread creates a list entry, then calls an atomic compare-and-swap^b instruction to attempt to make that entry the new head of the list.

Glossing over some technical details, to query for a recent article, a thread scans the linked-list ledger. To add a new article, a thread adds the article to the pool, and waits for a miner to append it to the ledger.

This use of a black-box consensus protocol may seem cumbersome, and indeed, there are many ways it could be made more efficient, but it has two compelling advantages even without further optimization: First, it is universal: it can implement any type of data structure, no matter how complex. Second, all questions of concurrency and



Cryptocurrencies and blockchains are very much in the news. Much of the coverage is lurid, sensationalistic, and irresistible.



fault-tolerance are compartmentalized in the consensus protocol.

A *consensus protocol* involves a collection of parties, some of whom are honest, and follow the protocol, and some of whom are dishonest, and may depart from the protocol for any reason. Consensus is a notion that applies to a broad range of computational models. In some contexts, dishonest parties might simply halt arbitrarily (so-called crash failures), while in other contexts, they may behave arbitrarily, even maliciously (so-called Byzantine failures). In some contexts, parties communicate through objects in a shared memory, and in others, they exchange messages. Some contexts restrict how many parties may be dishonest, some do not.

In consensus, each party proposes a transaction to append to the ledger, and one of these proposed transactions is chosen. Consensus ensures *agreement*: All honest parties agree on which transaction was selected, *termination*: All honest parties eventually learn the selected transaction, and *validity*: The selected transaction is valid for that application.

Consensus protocols have been the focus of decades of research in the distributed computing community. The literature contains many algorithms and impossibility results for many different models of computation (see surveys in Attiya¹ and Herlihy¹⁴).

Because ledgers are long-lived, they require the ability to do *repeated* consensus to append a stream of transactions to the ledger. Usually, consensus is organized in discrete rounds, where parties start round $r+1$ after round r is complete. Of course, this shared-memory universal construction is not yet a blockchain, because although it is concurrent, it is not distributed. Moreover, it does not tolerate truly malicious behavior (only crashes). Nevertheless, we have already introduced the key concepts underlying blockchains.

Private blockchain ledgers. Alice also owns a frozen yogurt parlor, and her business is in trouble. Several recent shipments of frozen yogurt have been spoiled, and Bob, her supplier, denies responsibility. When she sued, Bob's lawyers successfully pleaded that not only had Bob never handled

^b The *compare-and-swap* instruction atomically compares a memory location's contents with a given expected value and, if they match, updates that location's contents to a new given value.

those shipments, but they were spoiled when they were picked up at the yogurt factory, and they were in excellent condition when delivered to Alice's emporium.

Alice decides it is time to “block-chain” her supply chain. She rents some cloud storage to hold the ledger, and installs Internet-enabled temperature sensors in each frozen yogurt container. She is concerned that sensors are not always reliable (and that Bob may have tampered with some), so she wires the sensors to conduct a *Byzantine fault-tolerant consensus* protocol,⁴ which uses several rounds of voting to ensure temperature readings cannot be distorted by a small number of faulty or corrupted sensors. At regular intervals, the sensors reach consensus on the current temperature. They timestamp the temperature record, and add a hash of the prior record, so that any attempt to tamper with earlier records will be detected when the hashes do not match. They sign the record to establish authenticity, and then append the record to the cloud storage's list of records.

Each time a frozen yogurt barrel is transferred from Carol's factory to Bob's truck, Bob and Carol sign a statement agreeing on the change of custody. (Alice and Bob do the same when the barrel is delivered to Alice.) At each such transfer, the signed change-of custody certificate is time stamped, the prior record is hashed, the current record is appended to the cloud storage's list.

Alice is happy because she can now pinpoint when a yogurt shipment melted, and who had custody at the time. Bob is happy because he cannot be blamed if the shipment had melted before he picked it up at the factory, and Carol is similarly protected.

Here is a point that will become important later. At every stage, Alice's supply-chain blockchain includes identities and access control. The temperature sensors sign their votes, so voter fraud is impossible. Only Alice, Bob, and Carol (and the sensors) have permission to write to the cloud storage, so it is possible to hold parties accountable if someone tries to tamper with the ledger.

In the shared-memory universal construction, a linked list served as a ledger, and an atomic memory operation

served as consensus. Here, a list kept in cloud storage serves as a ledger, and a combination of Byzantine fault-tolerant voting and human signatures serves as consensus. Although the circumstances are quite different, the “ledger plus consensus” structure is the same.

Public blockchain ledgers. Alice sells her frozen yogurt business and decides to open a restaurant. Because rents are high and venture capitalists rapacious, she decides to raise her own capital via an *intriguing coupon offering* (ICO): she sells digital certificates redeemable for discount meals when the restaurant opens. Alice hopes that her ICO will go viral, and soon people all over the world will be clamoring to buy Alice's Restaurant's coupons (many with the intention of reselling them at a markup).

Alice is media savvy, and she decides her coupons will be more attractive if she keeps them on a blockchain as *cryptocoupons*. Alice's *cryptocoupons* have three components: a *private key*, a *public key*, and a *ledger entry* (see the sidebar “Public and Private Keys”). Knowledge of the private key confers *ownership*: anyone who knows that private key can transfer ownership of (“spend”) the coupon. The public key enables *proof of ownership*: anyone can verify that a message encrypted with the private key came from the coupon's owner. The ledger conveys value: it establishes the link between the public key and the coupon with an entry saying: “*Anyone who knows the secret key matching the following public key owns one cryptocoupon.*”

Public and Private Keys

Modern cryptography is based on the notions of matching *public* and *private* keys. Any string encrypted by one can be decrypted by the other. Encrypting a message with Alice's public key yields a message only Alice can read, and encrypting a message with Alice's private key yields a *digital signature*, a message everyone can read, but only Alice could have produced.

Proof of Work Puzzles

Here is a puzzle typical of those used in PoW implementations. Let b be the block the miner wants to append to the ledger, $H(\cdot)$ a cryptographic hash function, and “.” concatenation of binary strings. The puzzle is to find a value c such that $H(b \cdot c) < D$, where D is a *difficulty* setting (the smaller D , the more difficult). Because H is difficult to invert, there is no way to find c substantially more efficient than exhaustive search.

Suppose Bob owns a coupon, and decides to transfer half of it to Carol, and keep the other half for himself. Bob and Carol each generate a pair of private and public keys. Bob creates a new ledger entry with his current public key, his new public key, and Carol's public key, saying: “*I, the owner of the private key matching the first public key, do hereby transfer ownership of the corresponding coupon to the owners of the private keys matching the next two public keys.*” Spending one of Alice's cryptocoupons is like breaking a \$20 bill into two \$10 bills: the old coupon is consumed and replaced by two distinct coupons of smaller value. (This structure is called the *unspent transaction output* (UTXO) model in the literature).

Next, Alice must decide how to manage her blockchain. Alice does not want to do it herself because she knows that potential customers might not trust her. She has a clever idea: she will crowdsource blockchain management by offering additional coupons as a fee to anyone who volunteers to be a *miner*, that is, to do the work of running a consensus protocol. She sets up a shared gossip network (sometimes called a *peer-to-peer* network) to allow coupon aficionados to share data. Customers wishing to buy or sell coupons send their transactions to this gossip network. A group of volunteer *miners* pick up these transactions, batch them into *blocks* for efficiency, and collectively execute repeated consensus protocols to append these blocks to the shared ledger, which is itself broadcast over the gossip net-

Cryptographic Hash Function

A cryptographic hash function $H(\cdot)$ has the property that for any value v , it is easy to compute $H(v)$, but it is infeasible to discover a $v' \neq v$ such that $H(v') = H(v)$.

work. Every miner, and everyone else who cares, keeps a local copy of the ledger, kept more-or-less up-to-date over the gossip network.

Alice is still worried that crooked miners could cheat her customers. Most miners are probably honest, content to collect their fees, but there is still a threat that even a small number of dishonest miners might collude with one another to cheat Alice's investors. Alice's first idea is to have miners, identified by their IP addresses, vote via the *Byzantine fault-tolerant* consensus algorithm⁴ used in the frozen yogurt example.

Alice quickly realizes this is a bad idea. Alice has a nemesis, Sybil, who is skilled in the art of manufacturing fake IP addresses. Sybil could easily overwhelm any voting scheme simply by flooding the protocol with "sock-puppet" miners who appear to be independent, but are actually under Sybil's control.

We noted earlier that the frozen yogurt supply chain blockchain was not vulnerable to this kind of "Sybil attack" because parties had reliable identities: only Alice, Bob, and Carol were allowed to participate, and even though they did not trust one another, each one knew they would be held accountable if caught cheating. By contrast, Alice's Restaurant's cryptocoupon miners do not have reliable identities, since IP ad-

dresses are easily forged, and a victim would have no recourse if Sybil were to steal his coupons.

Essentially the same problem arises when organizing a street gang: how to ensure someone who wants to join the gang is not a plainclothes police officer, newspaper reporter, or just a freeloader? One approach is what sociologists call *costly signaling*:²⁹ the candidate is required to do something expensive and difficult to fake, like robbing a store, or getting a gang symbol tattoo.

In the public blockchain world, the most common form of costly signaling is called *proof of work* (PoW). In PoW, consensus is reached by holding a self-administered *lottery* among the miners to decide which transaction is appended next to the ledger. Here is the clever part: buying a lottery ticket is a form of costly signaling because, well, it is costly: expensive in terms of time wasted and electricity bills. Sybil's talent for impersonation is useless to her if each of her sock puppet miners must buy an expensive, long shot lottery ticket.

Specifically, in the PoW lottery, miners compete to solve a puzzle, where solving the puzzle is difficult, but proving one has solved the puzzle is easy (see sidebar "Proof of Work Puzzles"). Simplifying things for a moment, the first miner to solve the puzzle wins the consensus, and gets to choose the next block to append to the ledger. If that block is valid, that miner also receives a reward (another coupon), but the other miners receive nothing, and must start over on a new puzzle.

As hinted, the previous paragraph was an oversimplification. In fact, PoW consensus is not really consensus. If two miners both solve the puzzle at about the same time, they could append blocks to the blockchain in parallel, so that neither block precedes the other in the chain. When this happens, the blockchain is said to *fork*. Which block should subsequent miners build on? The usual answer is to build on the block whose chain is

longest, although other approaches have been suggested.²⁵

As a result, there is always some uncertainty whether a transaction on the blockchain is permanent, although the probability that a block, once on the blockchain, will be replaced decreases exponentially with the number of blocks that follow it.^{9,20} If Bob uses Alice's cryptocoupons to buy a car from Carol, Carol would be prudent to wait until Bob's transaction is fairly deep in the blockchain to minimize the chances that it will be displaced by a fork.

Although PoW is currently the basis for the most popular cryptocurrencies, it is not the only game in town. There are multiple proposals where cryptocurrency ownership assumes the role of costly signaling, such as Ethereum's Casper² or Algorand.¹⁰ Cachin and Vukolic³ give a comprehensive survey of blockchain consensus protocols.

Discussion. The distinction between *private* (or *permissioned*) blockchain systems, where parties have reliable identities, and only vetted parties can participate, and *public* (or *permissionless*) blockchain systems, where parties cannot be reliably identified, and anyone can participate, is critical for making sense of the blockchain landscape.

Private blockchains are better suited for business applications, particularly in regulated industries, like finance, subject to know-your-customer and anti-money-laundering regulations. Private blockchains also tend to be better at governance. For example, the lack of any orderly procedure for updating the ledger protocol in response to changing circumstances has caused feuding factions to split both Ethereum⁶ and Bitcoin¹² into distinct, incompatible currencies. Most prior work on distributed algorithms has focused on systems where participants have reliable identities.

Public blockchains are appealing for applications such as Bitcoin, which seek to ensure nobody can control who can participate, and participants may not be eager to have their identities known. Although PoW was invented by Dwork and Naor⁷ as a way to control spam, Nakamoto's application of PoW to large-scale consensus was a genuine innovation, one that launched the entire blockchain field.

Figure 1. Pseudocode for DAO-like contract.

```
function withdraw(unit amount){
  client = msg.sender;
  if (balance[ client ] >= amount){
    if (client . call . sendMoney(amount)){
      balance[ client ] -= amount;
    }}
}
```

Figure 2. Pseudocode for DAO-like exploit.

```
function sendMoney(unit amount){
  victim = msg.sender;
  balance += amount;
  victim.withdraw(amount)
}
```

Smart Contracts

Most blockchain systems also provide some form of scripting language to make it easier to add functionality to ledgers. Bitcoin provides a rudimentary stack-based language, while Ethereum⁸ provides a Turing-complete imperative language similar to JavaScript.

Such programs are often called *smart contracts* (or *contracts*) (though they are arguably neither smart nor contracts). Here we focus on Ethereum-style contracts.

Here are some examples of simple contract functionality. A *hashlock* h prevents an asset from being transferred until the contract receives a matching secret s , where $h = H(s)$, for H a cryptographic hash function (see the sidebar “Cryptographic Hash Function”). Similarly, a *timelock* t prevents an asset from being transferred until a specified future time t .

Suppose Alice wants to trade some of her coupons to Bob in return for some bitcoins. Alice’s coupons live on one blockchain, and Bob’s bitcoins live on another, so they must devise an *atomic cross-chain swap* protocol to consummate their deal. Naturally, neither one trusts the other.

Here is a simple protocol. Let us generously assume 24 hours is enough time for anyone to publish a smart contract on either blockchain, and for the other party to detect that the contract has been published.

1. Alice creates a secret s , $h = H(s)$, and publishes a contract on the coupon blockchain with hashlock h and timelock 48 hours in the future, ensuring the contract will transfer the coupons to Bob if Bob can produce s within 48 hours. If he cannot, the coupons will be refunded to Alice.

2. When Bob confirms that Alice’s contract has been published on the coupon blockchain, he publishes a contract on the Bitcoin blockchain with the same hashlock h but with timelock 24 hours in the future, ensuring the contract will transfer the bitcoins to Alice if Alice can produce s within 24 hours. If she cannot, the bitcoins will be refunded to Bob.

3. When Alice confirms that Bob’s contract has been published on the Bitcoin blockchain, she sends the secret s to Bob’s contract, taking possession of the bitcoins, and revealing s to Bob.



Public blockchains are appealing for applications such as Bitcoin, which seek to ensure nobody can control who can participate, and participants may not be eager to have their identities known.



4. Bob sends s to Alice’s contract, acquiring the coupons and completing the swap.

If Alice or Bob crashes during steps one or two, then the contracts time out and refund their assets to the original owners. If either crashes during steps three and four, then only the party who crashes ends up worse off. If either party tries to cheat, for example, by publishing an incorrect contract, then the other party can simply stop participating and its asset will be refunded. Alice’s contract needs a 48-hour timelock to give Bob enough time to react when she releases her secret before her 24 hours are up.

This example illustrates the power of smart contracts. There are many other uses for smart contracts, including finance,²³ digital rights management,²⁶ supply chain,¹⁹ insurance,¹⁶ and even off-chain transactions,²¹ a way of streamlining commerce by conducting most business off-chain, and falling back to the blockchain only as necessary to settle balances.

Smart contracts as objects. A smart contract resembles an object in an object-oriented programming language. A contract encapsulates *long-lived state*, a *constructor* to initialize that state, and one or more functions (methods) to manage that state. Contracts can call one another’s functions.

In Ethereum, all contracts are recorded on the blockchain, and the ledger includes those contracts’ current states. When a miner constructs a block, it fills that block with calls to smart contract functions, and executes them one-by-one, where each contract’s final state is the next contract’s initial state. These contract executions occur in order, so it would appear there is no need to worry about concurrency.

Smart contracts as monitors. The Decentralized Autonomous Organization (DAO) was an investment fund set up in 2016 to be managed entirely by smart contracts, with no direct human administration. Investors could vote on how the fund’s funds would be invested. At the time, there were breathless journalistic accounts explaining how the DAO would change forever the shape of investing.^{22,27}

Figure 1 shows a fragment of a DAO-like contract, illustrating a func-

Figure 3. ERC20 Token example.

```

contract ERC20Example {
    // Balances for each account
    mapping(address => uint256) balances;
    // Owner of account approves the transfer of an amount to another account
    mapping(address => mapping (address => uint256)) allowed;
    // other fields omitted
    ...
    // Allow spender to withdraw from your account, multiple times, up to the amount.
    function approve(address spender, uint amount) public returns (bool success) {
        allowed[msg.sender][spender] = amount; // alter approval
        Approval(msg.sender, spender, amount); // blockchain event
        return true;
    }
    function allowance(address tokenOwner, address spender) public returns (uint
    remaining){
        return allowed[tokenOwner][spender];
    }
    function transferFrom(address from, address to, uint tokens) public (bool success){
        balances[from] = balances[from].sub(tokens);
        allowed[from][msg.sender] = allowed[from][msg.sender].sub(tokens);
        balances[to] = balances[to].add(tokens);
        Transfer (from, to, tokens);
        return true;
    }
    ... // other functions omitted
}

```

tion that allows an investor to withdraw funds. First, the function extracts the client's address (Line 2), then checks whether the client has enough funds to cover the withdrawal (Line 3). If so, the funds are sent to the client through an external function call (Line 4), and if the transfer is successful, the client's balance is decremented (Line 5).

This code is fatally flawed. In June 2016, someone exploited this function to steal about \$50 million in funds from the DAO. As noted, the expression in Line 3 is a call to a function in the client's contract. Figure 2 shows the client's code. The client's contract immediately calls `withdraw()` again (Line 4). This *re-entrant* call again tests whether the client has enough funds to cover the withdrawal (Line 3), and because `withdraw()` decrements the balance only after the nested call is complete, the test erroneously passes,

and the funds are transferred a second time, then a third, and so on, stopping only when the call stack overflows.

This kind of *re-entrancy attack* may at first glance seem like an exotic hazard introduced by a radically new style of programming, but if we change our perspective slightly, we can recognize a pitfall familiar to any undergraduate who has taken a concurrent programming course.

First, some background. A *monitor* is a concurrent programming language construct invented by Hoare¹⁵ and Brinch Hansen.¹¹ A monitor is an object with a built-in mutex lock, which is acquired automatically when a method is called and released when the method returns. (Such methods are called *synchronized methods* in Java.) Monitors also provide a `wait()` call that allows a thread to release the monitor lock, suspend, eventually awaken, and reacquire the lock. For example, a thread attempting to consume an item from an empty buffer could call `wait()` to suspend until there was an item to consume.

The principal tool for reasoning about the correctness of a monitor implementation is the monitor invariant, an assertion that holds whenever no thread is executing in the monitor. The invariant can be violated while a thread is holding the monitor lock, but it must be restored when the thread releases the lock, either by returning from

a method, or by suspending via `wait()`.

If we view smart contracts through the lens of monitors and monitor invariants, then the re-entrancy vulnerability looks very familiar. An external call is like a `wait()` suspension, because even though there is no explicit lock, the call makes it possible for a second program counter to execute that contract's code concurrently with the first program counter. The DAO-like contract shown here implicitly assumed the invariant that each client's entry in the balance table reflects its actual balance. The error occurred when the invariant, which was temporarily violated between Lines 3 and 5, was not restored before giving up the (virtual) monitor lock by making an external call.

Here is why the distributed computing perspective is valuable. When explained in terms of monitors and monitor invariants, the re-entrancy vulnerability is a familiar, classic concurrency bug, but when expressed in terms of smart contracts, it took respected, expert programmers by surprise, resulting in substantial disruption and embarrassment for the DAO investors, and required rolling back troublesome but technically legal transactions and proceeding as if they had never taken place.⁶

Smart contracts as read-modify-write operations. The *ERC20* token standard²⁸ is the basis for many recent *initial coin offerings* (ICOs), a popular way to raise capital for an undertaking without actually selling ownership. The issuer of an *ERC20* token controls token creation. Tokens can be traded or sold, much like Alice's Restaurant's coupons discussed earlier. *ERC20* is a standard, like a Java interface, not a particular implementation.

As illustrated in Figure 3, an *ERC20* token contract keeps track of how many tokens each account owns (the `balances` mapping at Line 3), and also how many tokens each account will allow to be transferred to each other's account (the `allowed` mapping at Line 5). The `approve()` function (Lines 9–13) adjusts the limit on how many tokens can be transferred at one time to another account. It updates the `allowed` table (Line 10), and generates a blockchain event to make these changes easier to track (Line 11). The `allowance()` function queries this `allowance` (Lines 14–16).

Figure 4. An incorrect atomic decrement operation.

```

class Counter {
    private int counter;
    public void dec() {
        int temp = counter;
        temp = temp - 1;
        counter = temp;
    }
    ...
}

```

The `transferFrom()` function (Lines 17–23) transfers tokens from one account to another, and decreases the allowance by a corresponding amount. This function assumes the recipient has sufficient allowance for the transfer to occur.

Here is how this specification can lead to undesired behavior. Alice calls `approve()` to authorize Bob to transfer as many as 1,000 tokens from her account to his. Alice has a change of heart, and issues a transaction to reduce Bob's allowance to a mere 100 tokens. Bob learns of this change, and before Alice's transaction makes it onto the blockchain, Bob issues a `transferFrom()` call for 1,000 tokens to a friendly miner, who ensures Bob's transaction precedes Alice's in the next block. In this way, Bob successfully withdraws his old allowance of 1,000 tokens, setting his authorization to zero, and then, just to spite Alice, he withdraws his new allowance of 100 tokens. In the end, Alice's attempt to reduce Bob's allowance from 1,000 to 10 made it possible for Bob to withdraw 1,100 tokens, which was not her intent.

In practice, ERC20 token implementations often employ ad-hoc workarounds to avoid this vulnerability, the most common being to redefine the meaning of `allow()` so that it will reset an allowance from a positive value to zero, and in a later call, from zero to the new positive value, but will fail if asked to reset an allowance from one positive value to another.

The problem is that `approve()` blindly overwrites the old allowance with the new allowance, regardless of whether the old allowance has changed. This practice is analogous to trying to implement an atomic decrement as shown in Figure 4. Here, the decrement method reads the shared counter state into a local variable (Line 4), increments the local variable (Line 5), and stores the result back in the shared state (Line 6). It is not difficult to see that this method is incorrect if it can be called by concurrent threads, because the shared counter state can change between when it was read at Line 4 and when it was written at Line 6. When explained in terms of elementary concurrent programming, the ERC20 concurrency flaw is obvi-

ous, but when expressed in terms of smart contracts that ostensibly do not need a concurrency model, the same design flaw was immortalized in a token standard with a valuation estimated in billions of dollars.

Discussion. We have seen the notion that smart contracts do not need a concurrency model because execution is single-threaded is a dangerous illusion. Sergey and Hobor²⁴ give an excellent survey of pitfalls and common bugs in smart contracts, explaining how they are disguised versions of familiar concurrency pitfalls and bugs. Atzei et al. provide a comprehensive survey of vulnerabilities in Ethereum's smart contract design. Some of today's languages' pitfalls and traps can be avoided by carefully following codes of best practices.^{5,17}

Conclusion

Radical innovation often emerges more readily from outside an established research community than from inside. Would Nakamoto's original Bitcoin paper have been accepted to one of the principal distributed conferences back in 2008? We will never know, of course, but the paper's lack of a formal model, absence of rigorous proofs, and lack of performance numbers would have been a severe handicap.

Today, blockchain research is one of the more vibrant areas of computer science, with the potential of revolutionizing how our society deals with trust. The observation that many blockchain constructs have unacknowledged doppelgängers (or at least, precursors) is not a criticism of either research community, but rather an appeal to each side to pay more attention to the other. C

References

1. Attiya, H. and Welch, J. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.
2. Buterin, V. and Griffith, V. Casper the Friendly Finality Gadget. (2017); https://github.com/ethereum/research/commits/master/papers/casper-basics/casper_basics.pdf.
3. Cachin, C. and Vukolic, M. Blockchain consensus protocols in the wild (Keynote Talk). In *Proceedings of the 31st International Symposium on Distributed Computing*. Andréa W. Richa, ed. (2017), 1:1–1:16.
4. Castro, M. and Liskov, B. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.* 20, 4, (2002) 398–461.
5. Consensys, Inc. Ethereum Smart Contract Security Best Practices; <https://consensys.github.io/smart-contract-best-practices/>
6. del Castillo, M. Ethereum Executes Blockchain Hard Fork to Return DAO Funds, (2016); <https://www.coindesk.com/ethereum-executes-blockchain-hard-fork-return-dao-investor-funds/>.

7. Dwork, C. and Naor, M. *Pricing via Processing or Combatting Junk Mail*. Springer Berlin Heidelberg, 1993, 139–147.
8. Ethereum; <https://github.com/ethereum/>.
9. Garay, J., Kiayias, A., and Leonardos, N. *The Bitcoin Backbone Protocol: Analysis and Applications*. Springer Berlin, Heidelberg, 2015, 281–310.
10. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, (2017) 51–68.
11. Hansen, P.B. *Operating System Principles*. Prentice-Hall, Inc., 1973.
12. Hearn, M. The resolution of the Bitcoin experiment, (2016); <https://blog.plan9.net/the-resolution-of-the-bitcoin-experiment-dabb30201f7>.
13. Herlihy, M. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.* 13, 1 (1991) 124–149.
14. Herlihy, M. and Shavit, N. *The Art of Multiprocessor Programming*. Morgan Kaufmann Publishers, Inc., 2008.
15. Hoare, C.A.R. Monitors: An operating system structuring concept. *Commun. ACM* 17, 10 (Oct. 1974), 549–557.
16. Marr, B. Blockchain implications every insurance company needs to consider now. *Forbes*, (2017); <https://www.forbes.com/sites/bernardmarr/2017/10/31/blockchain-implications-every-insurance-company-needs-to-consider-now/#982922468825>.
17. Maurelian. Beyond Smart Contract Best Practices for UX and Interoperability; <https://medium.com/@maurelian/beyond-smart-contract-best-practices-for-ux-and-interoperability-6d94d27c1e0f>.
18. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System, (2009); <http://www.bitcoin.org/bitcoin.pdf>.
19. O'Byrne, R. How Blockchain Can Transform the Supply Chain, (2017); <https://www.logisticsbureau.com/how-blockchain-can-transform-the-supply-chain/>.
20. Pass, R., Seeman, L., and Shelat, A. Analysis of the Blockchain Protocol in Asynchronous Networks. *Cryptology ePrint Archive Report 2016/454*; <https://eprint.iacr.org/2016/454>.
21. Poon, J. and Dryja, T. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, (2016); <https://lightning.network/lightning-network-paper.pdf>.
22. Popper, N. A venture fund with plenty of virtual capital, but no capitalist. *New York Times* (May 22, 2016); <https://www.nytimes.com/2016/05/22/business/dealbook/crypto-ether-bitcoin-currency.html>.
23. Prisco, G. Smart Contracts and the Future of Banking, (2017); <https://www.nasdaq.com/article/smart-contracts-and-the-future-of-banking-cm849118>.
24. Sergey, I. and Hobor, A. A Concurrent Perspective on Smart Contracts. *CoRR abs/1702.05511* (2017). [arXiv:1702.05511](https://arxiv.org/abs/1702.05511); <http://arxiv.org/abs/1702.05511>
25. Sompolskiy, Y., Lewenberg, Y. and Zohar, A. SPECTRE: A Fast and Scalable Cryptocurrency Protocol. *Cryptology ePrint Archive, Report 2016/1159*; <http://eprint.iacr.org/2016/1159.pdf>
26. Tapscott, D. and Tapscott, A. Blockchain could help artists profit more from their creative works. *HBR*, (2017); <https://hbr.org/2017/03/blockchain-could-help-artists-profit-more-from-their-creative-works>.
27. Vigna, P. Chiefless Company rakes in more than \$100 million. *WSJ*, (2016); <https://www.wsj.com/articles/chiefless-company-rakes-in-more-than-100-million-1463399393>.
28. Vogelsteller, F. and Buterin, V. ERC-20 Token Standard; <https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md>.
29. Wikipedia. Signalling Theory; https://en.wikipedia.org/wiki/Signalling_theory.

Maurice Herlihy (maurice.herlihy@gmail.com) is the An Wang Professor of Computer Science at Brown University, Providence, RI, USA.

© 2019 ACM 0001-0782/19/2 \$15.00



Watch the authors discuss this work in the exclusive *Communications* video. <https://cacm.acm.org/videos/blockchains-from-a-distributed-computing-perspective>

Separation logic is a key development in formal reasoning about programs, opening up new lines of attack on longstanding problems.

BY PETER O'HEARN

Separation Logic

A FUNDAMENTAL TECHNIQUE in reasoning about programs is the use of logical assertions to describe properties of program states. Turing used assertions to argue about the correctness of a particular program in 1949,⁴⁰ and they were incorporated into general formal systems for program proving starting with the work of Floyd²¹ and Hoare²² in the 1960s. Hoare logic, which separation logic builds upon, is a formal system for proving specifications of the form

$$\{precondition\}code\{postcondition\}$$

where the precondition and postcondition are vassertions describing properties of the input and output states. For example,

$$\{x == N\}code\{x == N \wedge y == N!\}$$

can serve as a specification of an imperative program that computes the factorial of the value held in variable x and places it in y .

Hoare logic and related systems worked very well for programs manipulating simple primitive data types such as for integers or strings, but proofs became more complex when dealing with structured data containing

embedded pointers. One of the founding papers of separation logic summarized the problem as follows.³²

"The main difficulty is not one of finding an in-principle adequate axiomatization of pointer operations; rather there is a mismatch between simple intuitions about the way that pointer operations work and the complexity of their axiomatic treatments...when there is aliasing, arising from several pointers to a given cell, an alteration to a cell may affect the values of many syntactically unrelated expressions."

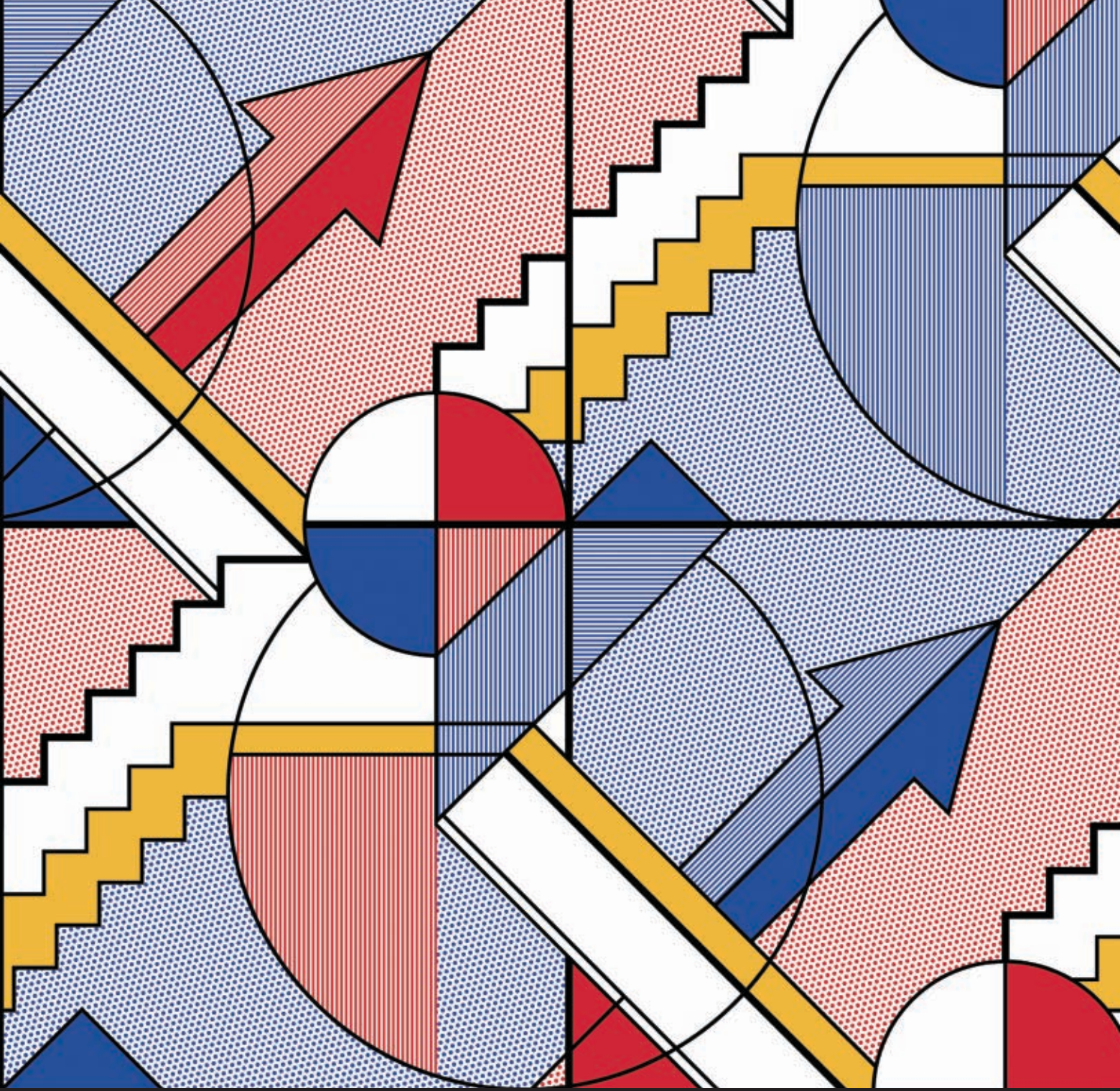
Bornat provided a good description of the struggles in reasoning about mutable data structures up to 2000.⁶

In joint work with John Reynolds and others we developed separation logic (SL) to address the fundamental problem of reasoning about programs that mutate data structures. From a special logic for heaps, it gradually evolved into a general theory for modular reasoning about concurrent as well as sequential programs. Efforts by many researchers established that the logic provides a basis for efficient proof search in automatic and semi-automatic proof tools, for example, giving rise to the Infer static analyzer, a tool that is in deployment at Facebook where it catches thousands of bugs per month before code reaches production in products used daily by over one billion people.

Separation logic is an extension of Hoare logic, which employs novel logical operators, most importantly the separating conjunction $*$ (pronounced "and

» key insights

- Separation logic supports in-place updating of facts as we reason, in a way that mirrors in-place update of memory during execution, and this leads to logical proofs about imperative programs that match computational intuition.
- Separation logic supports scalable reasoning by using an inference rule (the frame rule) that allows a proof to be localized to the resources that a program component accesses (its footprint).
- Concurrent separation logic shows that modular reasoning about threads that share storage and other resources is possible.



separately”) when writing assertions. For example, we might write:

$$\begin{aligned} &\{x \mapsto 0 * y \mapsto 0\} \\ &\quad [x] = y; \\ &\quad [y] = x \\ &\{x \mapsto y * y \mapsto x\} \end{aligned}$$

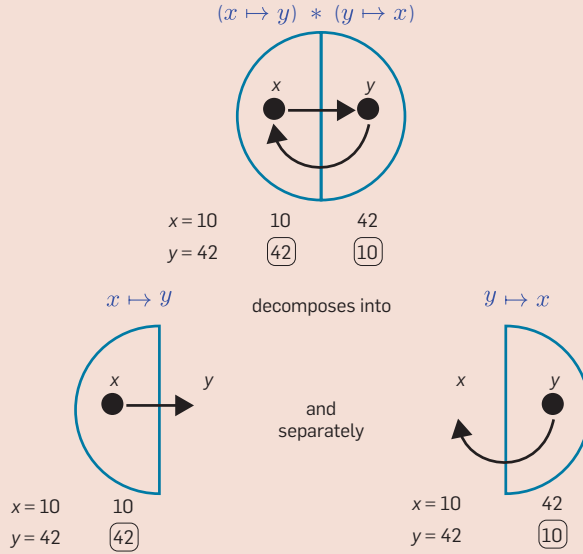
as a specification of code that wires together two memory locations into a cyclic linked list. Here $x \mapsto v$ says that pointer variable x holds the address of a memory location where v is stored (or more briefly, x points to v), and a command of the form $[x] = v$ updates the location referred to by x so that its contents becomes v .

The use of $*$ rather than the usual Boolean conjunction $\wedge$ ensures x and y are not aliases—distinct names for the same location—so that we have a two-element cyclic list in the postcondition. A central principle is that a command that mutates a single location affects only one $*$ -conjunct: operational in-place update is mirrored in the logic, addressing the key difficulty where “an alteration to a cell may affect the values of many syntactically unrelated expressions.”

Reynolds was the first to describe a program logic including the separating

conjunction; he defined an intuitionistic (constructive) logic with $*$,³⁷ building on earlier ideas of Burstall.¹⁰ O’Hearn, and Ishtiaq²⁶ realized the assertion language could be seen as an instance of the resource logic BI of O’Hearn and Pym;³¹ they independently discovered the same intuitionistic logic as Reynolds, and also saw that a more powerful Boolean (nonconstructive) variant was possible in which one could reason about explicit memory management (Reynolds had assumed garbage collection). They also introduced the separating implication $\multimap$.

Figure 1. Picture semantics.



proposed a concurrent separation logic (CSL). CSL showed efficient reasoning about threads that share access to storage, proofs that mirrored design principles espoused by Dijkstra at the birth of concurrent programming.¹⁶ The correctness of CSL's proof rules (its 'soundness') turned out to be a formidable problem, solved eventually by Brookes. Brookes and O'Hearn were awarded the 2016 Gödel prize for their papers on CSL,^{8,30} the significance of which was summed up as follows:

"For the last 30 years experts have regarded pointer manipulation as an unsolved challenge for program verification and shared-memory concurrency as an even greater challenge. Now, thanks to CSL, both of these problems have been elegantly and efficiently solved; and they have the same solution."

—2016 Gödel Prize citation^a

It is worth remarking that the first part of this citation, about pointer manipulation, applies to sequential and not just concurrent SL.

After the early papers, research on SL expanded rapidly. Starting from a special logic for heaps SL has evolved into a general theory for modular reasoning. Non-standard models of SL based on an abstract model theory due to Pym provided many potential avenues for wider application, and Gardner and others realized that there exist non-standard models that support modular reasoning about intertwined structures *as if* they were separate. SL has even been applied to interfering processes using fine-grained concurrency, a situation far removed from the original intuitions of the logic.

SL is the basis of numerous automated proof tools, and it has been used in significant verification efforts. It has been used to provide the first verification of a crash-proof file system,¹⁴ and to provide the first verification of a commercial, preemptive OS microkernel.⁴¹ These verification efforts are semi-automatic, done by a human together with a proof assistant (in these cases, the Coq proof assistant). SL has also been used in static program analysis, where weaker properties than full correctness are targeted but with higher automation, so that the tool can scale better both in the sizes of codebases

Figure 2. Mathematical semantics.

- Assume a partial commutative monoid $(H, \circ, e)$, where $\circ : H \times H \rightarrow H$ and $e \in H$. Pre/post assertions denote elements of the powerset $\mathcal{P}(H)$.
- $*$ lifts $\circ$ to the powerset $\mathcal{P}(H)$: $P * Q$ is $\{h_P \circ h_Q \mid h_P \circ h_Q \text{ defined and } h_P \in P \text{ and } h_Q \in Q\}$
- emp denotes the singleton set of the empty heaplet: $\{e\}$.
- $\multimap$ is an implication quantifying over separate heaps: $P \multimap Q$ is $\{h \mid \forall h_P. h \circ h_P \text{ defined and } h_P \in P \text{ implies } h \circ h_P \in Q\}$
- In the RAM model H is the set of finite partial functions from positive integers (addressable locations) to integers, $h \circ h'$ is the union of functions with disjoint domain, and undefined when h and h' overlap. e is the empty partial function. The assertion $n \mapsto m$ denotes the singleton set $\{f\}$ where f maps n to m and is undefined elsewhere.
- To deal with variables and also quantifiers consider functions s from variables to integers, and extend the above semantics pointwise to pairs (s, h) .

SL for sequential programs reached maturity in a further paper of O'Hearn, Reynolds and Yang.³² In that work O'Hearn proposed the following principle of local reasoning, both as a way to describe what was special about SL and as a guiding principle for development of reasoning methods.

"To understand how a program works, it should be possible for reasoning and specification to be confined to the cells that the program actually accesses. The value of any other cell will automatically remain unchanged."

A proof rule—the *frame rule*—allowed to infer that cells remain unchanged when they are not mentioned in a precondition. The frame rule was named in homage to the frame problem from artificial intelligence, which concerns axiomatizing state changes without enumerating all of the things that do not change. The frame rule is the key to scalable reasoning in SL.

Reynolds' influential survey article summarized the early developments up to 2002.³⁸ At the end of this early period, O'Hearn circulated a note that

a <https://bit.ly/2ywwlpp>

covered and the number of programmers served. Static analysis with SL has matured to the point where it has been applied industrially in the Facebook Infer program analyzer, an open source tool used at Facebook, Mozilla, Spotify, Amazon Web Services, and other companies (www.fbinfer.com).

The purpose of this article is to describe the basic ideas of SL as well as these and other developments.

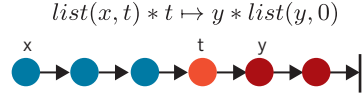
Separating Conjunction and Implication

Mathematical semantics has been critical to the discovery and further SL development, but many of the main points can be gleaned from “picture semantics.” Consider the first picture in Figure 1. We read the formula at the top of this figure as “ x points to y and separately y points to x .” Going down the middle of the diagram is a line that represents a heap partitioning: a separating conjunction asks for a partitioning that divides the heap into parts, *heaplets*, satisfying its two conjuncts. At the bottom of the first picture is an example of a concrete memory description that corresponds to the diagram. There, x and y have values 10 and 42 (in the “environment,” or “register bank”), and 10 and 42 are themselves locations with the indicated contents (in the “heaplet,” or even “RAM”).

The indicated separating conjunction here is true of the pictured memory because the parts satisfy the conjuncts, as indicated in the second picture. The meaning of “ x points to y and yet to nothing” is precisely disambiguated in the RAM description below the diagram: x and y denote values (10 and 42), x ’s value is an allocated memory address which contains y ’s value, but y ’s value is not allocated. The separating conjunction splits the heap/RAM, but it does not split the association of variables to values.

Generally speaking, the separating conjunction $P * Q$ is true of a heap if it can be split into two heaplets, one of which makes P true and the other of which makes Q true. A distinction between $*$ and Boolean conjunction $\wedge$ is that $P * P \neq P$ where $P \wedge P = P$. In particular, $x \mapsto v * x \mapsto v$ is always false: there is no way to divide any heap in such a way that a cell x goes to both partitions.

$*$ is often used with linked structures. If $list(x, y)$ describes an acyclic linked list running from x to y , then we can describe a structure with a list segment, followed by a single pointer, followed by a further list running up to 0 (null), as follows:



This is the kind of structure you might need to consider when deleting an element from a list, or inserting one into it.

There is a further connective, the separating implication or “magic wand.” $P \multimap Q$ says that whenever the current heaplet is extended with a separate heaplet satisfying P , the resulting combined heaplet will satisfy Q . For example, $(x \mapsto -) * ((x \mapsto 3) \multimap Q)$ says that x is allocated in the current heap, and that if you mutate its contents to 3 then Q will hold. This describes the “weakest precondition” for the mutation $[x] = 3$ with postcondition Q .²⁶

Finally, there is an assertion emp which says “the heaplet is empty,” emp is the unit of $*$, so that $P = \text{emp} * P = P * \text{emp}$. Also, $\multimap$ and $*$ fit together in a way similarly to how implication $\Rightarrow$ and conjunction $\wedge$ do in standard logic. For example, the entailment

$$A * (A \multimap B) \vdash B$$

(where $\vdash$ reads “entails”) is a SL relative of “modus ponens.”

Although we will concentrate on the informal picture semantics in this article, for the theoretically inclined we have included a glimpse of the formal semantics in Figure 2.

Rules for Program Proof

Figure 3 contains a selection of proof rules of SL. The rules are divided into axioms for basic mutation commands (the “small axioms”) and inference rules for modular reasoning. An inference rule says “if you can derive what is above the line, then so can you what is below,” and the axioms are derivable true statements that are given. The small axioms are for a programming language with load and store instructions similar to an assembly language. If we vary the programming language the small axioms change. The concurrency rule uses a composition operator $||$ for running two processes in parallel, derived from Dijkstra’s *parbegin/parend*.¹⁶

The first small axiom just says that if x points to something beforehand, then it points to v afterward, and it says this for a small portion of the state in which x is the only active cell.

Figure 3. Separation logic proof system (a selection).

SMALL AXIOMS

Pointer Write (Store)

$$\{x \mapsto -\}[x] = v \{x \mapsto v\}$$

Pointer Read (Load)

$$\{x \mapsto v\}y = [x] \{y == v \wedge x \mapsto v\}$$

Allocation

$$\{\text{emp}\}x = \text{alloc}() \{x \mapsto -\}$$

De-Allocation

$$\{x \mapsto -\}\text{free}(x) \{\text{emp}\}$$

LOCAL REASONING RULES

Frame Rule

$$\frac{\{pre\}code \{post\}}{\{pre * frame\}code \{post * frame\}}$$

Concurrency Rule

$$\frac{\{pre_1\}process_1 \{post_1\} \quad \{pre_2\}process_2 \{post_2\}}{\{pre_1 * pre_2\}process_1 || process_2 \{post_1 * post_2\}}$$

The second axiom says that if x points to v and we read x into y , then y will have value v . Here, we distinguish between the value in a variable or register (x and y) and the r -value in a heap cell whose l -value is the value held in x . The second axiom assumes that x does not appear in syntactic expression v (see O'Hearn et al.³² for a precise description of this and other variable side conditions).

The allocation axiom says: If you start with no heap, then you end with a heap of size 1. Conversely the De-Allocation axiom starts with a hap of size 1 and ends with the empty heap. The Application axiom assumes that allocation always succeeds. To model a case where allocation might fail we could use a disjunctive postcondition, like $x \mapsto \vee x = 0$; this is what tools such as SpaceInvader and Infer, discussed later, do for `malloc()` in C .

The small axioms are so named because each mentions a small amount of memory: a single memory cell. When people first see the axioms they can

come as a shock: aren't they too simple? Previous approaches had complex descriptions accounting for the effect of mutations on global properties of graph-like structures.⁶

In actuality, there is a sense in which the small axioms capture all that is needed to know about the statements they describe. In intuitive terms, we can say that imperative computation proceeds by in-place update, where these primitive statements update or access a single memory cell at a time; describing what happens to only that cell should be enough. The small axioms are thus an extreme illustration of the principle of local reasoning.

The frame rule in Figure 3 provides logical support for this intuition. It allows us to extend reasoning from one to multiple cells; so the seeming restriction to one cell in the small axioms is not a restriction at all, but rather a pleasantly succinct description. For instance, if we choose $x \mapsto y$ as our frame then the first instance in Figure 4 gives the reasoning

for the second step of the code to wire up a cyclic linked list described at the start of the paper.

The ultimate theoretical support for the small axioms came from a completeness theorem in Yang's Ph.D. thesis.⁴² He showed the small axioms and frame rule and several other inference rules (particularly Hoare's rules for strengthening preconditions and weakening postconditions, and a rule for existential quantifiers) can be used to derive all true Hoare triples for these statements.

Locality properties of program behavior, and their connection to logic,^{13,44} are critical for these results:

"An assertion talks about a heaplet rather than the global heap, and a spec $\{P\}C\{Q\}$ says that if C is given a heaplet satisfying P then it will never try to access heap outside of P (other than cells allocated during execution) and it will deliver a heaplet satisfying Q if it terminates."²

In-place reasoning as with the two-element cyclic list has been applied to many imperative programs. As an example, consider the insertion of a node y into a linked list after position x . We can do this in two steps: first we swing x 's pointer so it points to y , and then we swing y to point to z (the node after x).

$$\begin{aligned} & \{x \mapsto z * \text{list}(z) * y \mapsto -\} \\ & [x] = y \\ & \{x \mapsto y * \text{list}(z) * y \mapsto -\} \\ & [y] = z \\ & \{x \mapsto y * \text{list}(z) * y \mapsto z\} \end{aligned}$$

Here, in the precondition for each step we write the frame in red; it is the blue that is updated in place. The reader can see how, using the small axiom for free together with the frame rule, we could reason about the converse case of removing an element from a list.

This example generalizes to many other list and tree algorithms: insertion, deletion, reversal, and so on. The SL proofs resemble the box-and-pointer arguments that have long been used informally in describing data structure mutation.

These ideas extend to concurrent programs; for example, the second rule instance in Figure 4 uses the concurrency rule to reason about our two-element cyclic list, but wired up concurrently rather than sequentially. The $*$ in the precondition in this instance ensures that x and y are not aliases, so there is no data race in the parallel program.

Figure 4. Frame and concurrency examples.

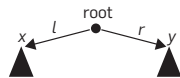
$$\frac{\frac{\{y \mapsto 0\} [y] = x \{y \mapsto x\}}{\{(y \mapsto 0) * (x \mapsto y)\} [y] = x \{(y \mapsto x) * (x \mapsto y)\}}}{\frac{\{x \mapsto 0\} [x] = y \{x \mapsto y\} \quad \{y \mapsto 0\} [y] = x \{y \mapsto x\}}{\{(y \mapsto 0) * (x \mapsto 0)\} [x] = y \parallel [y] = x \{(y \mapsto x) * (x \mapsto y)\}}}$$

Figure 5. deletetree example.

```
(1) void deletetree(struct node *root)
    { if( root != 0 )
      { struct node *left = root->l, *right = root->r;
        deletetree(left);
        deletetree(right);
        free( root );
      } }
```

(2) Spec: $\{tree(root)\} \text{deletetree}(root) \{\text{emp}\}$

(3) $tree(root) =$ if $root == 0$ then emp
else $\exists xy. root \mapsto [l: x, r: y] * tree(x) * tree(y).$



(4) $\{root \mapsto [l: left, r: right] * tree(left) * tree(right)\}$
deletetree(left);
 $\{root \mapsto [l: left, r: right] * \text{emp} * tree(right)\}$
deletetree(right);
 $\{root \mapsto [l: left, r: right] * \text{emp} * \text{emp}\}$
free(root);
 $\{\text{emp} * \text{emp} * \text{emp}\}$
 $\{\text{emp}\}$

The concurrency rule is the main rule of CSL. In applying CSL to languages with dynamic thread creation instead of parbegin/-parend different rules are needed, but the basic point that separation allows independent reasoning about processes carries over.

SL's concurrency rule took inspiration from the "disjoint concurrency rule" of Hoare.²³ Hoare's rule used $\wedge$ in place of $*$ together with side conditions to rule out interference.^b $*$ allows us to extend its applicability to pointer structures. But even without pointers, the CSL rule is more powerful. Indeed, upon seeing CSL

Hoare immediately exclaimed to the author: "We can prove parallel quicksort!" A direct proof can be given using $*$ to recognize and unite disjoint array partitions.³⁰

Frames, Footprints, and Local Reasoning

The previous section describes how the separating conjunction leads to simple proofs of the individual steps of heap mutations, and how the frame rule embeds reasoning about small chunks of memory within larger memories. Here, the rules' more fundamental role as a basis for scalable reasoning is explained.

I illustrate by reasoning about a recursive program for deleting the nodes in a binary tree. Consider the C program in (1) of Figure 5. This program satisfies the specification in (2) of the figure, where the *tree* predicate says that its argument points to a binary tree in memory. The predicate is defined recursively in (3), with a diagram below depicting what is described by the *else* part of the definition. Note that here we are using a "points-to" predicate $root \mapsto [l : x, r : y]$ for describing records with *l* and *r* fields.

The use of *emp* in the *if* branch of the definition means that *tree(r)* is true of a heaplet that contains all and only the cells in the tree; there are no additional cells. Thus, the specification of *deletetree(r)* does not mention nodes not in the tree. This is analogous to what we did with the small axioms for basic statements in Figure 3,

and is a typical pattern in SL reasoning: "small specifications" are used which mention only the cells touched by the program component (its footprint).

The critical part of the proof of the program is presented in (4), where the precondition at the beginning is obtained by unwinding the recursive definition using the *if* condition $root \neq 0$. The proof steps then follow the intuitive description of the algorithm: the first recursive call deletes the left subtree, the second call deletes the right subtree, and the final statement deletes the root node. In the pictured reasoning, the overall specification of the procedure is applied as an induction hypothesis at each call site, together with the Frame Rule for showing that the parts not touched by recursive calls are left unchanged. For instance, the assertions for the second recursive call are an instance of the Frame Rule with the triple $\{tree(right)\} deletetree(right) \{emp\}$ as the premise.

The simplicity of this proof comes about because of the principle of local reasoning. The frame rule allows in-place reasoning for larger-scale operations (entire procedures) than individual heap mutations. And it allows the specification to concentrate on the footprint of a procedure instead of the global state. Put contrapositively, the *deletetree* procedure could not be verified without the frame rule, unless we were to complicate the initial specification by including some representation of frame axioms (saying what does not change) to enable the proofs at the recursive call sites.

This reasoning uses a tree predicate suitable for reasoning about memory safety; it mentions that we have a tree, but not what data it holds. For functional correctness reasoning, it is typical to use inductive predicates that connect memory structures to mathematical entities. In place of *tree* (*root*) we could have a predicate *tree* (τ , *root*) that says *root* points to an area of memory representing the mathematical binary tree τ , where a mathematical tree is either empty or an atom or a pair of trees. We could then specify a procedure for copying a tree using a postcondition of the form

$$tree(\tau, oldroot) * tree(\tau, newroot)$$

that says we have two structures in memory representing the same mathemati-

cal tree. An assertion like this would tell us that we could mutate one of the trees without affecting the other (at which point they would cease to represent the same tree).

For data structures without much sharing, such as variations on lists and trees, reasoning in SL is reminiscent of reasoning about purely functional programs: you unroll an inductive definition, then mutate, then roll it back up. Inductive definitions using $*$ and mutation go well together. The first SL proof to address complex sharing was done by Yang in his Ph.D. thesis, where he provided a verification of the classic Schorr-Waite graph-marking algorithm. The algorithm works by reversing links during search, and then restoring them later: A space-saving representation of the stack of a recursive algorithm. Part of the main invariant in Yang's proof is

$$\begin{aligned} & (listMarkedNodesR(stack, p) \\ & * (restoredListR(stack, t) \multimap \\ & spansR(STree, root))) \end{aligned}$$

capturing the idea that if you replace the list of marked nodes by a restored list, then you get a spanning tree. Yang's proof reflected the intuition that the algorithm works by a series of local surgeries that mutate small parts of the structure: The proof decomposed into verifications of the surgeries, and ways of combining them.

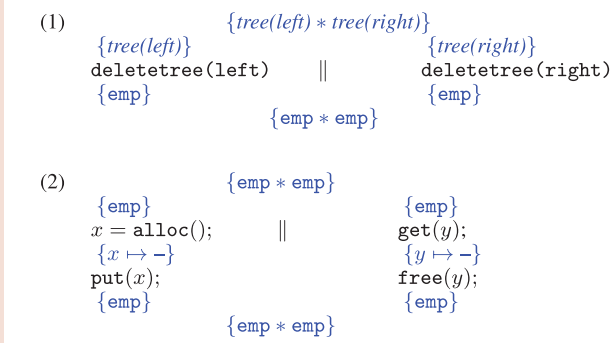
The idiomatic use of $\multimap$ in assertions of the form $A * (B \multimap C)$ to describe generalized update was elevated to a general principle in work of Hobor and Villard.²⁵ They give proofs of a number of programs with significant sharing, including graphs, dags, overlaid structures (for example, a list overlaying a tree), and culminating in the copying algorithm in Cheney's garbage collector.

Many papers on SL have avoided $\multimap$, often on the grounds that it complicates automation and is only needed for programs with significant sharing. However, $\multimap$ is recently making something of a comeback. For example, it is used routinely as a basic tool in the Iris higher-order logic.²⁹

Concurrency, Ownership, and Separation

The concurrency rule in Figure 3 says: To prove a parallel composition we give each process a separate piece of state, and separately combine the postcon-

^b There are variable conditions in some presentations of SL, that can technically be done away with eliminated by using a version of $*$ that separates variables as well as heap.³⁴ This article glosses over this issue.

Figure 6. Concurrency proofs.

ditions for each process. The rule supports completely independent reasoning about processes. This rule can be used to provide straightforward proofs of processes that don't share access to storage. We mentioned parallel quicksort earlier, and `deletetree()` provides another illustration: we can run the two recursive calls in parallel rather than sequentially, as presented in the proof outline (1) in Figure 6.

In work on CSL, proof outlines are often presented in a spatial fashion like this: this outline shows the premises of the concurrency rule in the left and right Hoare triples, the overall precondition (the $pre1 * pre2$) at the beginning, and the post at the end.

While this reasoning is simple, if CSL had only been able to reason about disjoint concurrency, where there is no inter-process interaction, then it would have rightly been considered rather restrictive. An important early example done with CSL was a pointer-transferring buffer, where one thread allocates a pointer and puts it into a buffer while the other thread reads it out and frees it. Crucially, not only is the pointer deemed to transfer from one process to another, but the “knowledge that it is allocated” transfers with the proof. The proof establishing absence of memory errors is shown in (2) of Figure 6. A way to implement the buffer code for `put` and `get` is to use locks to synchronize access to a shared variable and a Boolean to signal when the buffer is full. We will not delve into the subproofs of buffer operations here—for that, consult O’Hearn³⁰—but we want to talk about a shift in perspective on the meanings of logical assertions that the proof (2) led to.

Notice the assertion `emp` after the `put(x)` statement in the left process.

We could not prove a mutation were we to place it there, because `emp` is not a sufficient precondition for any mutation; that is fortunate as such a mutation could lead to a race condition. But it is not the case that we know the global heap is empty, because the pointer `x` could still persist. Rather, the knowledge that it points to something has been forgotten, transferred to the second process where it materializes as $y \mapsto -$. A reading of assertions began to form based on the “right to dereference” or “ownership” (taken as synonymous with right to dereference). On this reading `emp` says “I don’t have permission to dereference any heap,” or “I own nothing,” rather than “the heap is empty.” Similarly, $x \mapsto -$ says “I own `x`” (where “I” is the process from which the assertion is made).

The ownership transfer example made it clear that quite a few concurrent programs would have much simpler proofs than before. Modular proofs were provided of semaphore programs, of a toy memory manager, and programs with interacting resources. It seemed as if the proofs mirrored design principles used to simplify reasoning about concurrent processes, such as in Dijkstra’s idea of *loosely connected processes*:

“[A]part from the (rare) moments of explicit intercommunication, the individual processes are to be regarded as completely independent of each other.”¹⁶

However, the very feature that gave rise to the unexpected power, ownership transfer, made soundness (whether the rules prove only true statements) non-obvious. O’Hearn worked on soundness during 2001 and 2002, without success. In May of 2002 he turned to Brookes who eventually (with important input from

Reynolds), in 2004, proved the theorem, which justified the logic.

Abstraction and the Fiction of Separation

There was considerable work on extending SL after those early papers. Some of it concentrated on different programming paradigms, such as object-oriented programming or scripting languages, or on additional programming primitives such as message passing, reentrant lock and fork/join concurrency. Besides extensions to cover an ever-greater variety of programming, two conceptual developments opened major new directions.

► In his Ph.D. thesis, Parkinson showed how abstract predicates (predicate variables) fit together nearly with $*$ in the description of classes and other stateful data abstractions.³³

► Gardner and others emphasized a concept of fictional separation, where strong separation properties could be assumed of data abstractions, even for implementations relying on sharing.

These ideas were first described in a sequential setting. Dinsdale-Young, Gardner and Wheelhouse described an implementation of a module of sequences in terms of linked lists and noted a mismatch: at the abstract level an operation might affect a small part of a sequence, where at the implementation level its footprint could involve the entire list; conversely, locality can increase with abstraction.¹⁹ Meanwhile, Parkinson initially targeted a sequential subset of Java. Subsequent work showed how abstract predicates could be understood using higher-order versions of SL.⁵

While they could be expressed in a sequential setting, the ideas took flight when transported to concurrency. The CAP logic¹⁸ combined insights on abstract predicates and fiction, along with those of CSL, to reason about data abstractions with interference in their implementations. The views theory¹⁷ provided a foundation where separation does not appear in the normal execution semantics of programs, but only in an abstraction of it. Views showed that a simple version of CSL can embed many other techniques including even the classic rely-guarantee method;²⁷ this is surprising because rely-guarantee was invented for reasoning about interference, almost the opposite of the basis of original SL.

Today, advanced logics are often formulated as variations on the theme of “higher-order concurrent separation logic.” One of these, Verifiable C, is the foundation of Appel’s Verified Software Toolchain,¹ and includes an expressive higher-order logic supporting recursive predicates. Iris²⁹ encompasses reasoning about fine-grained concurrency and even relaxed memory, based on different instantiations of a single generic model. Iris has been used to provide a foundation of the type system of the Rust programming language,²⁸ which is very natural when you consider that ownership transfer is one of the central ideas in Rust.

Technically, these works are based on “non-standard models” of SL, different from the heaplet model but instances of Pym’s resource semantics as in Figure 2; see Pym et al.³⁶ There are many such models, including ones incorporating read and other permissions,⁷ auxiliary state,³⁹ time,³⁹ protocols,²⁹ and others. Abstract SL¹³ showed how general program logic could be defined based on these models, and the works just mentioned and others showed that some of them had surprising ramifications.

Fictional separation and views worked to reimagine fundamental concepts. The programs being proven go beyond the loosely connected processes that CSL was originally designed for. Significant new theoretical insights and soundness arguments were needed to justify the program-proof rules supporting the fine-grained concurrency examples.¹⁷ This led to a flowering of interest and new ideas which is still in progress. A recent survey on CSL provides many more references in addition to those mentioned here.⁹

Directions in Mechanized Reasoning

SL spawned new approaches to verification tools. In order to provide a taste of where the field has gone, we present a sampling of practical achievements; that is, we focus on the end points rather than the (important) advancements along the way that helped get there. Further references to the literature, including discussion on intermediate advances, may be found in the appendix (<https://bit.ly/2CQD9CU>).

Mostly automatic verification. *Smallfoot*,² from Calcagno, Berdine, and

O’Hearn, was the first SL verification tool. Given procedure pre/post specs, loop invariants and invariants governing lock usage, Smallfoot attempts to construct a proof. For the pointer-transferring buffer, given a buffer invariant and pre/post specs for put and get it can verify memory safety and race freedom.

Smallfoot used a decidable fragment of SL dubbed “symbolic heap,” formulae of the form $B \wedge H$ where H is a separating conjunction of heap facts and B is a Boolean assertion over non-heap data. The format was chosen to make in-place symbolic execution efficient. Smallfoot’s heap facts were restricted to points-to assertions, linked lists and trees. Subsequent works extended symbolic heaps in numerous directions, covering more inductive definitions as well as arrays and arithmetic; see appendix (<https://bit.ly/2CQD9CU>).

Some of the most substantial automatic verifications done with SL have been carried out with the VeriFast tool of Jacobs and colleagues. VeriFast employs a symbolic execution engine like Smallfoot, but integrates a dedicated SL theorem prover with a classical SMT solver for non-heap data. A paper reports on the verification of several industrial case studies, including Java Card programs and device drivers written in C;³⁵ see VeriFast’s GitHub site for these and many other examples (<https://github.com/verifast/verifast>).

Interactive verification. In an automatic verifier like Smallfoot, the proof construction is automatic, given the pre/post annotations plus invariants. In interactive verification the human helps guide the proof search, commonly using a proof assistant such as Coq, HOL4, or Isabelle. Interactive verification can often prove stronger properties than automatic verifiers, but the cost is higher.

Interactive verifiers have been used to prove small, intricate algorithms. A recent paper reports on the verification of low-level concurrent algorithms including a CAS-lock, a ticketed lock, a GC allocator, and a non-blocking stack.³⁹ An emphasis is placed on reusability; for instance, the stack uses the GC allocator, which in turn uses a lock, but the stack uses the spec of the allocator and the allocator uses the spec rather than the implementation of a lock.

The verifiable C logic¹ has been

used to prove crypto code. For example, OpenSSL’s HMAC authentication code, comprising 134 lines of C, was proven using 2,832 lines of Coq.⁴

A larger example is the FSCQ file system.¹⁴ The code and the proof are both done in Coq, taking up 31k lines of proof+code. This compares to 3k lines of C for a related unverified file system. Although the initial effort, which included development of a program logic framework in Coq, took several person years, experiments show incremental, lower cost when modifying code+proof.

A commercial example concerns key modules of a preemptive OS kernel, the μ C/OS-II.⁴¹ Modules verified include the scheduler, interrupt handlers, and message queues. 1.3k lines of C were proven using 216k lines of Coq. It took four person years to develop the framework, one-person year to prove the first module, and then the remaining modules, around 900 lines of C, took six person-months.

Automatic program analysis. With a verification-oriented program analysis the annotations that a human would supply to a mostly automatic verifier like Smallfoot—invariants and pre/post specs—are inferred. A tool will be able to prove weaker properties when the human is not supplying annotations, but can more easily be deployed broadly to many programmers.

Program analysis with SL has received a great deal of attention. At first, analysis was formulated for simple linked lists,²⁰ and progressively researchers moved on to more involved data structures. A practical high point in this line of work was the verification of pointer safety in Linux and Windows device drivers up to 10k LOC by the SpaceInvader program analyzer.⁴³ SpaceInvader was an academic tool; its sibling, SLAYER,³ developed in parallel at Microsoft, was used internally to find 10s of memory safety errors in Windows device drivers. SpaceInvader and SLAYER were able to analyze complex, linear data structures: for example, one Windows driver manipulated five-cyclic doubly linked lists sharing a common header node, three of which had acyclic sublists.

Like much research in verification-oriented program analysis these techniques worked in a whole-program fashion: you start from `main()` or

other entry points and explore the program graph, perhaps visiting procedure bodies multiple times. This can be expensive. While accurate analysis of 10k LOC can be a leading research achievement, 10k is tiny compared to software found in the wild. A single company can have tens of millions of lines of code. Progress toward big code called for a radical departure.

Bi-Abduction and Facebook Infer

In 2008 Calcagno asked: What is the main obstacle blocking application of SpaceInvader and similar tools to programs in the millions of LOC? O'Hearn answered: The need for the human to supply preconditions. He proposed that a “truly modular” analysis based on local reasoning could accept a program component with no human annotations, and generate a pre/post spec where the precondition approximates the footprint. The analysis would then “stitch” these specifications together to obtain results for larger program parts. The analysis would be compositional, in that a spec for a procedure could be obtained without knowing its callers, and the hypothesis was that it would scale because procedures could be visited independently. This implied giving up on whole-program analysis.

Calcagno, O'Hearn, Distefano and Yang set to work on realizing a truly modular analysis. Yang developed a scheme based on gleaning information from failed proofs to discover a footprint. Distefano made a breakthrough on the stitching issue for the modular analysis that involved a new inference problem:

Bi-abduction: given A and B , find $?frame$ and $?anti-frame$ such that

$$A * ?anti-frame \vdash B * ?frame$$

where $\vdash$ is read ‘entails’ or ‘implies.’ The inference of $?frame$ (the leftover part in A but not B) was present in Smallfoot, and is used in many tools. The $?anti-frame$ part (the missing bit needed to establish B), is *abduction*, or inference of hypotheses, an inference problem identified by the philosopher Charles Peirce in his conceptual analysis of the scientific method. As a simple example,

$$(x \mapsto -) * ?anti-frame \vdash (y \mapsto -) * ?frame.$$

can be solved with

$$?anti-frame = y \mapsto - \text{ and } ?frame = x \mapsto -.$$

With bi-abduction we can automate the local reasoning idea by abducting assertions that describe preconditions, and using frame inference to keep specifications small. Let us illustrate with the program we started the paper with. We begin symbolic execution with nothing in the precondition, and we ask a bi-abduction question, using the current state emp as the A part of the bi-abduction query and the pre of the small axiom for $[x] = y$ as B .

- Execution state:
 $\{\text{emp}\}[x] = y; [y] = x \{???\}$
- Bi-abduction query:
 $\text{emp} * ?anti-frame \vdash x \mapsto - * ?frame$
- Solution:
 $?anti-frame = x \mapsto -; ?frame = \text{emp}.$

Now, we move the abducted anti-frame to the overall precondition, we take one step of symbolic execution using the small axiom for Pointer Write from Figure 2, we install the post of the small axiom as the pre of the next instruction, and we continue.

- Execution state:
 $\{x \mapsto -\}[x] = y \{x \mapsto y\} [y] = x \{???\}$
- Bi-abduction query:
 $x \mapsto y * ?anti-frame \vdash y \mapsto - * ?frame$
- Solution:
 $?anti-frame = y \mapsto -; ?frame = x \mapsto y.$

The formula $y \mapsto -$ in the bi-abduction query is the precondition of the small axiom for the pointer write $[y] = x$: we abduce it as the anti-frame, and add it to the overall precondition. The frame rule tells us that the inferred frame $x \mapsto y$ is unaltered by $[y] = x$, when it is separately conjoined with $y \mapsto -$, and this with the small axiom gives us our overall postcondition in

$$\begin{aligned} &\{x \mapsto - * y \mapsto -\} \\ &\quad [x] = y; \\ &\quad [y] = x \\ &\{x \mapsto y * y \mapsto x\} \end{aligned}$$

So, starting from specifications for primitive statements, we can infer both a precondition and a postcondition for a compound statement by repeated applications of bi-abduction and the frame rule. This facility leads to a high degree of automation. Also, note that the precondition here is more general than the one at the start of the paper, because it does not mention 0. Bi-abductive analy-

sis not infrequently finds more general specifications than a top-down analysis that dives into procedures at call sites; finding general specs is important for both scalability and precision.

The main bi-abduction paper¹² contributed proof techniques and algorithms for abduction, and a novel compositional algorithm for generating pre/post specs of program components. Experimental results scaled to hundreds of thousands of lines, and a part of Linux of 3M lines. This form of analysis finds preconditions supporting safety proofs of clusters of procedures as well as indicating potential bugs where proofs failed.

This work led to the program proof startup Monoidics, founded by Calcagno, Distefano and O'Hearn in 2009. Monoidics developed and marketed the Infer tool, based on the abductive technique. Monoidics was acquired by Facebook in 2013 at which point Calcagno, Distefano, and O'Hearn moved to Facebook with the Monoidics engineering team (www.fbinfer.com).

The compositional nature of Infer turned out to be a remarkable fit for Facebook's software development process.¹¹ A codebase with millions of lines is altered thousands of times per day in “code diffs” submitted by the programmers. Instead of doing a whole-program analysis for each diff, Infer analyzes changes (the diffs) compositionally, and reports regressions as a bot participating in the internal code review process. Using bi-abduction, the frame rule picks off (an approximation of) just enough state to analyze a diff, instead of considering the entire global program state. The way that compositional analysis supports incremental diff analysis is even more important than the ability to scale; a linear-time analysis operating on the whole program would usually be too slow for this deployment model. Indeed, Infer has evolved from a standalone SL-based analyzer to a general framework for compositional analyses (<http://fbinfer.com/docs/checkers.html> and appendix; <https://bit.ly/2CQD9CU>).

Conclusion

Some time during 2001, while sitting together in his back garden, Reynolds turned to me and exclaimed: “The

logic is nice, but it's the model that's really important." My own prejudice for semantics made me agree immediately. We were both beguiled by the fact that this funky species of logic could be described using down-to-earth computer science concepts like RAMs and access bits.

What happened later came as a surprise. The specific heap/RAM model gave way in importance to a more general class of nonstandard models based on fictional rather than down-to-earth separation. And the logic itself, particularly its proof theory, turned out to be extremely useful in automatic verification, leading to many novel research tools and eventually to Facebook Infer.

Still, I expect that in the long run it will be the spirit rather than the letter of SL that is more significant. Concepts of frames, footprints, and separation as a basis for modular reasoning seem to be of fundamental importance, independently of the syntax used to describe them. Indeed, one of the more important directions I see for further work is in theoretical foundations that get at the essence of scalable, modular reasoning in as formalism-independent a way as possible. Theoretical synthesis would be extremely useful for three reasons: To make it easier for people to understand what has been achieved by each new idea; to provide a simpler jumping-off point for future work than the union of the many specific advances; and, to suggest new, unexplored avenues. Hoare has been advancing an abstract, algebraic theory related to CSL, which has components covering semantics, proof theory, and testing,²⁴ and work along these lines is well worth exploring further.

Other relevant reference points are works on general versions of SL,^{13,17} abstract interpretation,¹⁵ and work on "separation without SL" discussed in the appendix (<https://bit.ly/2CQD9CU>). Semantic fundamentals would be crucial to an adequate general foundation, but I stress that proof theoretic and especially algorithmic aspects addressing the central problem of scale should be covered as well.

In conclusion, scalable reasoning about code has come a long way since the birth of SL around the turn of the millennium, but it seems to me that much more is possible both in funda-

mental understanding and in mechanized techniques that help programmers in their daily work. I hope that scientists and engineers will continue to innovate on the fascinating problems in this area.

Acknowledgments. *This article is dedicated to the memory of John C. Reynolds (1935–2013).* Our work together at the formative stage of separation logic was incredibly intense, exciting, and huge fun. I am fortunate to have worked so closely with such a brilliantly insightful scientist, who was also a valued friend.

I thank my many other collaborators in the development of this research, particularly David Pym, Hongseok Yang, Richard Bornat, Cristiano Calcagno, Josh Berdine, Dino Distefano, Steve Brookes, Matthew Parkinson, Philippa Gardner, and Tony Hoare. Finally, thanks to my colleagues at Facebook for our work together and for teaching me about applying logic in the real world. 

References

- Appel, A.W. *Program Logics for Certified Compilers*. Cambridge University Press, U.K., 2014.
- Berdine, J., Calcagno, C. and O'Hearn, P.W. Smallfoot: Modular automatic assertion checking with separation logic. *LNCSE FMO 4111* (2005) 115–137, 2005.
- Berdine, J., Cook, B. and Ishtiaq, S. SLayer: Memory safety for systems-level code. In *Proceedings of CAV*, 2011, 178–183.
- Beringer, L., Petcher, A., Ye, K.Q. and Appel, A.W. Verified correctness and security of OpenSSL HMAC. In *Proceedings of 24th USENIX Security Symposium*, 2015, 207–221.
- Biering, B., Birkedal, L. and Torp-Smith, N. BI-hyperdoctrines, higher-order separation logic, and abstraction. *ACM TOPLAS* 29, 4 (2007).
- Bornat, R. Proving pointer programs in Hoare logic. *LNCSE MPC 1837* (2000) 102–126.
- Bornat, R., Calcagno, C., O'Hearn, P.W. and Parkinson, M.J. Permission accounting in separation logic. In *Proceedings of POPL*, 2005, 259–270.
- Brookes, S. A semantics for concurrent separation logic. *Theor. Comput. Sci.*, 375, 1–3 (2007), 227–270.
- Brookes, S. and O'Hearn, P.W. Concurrent separation logic. *SIGLOG News* 3, 3 (2016), 47–65.
- Burstall, R.M. Some techniques for proving correctness of programs which alter data structures. *Machine Intelligence* 7, 1 (1972), 23–50.
- Calcagno, C. et al. Moving fast with software verification. In *Proceedings of NASA Formal Methods Symposium*, 2015, 3–11.
- Calcagno, C., Distefano, D., O'Hearn, P.W. and Yang, H. Compositional shape analysis by means of bi-abduction. *J. ACM* 58, 6 (2011), 26. Preliminary version in *Proceedings of POPL* 09.
- Calcagno, C., O'Hearn, P.W. and Yang, H. Local action and abstract separation logic. *LICS*, 2007, 366–378.
- Chen, H., Ziegler, F., Chajed, T., Chlipala, A., Kaashoek, M.F. and Zeldovich, N. Using Crash Hoare logic for certifying the FSCQ file system. In *Proceedings of SOSR*, pages 18–37, 2015.
- Cousot, P. and Cousot, R. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of POPL*, 1977, 238–252.
- Dijkstra, E.W. *Cooperating sequential processes*. *Programming Languages*, Academic Press, 1968, 43–112.
- Dinsdale-Young, T., Birkedal, L., Gardner, P., Parkinson, M.J. and Yang, H. Views: Compositional reasoning for concurrent programs. In *Proceedings of POPL*, 2013, 287–300.
- Dinsdale-Young, T., Dodds, M., Gardner, M., Parkinson, M.J. and Vafeiadis, V. Concurrent abstract predicates. In *Proceedings of ECOOP*, 2010, 504–528.
- Dinsdale-Young, T., Gardner, P. and Wheelhouse, M.J. Abstraction and refinement for local reasoning. In *Proceedings of VSTTE*, 2010, 199–215.
- Distefano, D., O'Hearn, P.W. and Yang, H. A local shape analysis based on separation logic. In *Proceedings of TACAS*, 2006, 287–302.
- Floyd, R.W. Assigning meanings to programs. In *Proceedings of the Symposium on Applied Mathematics*. J.T. Schwartz, ed. AMS, 1967, 19–32.
- Hoare, C.A.R. An axiomatic basis for computer programming. *Commun. ACM* 12, 10 (1969), 576–580.
- Hoare, C.A.R. Towards a theory of parallel programming. *Operating Systems Techniques*. Academic Press, 1972.
- Hoare, T., Möller, B., Struth, G. and Wehrman, I. Concurrent Kleene algebra and its foundations. *J. Log. Algebr. Program* 80, 6 (2011), 266–296.
- Hobor, A. and Villard, J. The ramifications of sharing in data structures. In *Proceedings of 40th POPL*, 2013, 523–536.
- Ishtiaq, S.S. and O'Hearn, P.W. BI as an assertion language for mutable data structures. In *Proceedings of POPL*, 2001, 14–26.
- Jones, C.B. Specification and design of (parallel) programs. In *Proceedings of IFIP Congress*, 1983, 321–332.
- Jung, R., Jourdan, J.-H., Krebbers, R. and Dreyer, D. RustBelt: Securing the foundations of the Rust programming language. In *Proceedings of PACMPL*, 2018.
- Krebbers, R., Jung, R., Bizjak, A., Jourdan, J.-H., Dreyer, D. and Birkedal, L. The essence of higher-order concurrent separation logic. In *Proceedings of ESOP*, 2017, 696–723.
- O'Hearn, P.W. Resources, concurrency, and local reasoning. *Theor. Comput. Sci.* 375, 1–3 (2007), 271–307.
- O'Hearn, P.W. and Pym, D.J. The logic of bunched implications. *Bulletin of Symbolic Logic* 5, 2 (1999), 215–244.
- O'Hearn, P.W., Reynolds, J.C. and Yang, H. Local reasoning about programs that alter data structures. In *Proceedings of CSL*, 2001, 1–19.
- Parkinson, M.J. Local reasoning for Java. Ph.D. thesis. University of Cambridge, U.K., 2005.
- Parkinson, M.J., Bornat, R. and Calcagno, C. Variables as resource in Hoare logics. In *Proceedings of 21st LICS*, 2006, 137–146.
- Philippaerts, P., Mühlberg, J.T., Penninckx, W., Smans, J., Jacobs, B. and Piessens, F. Software verification with verifast: Industrial case studies. *Sci. Comput. Program.* 82 (2014), 77–97.
- Pym, D., O'Hearn, P. and Yang, H. Possible worlds and resources: The semantics of BI. *Theoret. Comp. Sci.* 315, 1 (2004), 257–305.
- Reynolds, J.C. Intuitionistic reasoning about shared mutable data structure. *Millennial Perspectives in Computer Science, Cornerstones of Computing*. Palgrave Macmillan, 2000.
- Reynolds, J.C. Separation logic: A logic for shared mutable data structures. *LICS*, 2002, 55–74.
- Sergey, I., Nanevski, A. and Banerjee, A. Mechanized verification of fine-grained concurrent programs. In *Proceedings of 36th PLDI*, 2015, 77–87.
- Turing, A.M. Checking a large routine. *Report of a Conference on High-Speed Automatic Calculating Machines*. Univ. Math. Lab., Cambridge, U.K., 1949, 67–69.
- Xu, F., Fu, M., Feng, X., Zhang, X., Zhang, H. and Li, Z. A practical verification framework for preemptive OS kernels. In *Proceedings of CAV*, 2016.
- Yang, H. Local Reasoning for Stateful Programs. Ph.D. thesis. University of Illinois, 2001.
- Yang, H., Lee, O., Berdine, J., Calcagno, C., Cook, B., Distefano, D. and O'Hearn, P.W. Scalable shape analysis for systems code. In *Proceedings of CAV*, 2008, 385–398.
- Yang, H. and O'Hearn, P.W. A semantic basis for local reasoning. In *Proceedings of FoSSaCS*, 2002, 402–416.

Peter O'Hearn (p.ohearn@ucl.ac.uk) is a research scientist at Facebook and professor of computer science at University College London, U.K.

research highlights

P. 97

**Technical
Perspective**
**How Economic
Theories Can
Help Computers
Beat the Heat**

By Thomas F. Wenisch

P. 98

Distributed Strategies for Computational Sprints

By Songchun Fan, Seyed Majid Zahedi, and Benjamin C. Lee

P. 107

**Technical
Perspective**
**To Do or Not to Do:
Extending SQL
with Integer Linear
Programming?**

By Surajit Chaudhuri

P. 108

Scalable Computation of High- Order Optimization Queries

By Matteo Brucato, Azza Abouzied, and Alexandra Meliou

Technical Perspective

How Economic Theories Can Help Computers Beat the Heat

By Thomas F. Wenisch

NEARLY EVERY COMPUTER system today runs hot ... *too* hot. For over a decade, thermal constraints have limited the computational capability of computing systems of all sizes—from mobile phones to datacenters. And, for nearly that long, system designers have cheated those thermal limits, allowing systems to burn more power, and produce more heat, for short periods to deliver bursts of peak performance beyond what can be sustained. This idea—running a computer too hot for a short period of time to get a burst of performance—is called *computational sprinting*.

We have likely all experienced computational sprinting on our smartphones; it turns out that, if all the cores, accelerators, and peripherals on a modern smartphone are turned on at once, the phone will generate several times more heat than can be dissipated through its case. If you play a demanding 3D video game for more than a few minutes, you might notice the phone get uncomfortably warm. As the phone heats up, eventually, processing speeds have to slow to keep temperature rise in check. When the phone cools, its processor can run full-tilt again.

What might be less widely known is that modern datacenters can play similar tricks; they oversubscribe both power delivery and cooling capability to eke out greater efficiency. Individual servers may sprint by consuming more than their fair share of power to maximize performance when their workload is high. In a datacenter running diverse workloads, different systems will likely sprint at different times, and the average demands of the facility will (probably) remain sustainable. But, a local spike in one server rack might draw too much power from a particular circuit, risking that a circuit breaker trips. Or, all the cores in a particular server might run a sustained compute job at full bore and risk local over-heating. To maximize efficiency, a datacenter should sprint as close to its power and thermal limits as it can ... without going over them.

Current datacenters must either run complex, centralized control systems to allocate power and thermal budgets at fine granularity, or reserve large guardbands to avoid power or thermal emergencies. But, because they require frequent communication, centralized systems are prone to failure and notoriously difficult to scale—the frequent communication rapidly becomes a bottleneck. Moreover, workloads benefit to different degrees at different times from computational sprinting; judicious use of scarce power and cooling budgets can lead to better overall performance. The challenges of allocating budgets grow even more daunting in cloud computing environments, where each cloud tenant seeks to maximize its own performance and may have no incentive to cooperate.

Economics has long studied the challenges of allocating scarce resources. Game theory, in particular, studies resource allocation among strategic agents that seek to maximize their individual utility and might even lie about their preferences to do so.

The authors of the following paper, *Distributed Strategies for Computational Sprints*, bring this rich theory to the challenge of managing computational sprinting in datacenters. They formulate the problem of managing computational sprinting as a repeated game: agents managing individual workloads

are free to choose when to sprint, but must wait for a cool-off period before sprinting again. Moreover, if too many nodes sprint at once, supplemental battery power must be used to avoid tripping circuit breakers; servers connected to that power circuit are not allowed to sprint again until the battery recharges. To “win” in this game, agents must choose to sprint when they achieve the maximum performance benefit while taking into account the risk they incur that too many concurrent sprinters cause a circuit to trip.

To optimize the datacenter as a whole, each agent provides a broker with its best estimate of its utility curve—how much benefit it gains from sprinting for various fractions of its execution while taking into account the risks of a circuit breaker trip. The broker then solves for a global equilibrium that maximizes utility, and provides each agent the strategy it should follow to reach that equilibrium. The strength of the underlying economic theory is that agents probably cannot gain an advantage from lying about their utility curve or deviating from their assigned strategy ... so, they are incentivized to cooperate.

The beauty of this approach is that it provides nearly the effectiveness of perfect centralized control while requiring only simple, infrequent interactions with the broker. Because agents cannot gain an advantage by cheating, this kind of coordination mechanism can be used even among mutually distrusting agents, as in the cloud. More generally, the paper teaches us that, when we consider the myriad resource management challenges that arise in computer systems, we ought to look beyond the confines of our own discipline; economics provides a rich toolset from which all of us can learn. 

Thomas F. Wenisch is an associate professor of computer science and engineering at the University of Michigan, Ann Arbor, MI, USA.

Copyright held by author/owner.

When we consider the resource management challenges that arise in computer systems, we should look beyond the confines of our own discipline.

Distributed Strategies for Computational Sprints

By Songchun Fan,[†] Seyed Majid Zahedi,[†] and Benjamin C. Lee

Abstract

Computational sprinting is a class of mechanisms that boost performance but dissipate additional power. We describe a sprinting architecture in which many, independent chip multiprocessors share a power supply and sprints are constrained by the chips' thermal limits and the rack's power limits. Moreover, we present the computational sprinting game, a multi-agent perspective on managing sprints. Strategic agents decide whether to sprint based on application phases and system conditions. The game produces an equilibrium that improves task throughput for data analytics workloads by 4–6× over prior greedy heuristics and performs within 90% of an upper bound on throughput from a globally optimized policy.

1. INTRODUCTION

Modern datacenters oversubscribe their power supplies to enhance performance and efficiency. A conservative datacenter that deploys servers according to their expected power draw will under-utilize provisioned power, operate power supplies at sub-optimal loads, and forgo opportunities for higher performance. In contrast, efficient datacenters deploy more servers than it can power fully and rely on varying computational load across servers to modulate demand for power.⁴ Such a strategy requires responsive mechanisms for delivering power to the computation that needs it most.

Computational sprinting is a class of mechanisms that supply additional power for short durations to enhance performance. In chip multiprocessors, for example, sprints activate additional cores and boost their voltage and frequency. Although originally proposed for mobile systems,^{13,14} sprinting has found numerous applications in datacenter systems. It can accelerate computation for complex tasks or accommodate transient activity spikes.^{16,21}

The system architecture determines sprint duration and frequency. Sprinting multiprocessors generate extra heat, absorbed by thermal packages and phase change materials (PCMs),^{14,16} and require time to release this heat between sprints. At scale, uncoordinated multiprocessors that sprint simultaneously could overwhelm a rack or cluster's power supply. Uninterruptible power supplies reduce the risk of tripping circuit breakers and triggering power emergencies. But the system requires time to recharge batteries between sprints. Given these physical constraints in chip multiprocessors and the datacenter rack, sprinters require recovery time. Thus, sprinting

mechanisms couple performance opportunities with management constraints.

We face fundamental management questions when servers sprint independently but share a power supply – which processors should sprint and when should they sprint? Each processor's workload derives extra performance from sprinting that depends on its computational phase. Ideally, sprinters would be the processors that benefit most from boosted capability at any given time. Moreover, the number of sprinters would be small enough to avoid power emergencies, which constrain future sprints. Policies that achieve these goals are prerequisites for sprinting to full advantage.

We present the computational sprinting game to manage a collection of sprinters. The sprinting architecture, which defines the sprinting mechanism as well as power and cooling constraints, determines rules of the game. A strategic agent, representing a multiprocessor and its workload, independently decides whether to sprint at the beginning of an epoch. The agent anticipates her action's outcomes, knowing that the chip must cool before sprinting again. Moreover, she analyzes system dynamics, accounting for competitors' decisions and risk of power emergencies.

We find the equilibrium in the computational sprinting game, which permits distributed management. In an equilibrium, no agent can benefit by deviating from her optimal strategy. The datacenter relies on agents' incentives to decentralize management as each agent self-enforces her part of the sprinting policy. Decentralized equilibria allow datacenters to avoid high communication costs and unwieldy enforcement mechanisms in centralized management. Moreover, equilibria outperform prior heuristics.

2. THE SPRINTING ARCHITECTURE

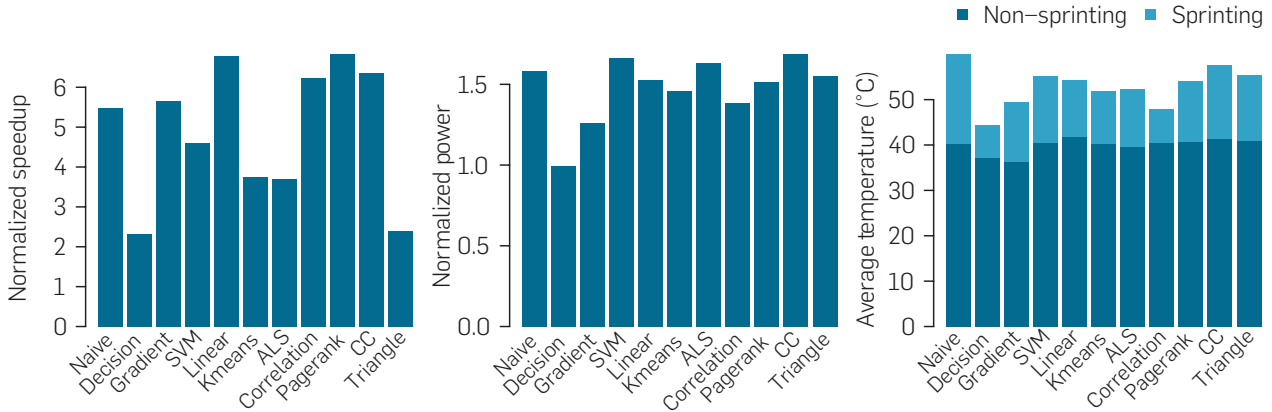
We present a sprinting architecture for chip multiprocessors in datacenters. Multiprocessors sprint by activating additional cores and increasing their voltage and frequency. Datacenter applications, with their abundant task parallelism, scale across additional cores as they become available. In Figure 1, Spark benchmarks perform 2–7× better on a sprinting multiprocessor, but dissipates 1.8× the power. Power produces heat.

Sprinters require infrastructure to manage heat and power. First, the chip multiprocessor's thermal package

The original version of this paper is entitled “The Computational Sprinting Game” and was published in *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems* (2016), ACM, NY.

[†] These authors contributed equally to this work.

Figure 1. Normalized speedup, power, and temperature for varied Spark benchmarks when sprinting. Nominal operation supplies three cores at 1.2GHz. Sprint supplies twelve cores at 2.7GHz.



and heat sink must absorb surplus heat during a sprint.^{14, 15} Second, the datacenter rack must employ batteries to guard against power emergencies caused by a surplus of sprinters on a shared power supply. Third, the system must implement management policies that determine which chips sprint.

2.1. System architecture

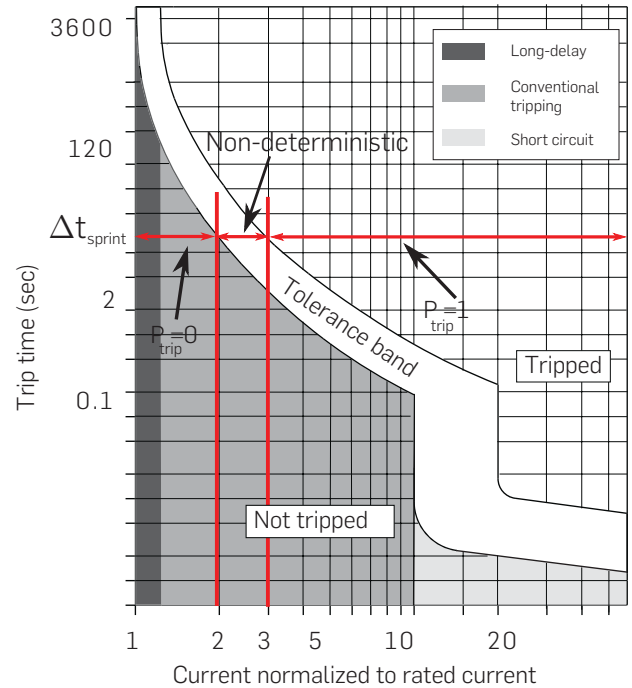
Chip multiprocessors and thermal packages. The quality of the multiprocessor's thermal package, measured by its thermal capacitance and conductance, determines the chip's maximum power level and dictates the duration of a sprint.^{13, 15} More expensive heat sinks employ PCMs, which increase thermal capacitance, and permit sprint durations on the order of minutes if not hours. We estimate a chip with paraffin wax can sprint with durations on the order of 150s.

After a sprint, the thermal package must release its heat before the chip can sprint again. The average cooling duration, denoted as Δt_{cool} , is the time required before the PCM returns to ambient temperature. The rate at which the PCM dissipates heat depends on its melting point and the thermal resistance between the material and the ambient. Both factors can be engineered and, with paraffin wax, we estimate a cooling duration on the order of 300s, twice the sprint's duration.

Power delivery and circuit breakers. Datacenter architects deploy servers and multiprocessors to oversubscribe power distribution units for efficiency. Oversubscription utilizes a larger fraction of the facility's provisioned power. But it relies on power capping and varied computational load across servers to avoid tripping circuit breakers or violating contracts with utility providers.⁴ Although sprints can boost computation, the risk of a power emergency increases with the number of sprinters in a power capped datacenter.

Figure 2 presents the circuit breaker's trip curve, which specifies how sprint duration and power combine to determine whether the breaker trips. The trip time corresponds to the sprint's duration. Longer sprints increase the probability of tripping the breaker. The current draw corresponds

Figure 2. Typical trip curve of a circuit breaker.⁵



to the number of simultaneous sprints as each sprinter contributes to the load above rated current. Higher currents increase the probability of tripping the breaker.

Let n_s denote the number of sprinters and let P_{trip} denote the probability of tripping the breaker. The breaker occupies one of the following regions:

- **Non-Tripped.** P_{trip} is zero when $n_s < N_{min}$
- **Non-Deterministic.** P_{trip} is a non-decreasing function of n_s when $N_{min} \leq n_s < N_{max}$
- **Tripped.** P_{trip} is one when $n_s \geq N_{max}$

Note that N_{min} and N_{max} depend on the breaker's trip curve and the application's demand for power when sprinting. For Spark on chip multiprocessors, we find that the breaker does

not trip when less than 25% of the chips sprint and definitely trips when more than 75% of the chips sprint. In other words, $N_{\min} = 0.25N$ and $N_{\max} = 0.75N$. We consider circuit breakers that can be overloaded to 125–175% of rated current for a 150s sprint.^{18, 21}

Uninterruptible power supplies. When the breaker trips and resets, power distribution switches from the branch circuit to the uninterruptible power supply (UPS).⁷ The rack augments power delivery with batteries to complete sprints in progress. Lead acid batteries support discharge times of 5–120min, long enough to support the duration of a sprint. After completing sprints and resetting the breaker, servers resume computation on the branch circuit.

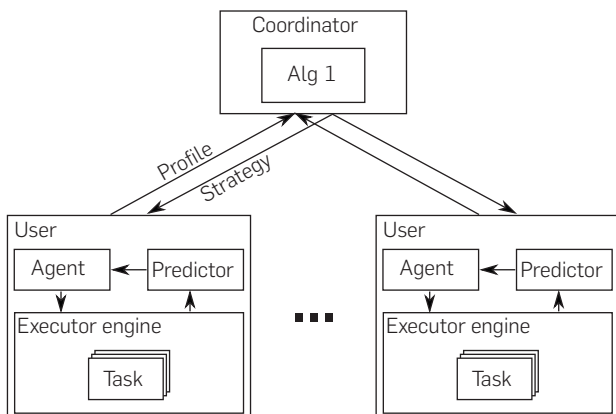
Servers are forbidden from sprinting again until UPS batteries are recharged. Sprints before recovery compromises server availability and increases vulnerability to power emergencies. Moreover, frequent discharges without recharges shorten battery life. The average recovery duration, denoted by $\Delta t_{\text{recovery}}$, depends on the UPS discharge depth and recharging time. A battery can be recharged to 85% capacity in 8–10 $\times$ the discharge time, which corresponds to 8–10 $\times$ the sprint duration.

2.2 Management architecture

Figure 3 illustrates the management framework for a rack of sprinting chip multiprocessors. The framework supports policies that pursue the performance of sprints while avoiding system instability. Unmanaged and excessive sprints may trip breakers, trigger emergencies, and degrade performance at scale. The framework achieves its objectives with strategic agents and coarse-grained coordination.

Users and agents. Each user deploys three run-time components: executor, agent, and predictor. Executors provide clean abstractions, encapsulating applications that could employ different software frameworks.¹⁰ The executor supports task-parallel computation by dividing an application into tasks, constructing a task dependence graph, and scheduling tasks dynamically based on available resources. Task scheduling is particularly important as it increases

Figure 3. Users deploy task executors and agents that decide when to sprint. Agents send performance profiles to a coordinator and receives optimized sprinting strategies.



parallelism when sprinting powers-on cores and tolerates faults when cooling and recovery powers-off cores.

Agents are strategic and selfish entities that act on users' behalf. They decide whether to sprint by continuously analyzing fine-grained application phases. Because sprints are followed by cooling and recovery, an agent sprints judiciously and targets application phases that benefit most from extra capability. Agents use predictors that estimate utility from sprinting based on software profiles and hardware counters. Each agent represents a user and her application on a chip multiprocessor.

Coordination. The coordinator collects profiles from all agents and assigns tailored sprinting strategies to each agent. The coordinator interfaces with strategic agents who may attempt to manipulate system outcomes by misreporting profiles or deviating from assigned strategies. Fortunately, our game-theoretic mechanism guards against such behavior.

First, agents will truthfully report their performance profiles. In large systems, game theory provides incentive compatibility, which means that agents cannot improve their utility by misreporting their preferences. An agent who misreports her profile has little influence on conditions in a large system. Not only does she fail to affect others, an agent who misreports suffers degraded performance as the coordinator assigns her a poorly suited strategy based on inaccurate profiles.

Second, agents will implement their assigned strategies because the coordinator optimizes those strategies to produce an equilibrium. In equilibrium, every agent implements her strategy and no agent benefits by deviating from it. An equilibrium has compelling implications for management overheads. If each agent knows that every other agent is playing her assigned strategy, she will do the same without further communication with the coordinator. Global communication between agents and the coordinator is infrequent and occurs only when system profiles change. In effect, an equilibrium permits the distributed enforcement of sprinting policies.

Equilibria are especially compelling when compared to the centralized enforcement of coordinated policies, which poses several challenges. First, centralized enforcement requires frequent and global communication as each agent decides whether to sprint by querying the coordinator at the start of each epoch. The length of an epoch is short and corresponds to sprint duration. Moreover, without equilibria, agents with kernel privileges could ignore prescribed policies, sprint at will, and cause power emergencies that harm all agents.

3. THE SPRINTING GAME

We design a sprinting game to govern power supply and manage system dynamics. The game divides time into epochs and asks agents to play repeatedly. Agents represent chip multiprocessors that share power. Each agent chooses to sprint independently, pursuing benefits in the current epoch and estimating repercussions in future epochs. An agent's utility from sprinting varies across epochs according to her application's phases. Multiple agents can sprint

simultaneously, but they risk tripping the circuit breaker and triggering power emergencies that harm global performance.

The game considers N agents who run task-parallel applications on N chip multiprocessors. Each agent computes in either normal or sprinting mode. The normal mode uses a fraction of the cores at low frequency whereas sprints use all cores at high frequency. Sprints rely on the executor to increase task parallelism and exploit extra cores. In this article, we consider three cores at 1.2GHz in normal mode and twelve cores at 2.7GHz in a sprint.

In any given epoch, an agent occupies one of three states—active (A), chip cooling (C), and rack recovery (R)—according to her actions and those of others in the rack. An agent's state describes whether she can sprint, and describes how cooling and recovery impose constraints on her actions.

Active (A) – Agent can safely sprint. An agent in the active state operates her chip in normal mode by default. The agent may decide to sprint by comparing benefits in the current epoch against benefits from deferring the sprint to a future epoch. If the agent sprints, her state in the next epoch is cooling.

Chip cooling (C) – Agent cannot sprint. After a sprint, an agent remains in the cooling state until excess heat has been dissipated. Cooling requires a number of epochs Δt_{cool} , which depends on the chip's thermal package. An agent in the cooling state stays in this state with probability p_c and returns to the active state with probability $1 - p_c$. Probability p_c is defined so that $1/(1 - p_c) = \Delta t_{\text{cool}}$.

Rack recovery (R) – Agent cannot sprint. When multiple chips sprint simultaneously, total current draw may trip the circuit breaker, trigger a power emergency, and require supplemental current from batteries. After an emergency, all agents remain in the recovery state until batteries recharge. Recovery requires a number of epochs $\Delta t_{\text{recover}}$, which depends on the power supply and battery capacity. Agents in the recovery state stay in this state with probability p_r and return to the active state with probability $1 - p_r$. Probability p_r is defined so that $1/(1 - p_r) = \Delta t_{\text{recover}}$.

4. GAME DYNAMICS AND STRATEGIES

Strategic agents decide between sprinting or not to maximize utilities. Sophisticated strategies produce several desirable outcomes. Agents sprint during the epochs that benefit most from additional cores and higher frequencies. Moreover, agents consider other agents' strategies because the probability of triggering a power emergency and entering the recovery state increases with the number of sprinters.

We analyze the game's dynamics to optimize each agent's strategy for her performance. A comprehensive approach to optimizing strategies considers each agent—her state, utility, and history—to determine whether sprinting maximizes her performance given her competitor's strategies and system state. In practice, however, this optimization is intractable for hundreds or thousands of agents.

4.1 Mean field equilibrium

The mean field equilibrium (MFE) is an approximation

method when analyzing individual agents in a large system is intractable.¹ First, we define key probability distributions on population behavior. Second, we optimize each agent's strategy in response to the population rather than individual competitors. Third, we find an equilibrium in which no agent can perform better by deviating from her optimal strategy. Thus, we reason about the population and neglect individual agents because any one agent has little impact on overall behavior in a large system.

The mean field analysis for the sprinting game focuses on the sprint distribution, which characterizes the number of agents who sprint when the system is not in recovery. In equilibrium, the sprint distribution is stationary and does not change across epochs. In any given epoch, some agents complete a sprint and enter the cooling state while others leave the cooling state and begin a sprint. Yet the number of agents who sprint is unchanged in expectation.

The stationary distribution for the number of sprinters translates into stationary distributions for the rack's current draw and the probability of tripping the circuit breaker. Given the tripping probability, which concisely describes population dynamics, an agent can formulate her best response and optimize her sprinting strategy to maximize performance. We find an equilibrium by specifying an initial value for the tripping probability and iterating.

- **Optimize sprint strategy (§4.2).** Given the probability of tripping the breaker P_{trip} , each agent optimizes her sprinting strategy to maximize her performance. She sprints if performance gains from doing so exceed some threshold. Optimizing her strategy means setting her threshold u_r .
- **Characterize sprint distribution (§4.3).** Given that each agent sprints according to her threshold u_r , the game characterizes population behavior. It estimates the expected number of sprinters n_s , calculates their demand for power, and updates the probability of tripping the breaker P'_{trip} .
- **Check for equilibrium.** The game is in equilibrium if $P'_{\text{trip}} = P_{\text{trip}}$. Otherwise, iterate with the new probability of tripping the breaker.

4.2 Optimizing the sprint strategy

Sprinting defines a repeated game in which an agent acts in the current epoch and encounters consequences of that action in future epochs. An agent optimizes her sprinting strategy accounting for the probability of tripping the circuit breaker P_{trip} , her utility from sprinting u , and her state. To decide whether to sprint, each agent optimizes the following Bellman equation.

$$V(u, A) = \max\{V_s(u, A), V_{\neg s}(u, A)\} \quad (1)$$

The equation quantifies value when an agent acts optimally in every epoch. V_s and $V_{\neg s}$ are the expected values from sprinting and not sprinting, respectively. If $V_s(u, A) > V_{\neg s}(u, A)$, then sprinting is optimal. The game solves the Bellman equation and identifies actions that maximize value with

dynamic programming.

Value in active state. An action's value depends on benefits in the current epoch plus the discounted value from future epochs. Suppose an agent in the active state decides to sprint. Her value from sprinting is her immediate utility u plus her discounted future utility. When she sprints, future utility is calculated for the cooling state $V(C)$ or the recovery state $V(R)$ when her sprint trips the breaker.

$$V_s(u, A) = u + \delta[V(C)(1 - P_{\text{trip}}) + V(R)P_{\text{trip}}] \quad (2)$$

However, an agent who does not sprint will remain in the active state unless other sprinting agents trip the circuit breaker and require recovery.

$$V_{-s}(u, A) = \delta[V(A)(1 - P_{\text{trip}}) + V(R)P_{\text{trip}}] \quad (3)$$

$V(A)$ denotes an agent's expected value from being in the active state. The game profiles an application and its time-varying computational phases to obtain a density function $f(u)$, which characterizes how often an agent derives utility u from sprinting. With this density, the game estimates expected value.

$$V(A) = \int V(u, A) f(u) du \quad (4)$$

Value in cooling and recovery states. An active agent transitions into cooling and recovery states when she and/or others sprint.

$$V(C) = \delta[V(C)p_c + V(A)(1 - p_c)](1 - P_{\text{trip}}) + \delta V(R)P_{\text{trip}} \quad (5)$$

$$V(R) = \delta[V(R)p_r + V(A)(1 - p_r)] \quad (6)$$

Parameters p_c and p_r are technology-specific probabilities of an agent in cooling and recovery states staying in those states. The game tunes these parameters to reflect the time required for chip cooling after a sprint and for rack recovery after a power emergency.

Threshold strategy. An agent should sprint if her utility from doing so is greater than not. Equation (7), which follows from Equations (2) and (3), states that an agent should sprint if her utility u is greater than her optimal threshold for sprinting u_T . Applying this strategy in every epoch maximizes expected value across time in the repeated game.

$$V_s(u, A) > V_{-s}(u, A) \\ u > \underbrace{\delta(V(A) - V(C))}_{u_T} (1 - P_{\text{trip}}) \quad (7)$$

4.3 Characterizing the sprint distribution

Given threshold u_T , an agent estimates the probability that she sprints, p_s , in a given epoch.

$$p_s = \int_{u_T}^{u_{\max}} f(u) du \quad (8)$$

The probabilities of sprinting (p_s) and cooling (p_c) define a

Markov chain that describes each agent's behavior. As agents play their strategies, the Markov chain converges to a stationary distribution in which each agent is active with probability p_A . Given N agents, the expected number of sprinters is

$$n_s = p_s \times p_A \times N \quad (9)$$

Given the expected number of sprinters, the game updates the probability of tripping the breaker according to its trip curve (e.g., Figure 2).

$$P_{\text{trip}} = \begin{cases} 0 & \text{if } n_s < N_{\min} \\ \frac{n_s - N_{\min}}{N_{\max} - N_{\min}} & \text{if } N_{\min} \leq n_s \leq N_{\max} \\ 1 & \text{if } n_s > N_{\max} \end{cases} \quad (10)$$

P_{trip} may change u_T and n_s , which may produce a new P'_{trip} . If $P_{\text{trip}} = P'_{\text{trip}}$, then agents are playing optimized strategies that produce an equilibrium.

4.4 Finding the equilibrium

When the game begins, agents make initial assumptions about population behavior and the probability of tripping the breaker. Agents optimize their strategies in response to population behavior. Strategies produce sprints that affect the probability of tripping the breaker. Over time, population behavior and agent strategies converge to a stationary distribution. The game is in equilibrium if the following conditions hold.

- Given tripping probability P_{trip} , the sprinting strategy dictated by threshold u_T is optimal and solves the Bellman equation in Equations (1)–(3).
- Given sprinting strategy u_T , the probability of tripping the circuit breaker is P_{trip} and is calculated by Equations (8)–(10).

In equilibrium, every agent plays her optimal strategy and no agent benefits when deviating from her strategy. In practice, the coordinator in the management framework finds and maintains an equilibrium with a mix of offline analysis and online play.

Offline analysis. Agents sample epochs and measure utility from sprinting to produce a density function $f(u)$, which characterizes how often an agent sees utility u from sprinting. The coordinator collects agents' density functions, analyzes population dynamics, and tailors sprinting strategies for each agent. Finally, the coordinator assigns optimized strategies to support online sprinting decisions.

Algorithm 1 describes the coordinator's offline analysis. It initializes the probability of tripping the breaker. Then, it iteratively analyzes population dynamics to find an equilibrium. Each iteration proceeds in three steps. First, the coordinator optimizes sprinting threshold u_T by solving the dynamic program defined in Equations (1)–(7). Second, it estimates the number of sprinters according to Equation (9). Finally, it updates the probability of tripping the breaker according to Equation (10). The algorithm terminates when thresholds, number of sprinters, and tripping probability

Algorithm 1: Optimizing the Sprint Strategy

input : Density for sprinting utilities ($f(u)$)
output: Optimal sprinting threshold (u_t)

```
j ← 1
Plstrip0 ← 1
while Ptripj not converged do
    urj ← DP solution for Equations (1)–(7) with Ptripj
    psj ← Equation (8) with f(u), urj
    nsj ← Equation (9) with MC solution and Psj
    Ptripj+1 ← Equation (10)
    j ← j + 1
end
```

are stationary.

The analysis runs periodically to update sprinting strategies and the tripping probability as application mix and system conditions evolve. The analysis does not affect an application's critical path as agents use updated strategies when they become available but need not wait for them. On an Intel® Core™ i5 processor with 4GB of memory, the analysis completes in less than 10s, on average.

Online play. An agent decides whether to sprint at the start of each epoch by estimating a sprint's utility and comparing it against her threshold. Estimation could be implemented in several ways. An agent could use the first few seconds of an epoch to profile her normal and sprinting performance. Alternatively, an agent could use heuristics to estimate utility from additional cores and higher clock rates. For example, task queue occupancy and cache misses are associated with a sprint's impact on task parallelism and instruction throughput, respectively. Comparisons with a threshold are trivial.

5. EXPERIMENTAL METHODOLOGY

Server measurements. The agent and its application are pinned to a chip multiprocessor, an Intel® Xeon® E5-2697 v2. In normal mode, the agent uses three 1.2GHz cores. In sprinting mode, the agent uses twelve 2.7GHz cores. We turn cores on and off with Linux `sysfs`. In principle, sprinting represents any mechanism that performs better but consumes more power.

We evaluate Apache Spark workloads. The Spark runtime engine dynamically schedules tasks to use available cores and maximize parallelism, adapting as sprints cause the number of available cores to vary across epochs. We profile workloads by modifying Spark (v1.3.1) to log the IDs of jobs, stages, and tasks as they complete. We profile system and power temperature using the Intel® Performance Counter Monitor 2.8.

We measure workload performance in terms of tasks completed per second (TPS). The total number of tasks in a job is constant and independent of the available hardware resources such that TPS measures performance for a fixed amount of work. In our experiments, we trace TPS during application execution in normal and sprinting

modes and we estimate speedups by comparing the two traces, epoch by epoch. In a practical system, online profiling and heuristics would be required to estimate speedups.

Datacenter simulation. We simulate 1000 users and evaluate their performance in the sprinting game. The simulator uses server traces and models system dynamics as agents sprint, cool, and recover. Simulations evaluate homogeneous agents who arrive randomly and launch the same type of Spark application; randomized arrivals cause application phases to overlap in diverse ways. Diverse phase behavior exercises the sprinting game as agents optimize strategies in response to varied competitors'.

Table 1 summarizes technology and system parameters. Parameters $N_{\min}$ and $N_{\max}$ are set by the circuit breaker's tripping curve. Parameters p_c and p_r are set by the chip's cooling mechanism and the system's batteries. These probabilities decrease as cooling efficiency and recharge speed increase.

6. EVALUATION

We evaluate the sprinting game and its equilibrium threshold against several alternatives that represent broader perspectives on power management. First, greedy heuristics focus on the present and neglect the future.²¹ Second, control-theoretic heuristics are reactive rather than proactive.² Third, centralized heuristics focus on the system and neglect individual users. Unlike these approaches, the sprinting game anticipates the future and models strategic agents in a shared system.

Greedy (G) permits agents to sprint as long as the chip is not cooling and the rack is not recovering. This mechanism may frequently trip the breaker and require rack recovery. Greedy produces a poor equilibrium—knowing that everyone is sprinting, an agent's best response is to sprint as well.

Exponential Backoff (E-B) throttles the frequency at which agents sprint. An agent sprints greedily until the breaker trips. After the t -th trip, agents wait for some number of epochs drawn randomly from $[0, 2^t - 1]$ before sprinting again. The waiting interval contracts by half if the breaker has not been tripped in the past 100 epochs.

Cooperative Threshold (C-T) assigns each agent the globally optimal sprinting threshold. The coordinator identifies and enforces thresholds that maximize system performance. Although these thresholds provide an upper bound on performance, they do not produce an equilibrium because thresholds do not reflect agents' best responses to system dynamics.

Equilibrium Threshold (E-T) assigns each agent her optimal threshold from the sprinting game. The coordinator collects performance profiles and finds thresholds that

Table 1. Experimental Parameters.

Description	Symbol	Value
Min # sprinters	$N_{\min}$	250
Max # sprinters	$N_{\max}$	750
Prob. of staying in cooling	p_c	0.50
Prob. of staying in recovery	p_r	0.88
Discount factor	δ	0.99

reflect agents' best responses to system dynamics. These thresholds produce an equilibrium and agents cannot benefit by deviating from their assigned strategy.

6.1 Sprinting behavior

Figure 4 compares sprinting policies and resulting system dynamics as 1000 instances of *Decision Tree*, a representative application, computes across over time. Sprinting policies determine how often agents sprint and whether sprints trigger emergencies. Ideally, policies would permit agents to sprint up until they trip the circuit breaker. In this example, 250 of the 1000 agents can sprint before triggering a power emergency.

Greedy heuristics are aggressive and inefficient. A sprint in the present precludes a sprint in the near future, harming subsequent tasks that could have benefited more from the sprint. Moreover, frequent sprints risk power emergencies and require rack-level recovery. G produces an unstable system, oscillating between full-system sprints that trigger emergencies and idle recovery that harms performance.

Control-theoretic approaches are more conservative, throttling sprints in response to power emergencies. E-B adaptively responds to feedback, producing a more stable system with fewer sprints and emergencies. Indeed, E-B may be too conservative, throttling sprints beyond what is necessary to avoid tripping the circuit breaker. The number of sprinters is consistently lower than N_{min} , which is safe but leaves sprinting opportunities unexploited. In neither G nor E-B do agents sprint to full advantage.

In contrast, the computational sprinting game performs

well by embracing agents' strategies. E-T produces an equilibrium in which agents play their optimal strategies and converge to a stationary distribution. In equilibrium, the number of sprinters is just slightly above N_{min} , the number that causes a breaker to transition from the non-tripped region to the tolerance band. After emergency and recovery, the system quickly returns to equilibrium.

Figure 5 shows the percentage of time an agent spends in each state. E-T and C-T sprints are timely as strategic agents sprint only when estimated benefits exceed an optimized threshold. A sprint in E-T or C-T contributes more to performance than one in G or E-B. Moreover, G and E-B ignore the consequences of a sprint. With G, an agent spends more than 50% of its time in recovery, waiting for batteries to recharge after an emergency. With E-B, an agent spends nearly 40% of its time in active mode but not sprinting.

6.2 Sprinting performance

Figure 6 shows task throughput under varied policies. The sprinting game outperforms greedy heuristics and is competitive with globally optimized heuristics. Rather than sprinting greedily, E-T uses equilibrium thresholds to select more profitable epochs for sprinting. E-T outperforms G and E-B by up to 6.8× and 4.8×, respectively. Agents who use their own strategies to play the game competitively produce outcomes that rival expensive cooperation. E-T's task throughput is 90% that of C-T's for most applications.

Linear Regression and *Correlation* are outliers, achieving only 36% and 65% of cooperative performance. For these applications, E-T performs as badly as G and E-B because the applications' performance profiles exhibit little variance

Figure 4. Sprinting behavior for a representative application, *Decision Tree*. Black line denotes number of sprinters. Gray line denotes the point at which sprinters risk a power emergency, N_{min} .

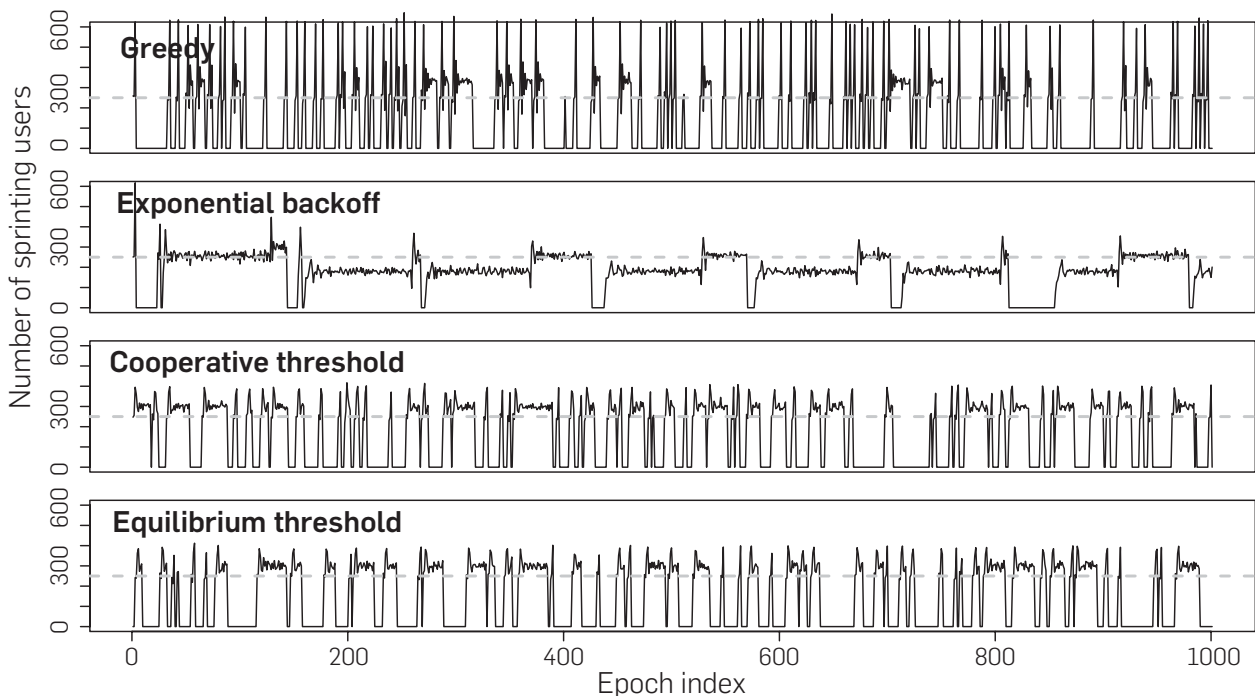


Figure 5. Percentage of time spent in agent states for a representative application, Decision Tree.

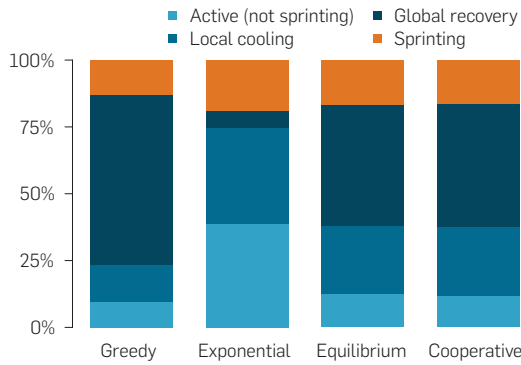
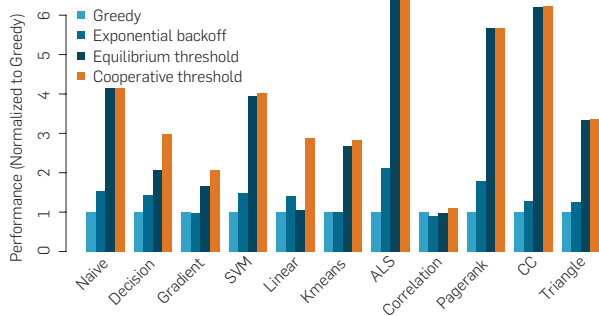


Figure 6. Performance, measured in tasks per second and normalized against greedy, for a single application type.



and all epochs benefit similarly from sprinting. When an agent cannot distinguish between epochs, she sets a low threshold and sprints for every epoch. In effect, for such applications, E-T produces a greedy equilibrium.

6.3 Sprinting strategies

Figure 7 uses density plots for two representative applications, *Linear Regression* and *PageRank*, to show how often and how much their tasks benefit from sprinting. *Linear Regression* presents a narrower distribution and performance gains from sprinting vary in a band between 3× and 5×. In contrast, *PageRank*'s performance gains can often exceed 10×.

The coordinator uses density plots to optimize threshold strategies. *Linear Regression*'s strategy is aggressive and uses a low threshold that often induces sprints. This strategy arises from its relatively low variance in performance gains. If sprinting's benefits are indistinguishable across tasks and epochs, an agent sprints indiscriminately and at every opportunity. *PageRank*'s strategy is more nuanced and uses a high threshold, which cuts her bimodal distribution and implements judicious sprinting. She sprints for tasks and epochs that benefit most (*i.e.*, those that see performance gains greater than 10×).

Figure 8 illustrates diversity in agents' strategies by reporting their propensities to sprint. *Linear Regression* and *Correlation*'s narrow density functions and low thresholds cause these applications to sprint at every opportunity. The

Figure 7. Probability density for sprinting speedups.

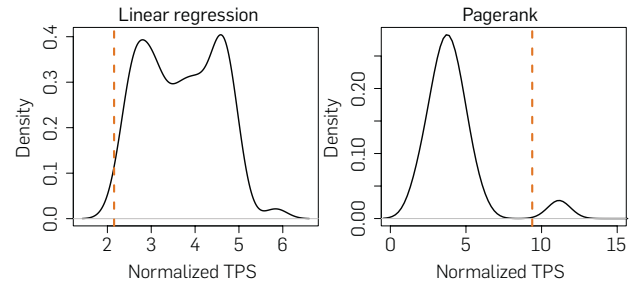
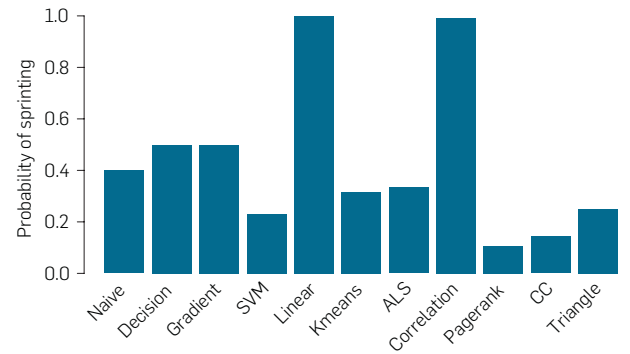


Figure 8. Probability of sprinting.



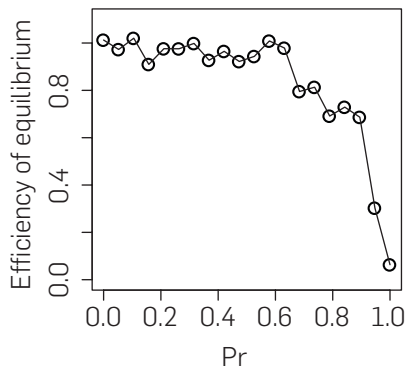
majority of applications, however, resemble *PageRank* with higher thresholds and judicious sprints.

6.4 Equilibrium versus cooperation

Equilibrium thresholds are robust to strategic behavior and perform well, but cooperative thresholds can perform even better. The sprinting game's equilibrium delivers 90% of the performance from cooperation because the penalties from non-cooperative behavior are low. Figure 9 shows how efficiency falls as recovery from power emergencies become increasingly expensive. Recall that p_r is the probability an agent in recovery stays in that state.

The sprinting game fails when an emergency requires indefinite recovery and p_r is one. This game has no equilibrium that avoids tripping the breaker and triggering indefinite recovery. If a strategic agent were to observe system dynamics that avoid tripping the breaker, which means P_{trip} is zero, she would realize that other agents have set high thresholds to avoid sprints. Her best response would be lowering her threshold and sprinting more often. Others would behave similarly and drive P_{trip} higher. In equilibrium, P_{trip} would rise above zero and agents would eventually trip the breaker, putting the system into indefinite recovery. Thus, selfish agents would produce inefficient equilibria—the Prisoner's Dilemma in which each agent's best response performs worse than a cooperative one.

The Folk theorem guides agents to a more efficient equilibrium by punishing agents whose responses harm the system. The coordinator would assign agents the best cooperative thresholds to maximize system performance from sprinting. When an agent deviates, she is punished such that

Figure 9. Efficiency of equilibrium thresholds.

performance lost exceeds performance gained. In our example, punishments would allow the system to escape inefficient equilibria as agents are compelled to increase their thresholds and ensure P_{trip} remains zero. The coordinator could monitor sprints, detect deviations from assigned strategies, and forbid agents who deviate from ever sprinting again. Note that threat of punishment is sufficient to shape the equilibrium.

7. CONCLUSION

Economics and game theory have proven effective in data-center power and resource management. Game-theoretic notions of fairness can incentivize strategic users when sharing hardware.^{6,12,19,20} Markets and price theory can allocate and manage heterogeneous servers.^{8,9,17} Demand response models can handle power emergencies.^{3,11}

We link system architecture and algorithmic economics to decentralize the allocation of shared resources to strategic users. The computational sprinting game is a management architecture that governs how independent chip multiprocessors share a power supply. The approach generalizes beyond datacenters and is relevant to systems that are distributed, heterogeneous, and dynamic. The game's approach to sprinting applies to any mechanism that briefly accelerates performance using additional resources be they processor, memory, network, or power. The game's equilibrium highlights a path to scalable management because mean field analysis provides tractability when the number of system components is large. However, finding the equilibrium requires statistical distributions of agent behaviors and further research is needed to reduce offline profiling costs and accelerate online utility prediction.

Acknowledgments

This work is supported by National Science Foundation grants CCF-1149252, CCF-1337215, SHF-1527610, and AF-1408784. This work is also supported by STARnet, a SRC program, sponsored by MARCO and DARPA. 

References

- Adlakha, S., Johari, R. Mean field equilibrium in dynamic games with strategic complementarities. *Oper. Res.* 61, 4 (2013), 971–989.
- Brooks, D. Martonosi, M. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the 7th IEEE International Symposium on High Performance Computer Architecture (HPCA)* (Monterrey, Nuevo Leon, Mexico, 2001), 171–182.

- Chase, J.S., Anderson, D.C., Thakar, P.N., Vahdat, A.M., Doyle, R.P. Managing energy and server resources in hosting centers. In *Proceedings of the 18th Symposium on Operating Systems Principles (SOSP)* (Banff, Alberta, Canada, 2001), 103–116.
- Fan, X., Weber, W.-D., Barroso, L.A. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA)* (San Diego, CA, USA, 2007), 13–23.
- Fu, X., Wang, X., Lefurgy, C. How much power oversubscription is safe and allowed in data centers. In *Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC)* (Karlsruhe, Germany, 2011), 21–30.
- Ghods, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., Stoica, I. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI)* (Boston, MA, USA, 2011), 323–336.
- Govindan, S., Sivasubramanian, A., Urgaonkar, B. Benefits and limitations of tapping into stored energy for datacenters. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)* (San Jose, CA, USA, 2011), 341–351.
- Guevara, M., Lubin, B., Lee, B.C. Navigating heterogeneous processors with market mechanisms. In *Proceedings of the 19th IEEE International Symposium on High Performance Computer Architecture (HPCA)* (Shenzhen, China, 2013), 95–106.
- Guevara, M., Lubin, B., Lee, B.C. Strategies for anticipating risk in heterogeneous system design. In *Proceedings of the 20th IEEE International Symposium on High Performance Computer Architecture (HPCA)* (Orlando, FL, USA, 2014), 154–164.
- Hindman, B., Konwinski, A., Zaharia, M., Ghods, A., Joseph, A.D., Katz, R., Shenker, S., Stoica, I. Mesos: A platform for fine-grained resource sharing in the data center. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI)* (Boston, MA, USA, 2011), 295–308.
- Liu, Z., Wierman, A., Chen, Y., Razon, B., Chen, N. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Perform. Eval.* 70, 10 (2013), 770–791.
- Lull, Q., Fan, S., Zahedi, S.M., Lee, B.C. Cooper: Task colocation with cooperative games. In *Proceedings of the 23rd IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (Austin, TX, USA, 2017), 421–432.
- Raghavan, A., Emurian, L., Shao, L., Papaefthymiou, M., Pipe, K.P., Wenisch, T.F., Martin, M.M. Computational sprinting on a hardware/software testbed. In *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (Houston, TX, USA, 2013), 155–166.
- Raghavan, A., Luo, Y., Chandawalla, A., Papaefthymiou, M., Pipe, K.P., Wenisch, T.F., Martin, M.M. Computational sprinting. In *Proceedings of the 18th IEEE International Symposium on High Performance Computer Architecture (HPCA)* (New Orleans, LA, USA, 2012), 1–12.
- Shao, L., Raghavan, A., Emurian, L., Papaefthymiou, M.C., Wenisch, T.F., Martin, M.M., Pipe, K.P. On-chip phase change heat sinks designed for computational sprinting. In *Proceedings of the 30th Annual Semiconductor Thermal Measurement and Management Symposium* (San Jose, CA, USA, 2014), 29–34.
- Skach, M., Arora, M., Hsu, C.-H., Li, Q., Tullsen, D., Tang, L., Mars, J. Thermal time shifting: Leveraging phase change materials to reduce cooling costs in warehouse-scale computers. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA)* (Portland, OR, USA, 2015), 439–449.
- Somu, Muthukaruppan, T., Pathania, A., Mitra, T. Price theory based power management for heterogeneous multi-cores. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (Salt Lake City, UT, USA, 2014), 161–176.
- Wang, X., Chen, M., Lefurgy, C., Keller, T.W. Ship: A scalable hierarchical power control architecture for large-scale data centers. *IEEE Trans. Parallel Distrib. Syst.* 23, 1 (2012), 168–176.
- Zahedi, S.M., Lee, B.C. Sharing incentives and fair division for multiprocessors. *IEEE Micro* 35, 3 (2015), 92–100.
- Zahedi, S.M., Lull, Q., Lee, B.C. Amdahl's Law in the datacenter era: A market for fair processor allocation. In *Proceedings of the 24th IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (Vienna, Austria, 2018).
- Zheng, W., Wang, X. Data center sprinting: Enabling computational sprinting at the data center level. In *Proceedings of the 35th International Conference on Distributed Computing Systems (ICDCS)* (Columbus, OH, USA, 2015), 175–184.

Songchun Fan[†] (songchun.fan@duke.edu), Duke University, California, USA.

Seyed Majid Zahedi[†] and Benjamin C. Lee ([seyedmajid.zahedi, benjamin.c.lee]@duke.edu), Duke University, Durham, NC, USA.

Technical Perspective

To Do or Not to Do: Extending SQL with Integer Linear Programming?

By Surajit Chaudhuri

RELATIONAL QUERY LANGUAGES have enabled the programmer to express queries using a logical model of data without any knowledge of the underlying physical structures. To help applications realize the benefits of such declarative querying of data fully, there has been much work along the following three dimensions:

a) Application programming interfaces (for example, ODBC, JDBC) have been developed to enable applications connect to and access data in a relational database system. However, when connecting using these interfaces, the application programmer must still handle two different programming models. Language integrated query (LINQ) is an elegant example of integration where query expressions are introduced as a first-class citizen in the programming languages to avoid the above problem, and a mapping tool (LINQ to SQL) translates language-integrated queries into SQL for the database backend. More recently, databases have been exposing a REST API for the ease of mobile and web applications.

b) Modern database systems provide extensibility so that applications programmers are not limited to using the built-in types and functions in SQL. All major database systems support user-defined functions that may be used in selection, aggregation, or table expressions in a query. These user-defined functions (potentially with parameters) are written in native SQL or programming languages for which the database server provides runtime support. Such extensibility mechanisms have been used by database systems to add support for data types such as geospatial.

c) The SQL standard has added new operators and constructs to make declarative querying in relational languages more convenient or expressive, for example, recursion, window functions, grouping sets, within group.

Despite the advances that have already taken place along these three dimensions, there continues to be proposals from time to time to further enrich functionality of relational databases to support important classes of applications.

The following paper by Brucato et al. is one such proposal for making relational databases do more. It makes a case for marrying the well-established paradigms of constrained optimization (specifically, ILP or integer linear programming) and traditional SQL querying.

The challenge of augmenting query languages with the power of specifying constraints has been well studied in the literature, both in the context of database querying as well as logic programming. Earlier research has studied schemes for adding constraints on individual rows (beyond simple selection) as well as *aggregate constraints* that the set of answer rows to a query must satisfy collectively. Introduction of aggregate constraints makes query evaluation especially challenging. The paper demonstrates that when you add an optimization criterion to a query language with aggregate constraints to choose among qualifying sets of answer sets, the query evaluation can be accomplished by a combination of the relational query execution engine and an off-the-shelf ILP solver.

The authors explain how such queries may be specified declaratively (referred to as *package queries*). These package queries are evaluated by first executing the traditional relational part of the query and then mapping the constraint satisfaction and objective criterion as an instance of the ILP problem. The extensibility features of the database system, as explained in (b), may be used to add such an ILP solver to the database systems just like the support for user defined functions written in programming languages (for example,

Java or C#) other than the native SQL. The paper also addresses techniques for solving large ILP problems using offline partitioning and approximation techniques to break down the global ILP instance into smaller ILP sub-problems. However, while their offline partitioning is a good physical design optimization to have in the repertoire, its applicability also depends on the characteristics of the production workload on the system.

Adding any new functionality to a query language as rich as SQL has complex trade-offs. Issues that influence such a decision are ease of specification of the new functionality in the query, execution efficiency of the enriched query system, data movement, and increased software complexity of the database systems. Moreover, even when a new functionality is incorporated, there is a question of whether the core SQL should be enriched like other examples in (c), as suggested by this paper, or if the functionality should be incorporated strictly via the extensibility mechanisms. Specifically, in this case, an alternative to extending SQL will be to have a separate domain-specific language (potentially using a syntax like that of package queries), interpreted by the ILP solver runtime, and integrated with the database system.

If you are interested in the topic of constraint specification and optimization over data stored in databases, this paper is sure to interest you. Also, it is worth a read for anyone who wants to consider adding extensions to SQL to ease application tasks, as the authors illustrate the key dimensions of what it takes to add any new functionality to relational querying: language extension, changes to the query execution engine, and techniques to cope with scale. 

Surajit Chaudhuri is a Distinguished Scientist at Microsoft Research, Redmond, WA, USA.

Copyright held by author.

Scalable Computation of High-Order Optimization Queries

By Matteo Brucato, Azza Abouzied, and Alexandra Meliou

Abstract

Constrained optimization problems are at the heart of significant applications in a broad range of domains, including finance, transportation, manufacturing, and healthcare. Modeling and solving these problems has relied on application-specific solutions, which are often complex, error-prone, and do not generalize. Our goal is to create a domain-independent, declarative approach, supported and powered by the system where the data relevant to these problems typically resides: the database. We present a complete system that supports *package queries*, a new query model that extends traditional database queries to handle complex constraints and preferences over answer sets, allowing the declarative specification and efficient evaluation of a significant class of constrained optimization problems—integer linear programs (ILP)—within a database.

1. INTRODUCTION

Traditional database queries follow a simple model: they define constraints, in the form of selection predicates, that each tuple in the result must satisfy. This model is computationally efficient, as the database system can evaluate each tuple individually to determine whether it satisfies the query conditions. However, many practical, real-world problems require a collection of result tuples to satisfy constraints collectively, rather than individually.

EXAMPLE 1 (MEAL PLANNER). *A dietitian needs to design a daily meal plan for a patient. She wants a set of three gluten-free meals, between 2000 and 2500 calories in total, and with a low total intake of saturated fats.*

Similar scenarios, requiring complex, high-order constraints arise frequently, and in many practical settings. A broad set of domains have applications that boil down to modeling and solving constrained optimization problems, for example, coordinating fleet and crew assignments in airline scheduling to reduce delays and costs,¹⁹ managing delinquent consumer credit to minimize losses,¹⁴ optimizing organ transplant allocation and acceptance,¹ and planning of cancer radiotherapy treatments.^{20,21} A significant class of constrained optimization problems are *integer linear programs* (ILP). ILP solutions alone account for billions in US dollars of projected benefits within each of these and other industry sectors.⁷

Modeling and solving these problems has relied on application-specific solutions,^{2, 9, 13, 17, 23, 18} which can often be complex and error-prone, and fail to generalize. Our goal is to create a domain-independent, declarative approach, supported and powered by the system where the data

relevant to these problems typically resides: the database. We present a complete system that supports *package queries*, a new query model that extends traditional database queries to handle complex constraints and preferences over answer sets, allowing the declarative specification and efficient evaluation of a significant class of constrained optimization problems—ILP—within a database. Package queries are defined over traditional relations, but return *packages*. A package is a collection of tuples that (a) individually satisfy *base predicates* (traditional selection predicates), and (b) collectively satisfy *global predicates* (package-specific predicates). Package queries are combinatorial in nature: the result of a package query is a (potentially infinite) set of packages, and an *objective criterion* can define a preference ranking among them.

Extending traditional database functionality to provide support for packages, rather than supporting packages at the application level, is justified by two reasons: First, the features of packages and the algorithms for constructing them are not unique to each application; therefore, the burden of package support should be lifted off application developers, and database systems should support package queries like traditional queries. Second, the data used to construct packages typically reside in a database system, and packages themselves are structured data objects that should naturally be stored in and manipulated by a database system.

Our work addresses *three important challenges*. The first challenge is to support *declarative* specification of packages. SQL enables the declarative specification of properties that result tuples should satisfy. In Example 1, it is easy to specify the exclusion of meals with gluten using a regular selection predicate in SQL. However, it is difficult to specify global constraints (e.g., total calories of a set of meals should be between 2000 and 2500 calories). Expressing such a query in SQL requires either complex self-joins that explode the size of the query, or recursion, which results in extremely complex queries that are hard to specify and optimize. Our goal is to maintain the declarative power of SQL, while extending its expressiveness to allow for the easy specification of packages.

The second challenge relates to the *evaluation* of package queries. Due to their combinatorial complexity, package queries are harder to evaluate than traditional database queries.¹⁰ Package queries are in fact as hard as ILP.⁵ Existing database technology is ineffective at

The original version of this paper is entitled "Scalable Package Queries in Relational Database Systems" and was published in the *Proceedings of the VLDB Endowment*, Vol. 9, No. 7 (2016), 576–587.

evaluating package queries, even if one were to express them in SQL. Figure 1 shows the performance of evaluating a package query expressed as a multi-way self-join query in traditional SQL. As the cardinality of the package increases, so does the number of joins, and the runtime quickly becomes prohibitive: In a small set of 100 tuples from the Sloan Digital Sky Survey (SDSS) dataset,²² SQL evaluation takes almost 24 hours to construct a package of 7 tuples. Our goal is to extend the database evaluation engine to take advantage of external tools, such as ILP solvers, which are more effective for combinatorial problems.

The third challenge pertains to query evaluation *performance* and *scaling* to large datasets. Integer programming solvers have two major limitations: they require the entire problem to fit in main memory, and they fail when the problem is too complex (e.g., too many variables and/or too many constraints). Our goal is to overcome these limitations through sophisticated evaluation methods that allow solvers to scale to large data sizes.

Our work addresses these challenges through the design of language and algorithmic support for the specification and evaluation of package queries. We present PaQL (Package Query Language), a declarative language that provides simple extensions to standard SQL to support constraints at the package level. PaQL is at least as expressive as ILP, which implies that evaluation of package queries is NP-hard.⁵ We present a fundamental evaluation strategy, **DIRECT**, that combines the capabilities of databases and constraint optimization solvers to derive solutions to package queries. The core of our approach is a set of translation rules that transform a package query to an ILP. This translation allows for the use of highly-optimized external solvers for the evaluation of package queries. We introduce an offline data partitioning strategy that allows package query evaluation to scale to large data sizes. The core of our evaluation strategy, **SKETCHREFINE**, lies in separating the package computation into multiple stages, each with small subproblems, which the solver can evaluate efficiently. In the first stage, the algorithm “sketches” an initial sample package from a set of representative tuples, while the subsequent stages “refine” the current package by solving an ILP within each partition. **SKETCHREFINE** offers strong approximation guarantees for the package results compared to **DIRECT**. We present an extensive experimental evaluation on real-world data that shows that our query evaluation method **SKETCHREFINE**: (1) is able to produce packages an

order of magnitude faster than the ILP solver used directly on the entire problem; (2) scales up to sizes that the solver cannot manage directly; (3) produces packages of very good quality in terms of objective value.

2. LANGUAGE SUPPORT FOR PACKAGES

Database systems do not natively support package queries. While there are ways to express package queries in SQL, these are cumbersome and inefficient.

Specifying packages with self-joins. In the limited case of packages with strict cardinality, that is, a fixed number of tuples, it is possible to express package queries using relational self-joins. The query of Example 1 requires three meals (a package with cardinality three) and can be expressed as a three-way self-join:

```
SELECT * FROM Recipes R1, Recipes R2, Recipes R3
WHERE   R1.pk < R2.pk AND R2.pk < R3.pk AND
        R1.gluten = 'free' AND R2.gluten = 'free' AND R3.gluten = 'free'
        AND R1.kcal + R2.kcal + R3.kcal BETWEEN 2.0 AND 2.5
ORDER BY R1.saturated_fat + R2.saturated_fat + R3.saturated_fat
```

Such a query is efficient only for constructing packages with very small cardinality: larger cardinality requires a larger number of self-joins, quickly rendering evaluation time prohibitive (Figure 1). The benefit of this specification is that the optimizer can use the traditional relational algebra operators and augment its decisions with package-specific strategies. However, this method does not apply for packages of unbounded cardinality.

Specifying packages using recursion. SQL can express package queries by generating and testing each possible subset of the input relation. This requires recursion to build a *powerset table*; checking each set in the powerset table for the query conditions will yield the result packages. This approach has three major drawbacks. First, it is not declarative, and the specification is tedious and complex. Second, it is not amenable to optimization in existing systems. Third, it is extremely inefficient to evaluate, because the powerset table generates an exponential number of candidates.

2.1. PaQL: The package query language

Our goal is to support declarative and intuitive package specification. In this section, we describe PaQL, a declarative query language that introduces simple extensions to SQL to define package semantics and package-level constraints. Figure 2 shows the general syntax of PaQL (left) and the specification for the query of Example 1 (right), which we use as a running example to demonstrate PaQL’s features. Square brackets enclose optional clauses and arguments, and a vertical bar separates syntax alternatives. In this specification, **repeat** is a non-negative integer; **w_expression** is a Boolean expression over tuple values (as in standard SQL) and can only contain references to **relation_name** and **relation_alias**; **st_expression** is a Boolean expression and **obj_expression** is an expression over aggregate functions or SQL subqueries with aggregate functions; both **st_expression** and **obj_expression** can

Figure 1. Traditional database technology is ineffective at package evaluation, and the runtime of a SQL formulation of a package query grows exponentially. In contrast, tools such as ILP solvers are more effective.

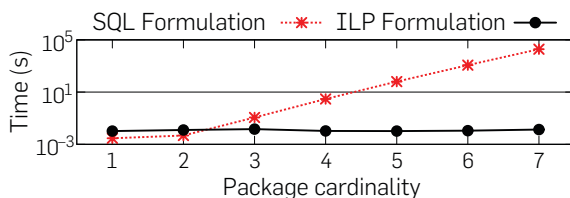


Figure 2. Specification of the PaQL syntax (left), and the PaQL query for Example 1 (right).

PaQL syntax specification	PaQL query for Example 1
SELECT PACKAGE (* column_name [,...]) [AS] package_name FROM relation_name [AS] relation_alias [REPEAT repeat] [,...] [WHERE w_expression] [SUCH THAT st_expression] [(MINIMIZE MAXIMIZE) obj_expression]	Q: SELECT PACKAGE (*) AS P FROM Recipes R REPEAT 0 WHERE R.gluten = 'free' SUCH THAT COUNT (P.*) = 3 AND SUM(P.kcal) BETWEEN 2.0 AND 2.5 MINIMIZE SUM(P.sat_fat)

only contain references to **package_name**, which specifies the name of the package result.

Basic package query. The new keyword **PACKAGE** differentiates PaQL from traditional SQL queries.

Q_1 : SELECT * FROM Recipes R Q_2 : SELECT **PACKAGE**(*) AS P FROM Recipes R

The semantics of Q_1 and Q_2 are fundamentally different: Q_1 is a traditional SQL query, with a unique, finite result set (the entire Recipes table), whereas there are infinitely many packages that satisfy the package query Q_2 : all possible *multisets* of tuples from the input relation. The result of a package query like Q_2 is a set of packages. Each package resembles a relational table containing a collection of tuples (with possible repetitions) from relation Recipes, and therefore a package result of Q_2 follows the schema of Recipes. Similar to SQL, the PaQL syntax allows the specification of the output schema in the SELECT clause. For example, **PACKAGE**(sat_fat, kcal) only returns the saturated fat and calorie attributes of the package.

Although semantically valid, a query like Q_2 would not occur in practice, as most application scenarios expect few, or even exactly one result. We proceed to describe the additional constraints in the example query Q (Figure 2) that restrict the number of package results.

Repetition constraints. The **REPEAT 0** statement in query Q from Figure 2 specifies that each tuple from the input relation Recipe can appear in a package result at most once (no repetitions are allowed). If this restriction is absent (as in query Q_2), the multiplicity of a tuple is unbounded. By allowing no repetitions, Q restricts the package space from infinite to 2^n , where n is the size of the input relation. Generalizing, **REPEAT ρ** allows a package to repeat tuples up to ρ times, resulting in $(2 + \rho)^n$ candidate packages.

Base and global predicates. A package query defines two types of predicates. A *base predicate*, defined in the WHERE clause, is equivalent to a selection predicate and can be evaluated with standard SQL: any tuple in the package needs to *individually* satisfy the base predicate. For example, query Q from Figure 2 specifies the base predicate: $R.gluten = \text{'free'}$. Since base predicates directly filter input tuples, they are specified over the input relation R . *Global predicates* are the core of package queries, and they appear in the new **SUCH THAT** clause. Global predicates are higher-order than base predicates: they cannot be evaluated on individual tuples, but on tuple collections. Since they describe package-level

constraints, they are specified over the package result P , for example, $COUNT(P.*) = 3$, which limits the query results to packages of exactly 3 tuples.

The global predicates in query Q abbreviate aggregates that are in reality SQL subqueries. For example, $COUNT(P.*) = 3$, abbreviates $(SELECT COUNT(*) FROM P) = 3$. Using subqueries, PaQL can express arbitrarily complex global constraints among aggregates over a package.

Objective clause. The objective clause specifies a ranking among candidate package results and appears with either the **MINIMIZE** or **MAXIMIZE** keyword. It is a condition on the package-level, and hence it is specified over the package result P , for example, **MINIMIZE SUM(P.sat_fat)**. Similar to global predicates, this form is a shorthand for **MINIMIZE (SELECT SUM(sat_fat) FROM P)**. A PaQL query with an objective clause returns a single result: the package that optimizes the value of the objective. The evaluation methods that we present in this work focus on such queries. In prior work,⁶ we described preliminary techniques for returning multiple packages in the absence of optimization objectives, but a thorough study of such methods is left to future work.

Expressiveness and complexity. PaQL can express general ILP, which means that evaluation of package queries is NP-complete.^{4,5} As a first step in package evaluation, we proceed to show how a PaQL query can be transformed into a linear program and solved using general ILP solvers.

3. ILP FORMULATION

In this section, we present an ILP formulation for package queries, which is at the core of our evaluation methods **DIRECT** and **SKETCHREFINE**. The results in this section are inspired by the translation rules employed by Tiresias¹⁵ to answer *how-to queries*.

3.1. PaQL to ILP translation

Let R indicate the input relation of the package query, $n = |R|$ be the number of tuples in R , $R.attr$ an attribute of R , P a package, f a linear aggregate function (such as **COUNT** and **SUM**), $\odot \in \{\leq, \geq\}$ a constraint inequality, and $v \in \mathbb{R}$ a constant. For each tuple t_i from R , $1 \leq i \leq n$, the ILP problem includes a nonnegative integer variable x_i , $x_i \geq 0$, indicating the number of times t_i is included in an answer package. We also use $\bar{x} = \langle x_1, x_2, \dots, x_n \rangle$ to denote the vector of all integer variables. A PaQL query is formulated as an ILP problem using the following translation rules.

Repetition constraint. The **REPEAT** keyword, expressible in the **FROM** clause, restricts the domain that the variables

can take on. Specifically, REPEAT ρ implies $0 \leq x_i \leq \rho + 1$.

Base predicate. Let β be a base predicate, for example, $R_{\text{gluten}} = \text{'free'}$, and R_β the relation containing tuples from R satisfying β . We encode β by setting $x_i = 0$ for every tuple $t_i \notin R_\beta$.

Global predicate. Each global predicate in the SUCH THAT clause takes the form $f(P) \odot v$. For each such predicate, we derive a linear function $f'(\bar{x})$ over the integer variables. A cardinality constraint $f(P) = \text{COUNT}(P.*)$ is translated into a linear function $f'(\bar{x}) = \sum_i x_i$. A summation constraint $f(P) = \text{SUM}(P.\text{attr})$ is translated into a linear function $f'(\bar{x}) = \sum_i (t_i.\text{attr})x_i$. Other nontrivial constraints and general Boolean expressions over the global predicates can be encoded into a linear program with the help of Boolean variables and linear transformation tricks found in the literature.³ We refer to the original version of this paper for further details.^{4,5}

Objective clause. We encode MAXIMIZE $f(P)$ as $\max f'(\bar{x})$, where $f'(\bar{x})$ is the encoding of $f(P)$. Similarly MINIMIZE $f(P)$ is encoded as $\min f'(\bar{x})$.

EXAMPLE 2 (ILP TRANSLATION). Figure 3 shows a toy example of the Recipes table, with two columns and 5 tuples. To transform $\mathcal{Q}$ into an ILP, we first create a non-negative, integer variable for each tuple: $x_1, \dots, x_5$. The cardinality constraint specifies that the sum of the x_i variables should be exactly 3. The global constraint on $\text{SUM}(P.\text{kcal})$ is formed by multiplying each x_i with the value of the kcal column of the corresponding tuple, and specifying that the sum should be between 2 and 2.5. The objective of minimizing $\text{SUM}(P.\text{sat_fat})$ is similarly formed by multiplying each x_i with the sat_fat value of the corresponding tuple.

3.2. Query evaluation with DIRECT

Using the ILP formulation, we develop DIRECT, our basic evaluation method for package queries. In Section 4, we extend this technique to our main algorithm, SKETCHREFINE, which supports efficient package evaluation in large datasets. Package evaluation with DIRECT employs three steps:

1. **Base Relations:** We first compute the base relations, such as R_β , R_c , and R_p , with a series of standard SQL queries, one for each, or by simply scanning R once and populating these relations simultaneously.
2. **ILP Formulation:** We transform the PaQL query to an ILP problem using the rules described in Section 3.1. After this phase, all variables x_i such that $x_i = 0$ can be eliminated from the ILP problem because the corresponding tuple t_i cannot appear in any package solution.
3. **ILP Execution:** We employ an off-the-shelf ILP solver, as a black box, to get a solution to each of the integer variables x_i . Each x_i informs the number of times tuple t_i should be included in the answer package.

EXAMPLE 3 (ILP SOLUTION). The ILP solver operating on the program of Figure 3 returns the variable assignments to x_i that lead to the optimal solution; $x_i = 0$ means that tuple t_i is

Figure 3. Example ILP formulation and solution for query $\mathcal{Q}$, on a sample Recipe dataset. There are only two packages that satisfy all the constraints, namely $\{t_2, t_3, t_5\}$ and $\{t_1, t_2, t_5\}$, but the first one is the optimal because it minimizes the objective function.

Recipes				
	sat	fat	kcal	
t_1	7.1	0.45		$x_1 = 0$
t_2	5.2	0.55		$x_2 = 1$
t_3	3.2	0.25		$x_3 = 1$
t_4	6.5	0.15		$x_4 = 0$
t_5	2.0	1.20		$x_5 = 1$

min	$7.1x_1 + 5.2x_2 + 3.2x_3 + 6.5x_4 + 2.0x_5$
s.t.	$x_1 + x_2 + x_3 + x_4 + x_5 = 3$
	$0.45x_1 + 0.55x_2 + 0.25x_3$
	$+ 0.15x_4 + 1.20x_5 \geq 2.0$
	$0.45x_1 + 0.55x_2 + 0.25x_3$
	$+ 0.15x_4 + 1.20x_5 \leq 2.5$
	$x_1, x_2, x_3, x_4, x_5 \in [0, 1]$

not included in the output package, and $x_i = k$ means that tuple t_i is included k times. Thus, the result of $\mathcal{Q}$ is the package: $\{t_2, t_3, t_5\}$.

4. SCALABLE PACKAGE EVALUATION

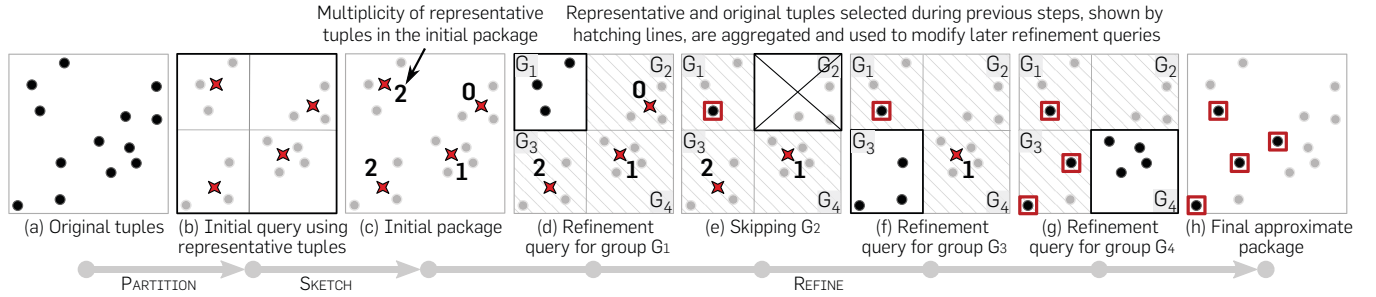
The DIRECT algorithm has two crucial drawbacks. First, it is only applicable if the input relation is small enough to fit entirely in main memory: ILP solvers, such as IBM's CPLEX, require the entire problem to be loaded in memory before execution. Second, even for problems that fit in main memory, this approach may fail due to the complexity of the integer problem. In fact, ILP is a notoriously hard problem, and modern ILP solvers use algorithms, such as *branch-and-cut*,¹⁶ that often perform well in practice, but can “choke” even on small problem sizes due to their exponential worst-case complexity.⁸ This may result in unreasonable performance if the solvers use too many resources (main memory, virtual memory, CPU time), eventually thrashing the entire system.

In this section, we present SKETCHREFINE, an approximate divide-and-conquer evaluation technique for efficiently answering package queries on large datasets. Rather than solving the original large problem with DIRECT, SKETCHREFINE smartly decomposes a query into smaller queries, formulates them as ILP problems, and employs an ILP solver as a black-box evaluation method to answer each individual query. By breaking down the problem into smaller subproblems, the algorithm avoids the drawbacks of DIRECT.

The algorithm is based on an important observation: *similar tuples are likely to be interchangeable within packages*. A group of similar tuples can therefore be “compressed” to a single *representative tuple* for the entire group. SKETCHREFINE *sketches* an initial answer package using only the set of representative tuples, which is substantially smaller than the original dataset. This initial solution is then *refined* by evaluating a subproblem for each group, iteratively replacing the representative tuples in the current package solution with original tuples from the dataset. Figure 4 provides a high-level illustration of the three main steps of SKETCHREFINE:

1. **Offline Partitioning (Section 4.1):** The algorithm assumes a partitioning of the data into groups of similar tuples, with a representative tuple chosen for each group. This partitioning is performed offline (not at query time).
2. **Sketch (Section 4.2.1):** SKETCHREFINE sketches an

Figure 4. The original tuples (a) are partitioned into four groups and a representative is constructed for each group (b). The initial sketch package (c) contains only representative tuples, with possible repetitions up to the size of each group. The refine query for group G_1 (d) involves the original tuples from G_1 and the aggregated solutions to all other groups (G_2 , G_3 , and G_4). Group G_2 can be skipped (e) because no representatives could be picked from it. Any solution to previously refined groups is used while refining the solution for the remaining groups (f and g). The final approximate package (h) contains only original tuples.



initial package by evaluating the package query only over the set of representative tuples.

3. **Refine (Section 4.2.2):** Finally, SKETCHREFINE transforms the initial package into a complete package by replacing each representative tuple with some of the original tuples from the same group, one group at a time.

SKETCHREFINE always constructs *approximate feasible* packages, that is, packages that satisfy all the query constraints, but with a possibly sub-optimal objective value that is guaranteed to be within certain approximation bounds. SKETCHREFINE may suffer from *false infeasibility*, which happens when the algorithm reports a feasible query to be infeasible. The probability of false infeasibility is, however, low and bounded. We formalize these properties in Section 4.3.

In the subsequent discussion, we use $R(\text{attr}_1, \dots, \text{attr}_k)$ to denote an input relation with k attributes. R is partitioned into m groups $G_1, \dots, G_m$. Each group $G_i \subseteq R$, $1 \leq i \leq m$, has a representative tuple $\tilde{t}_i$, which may not always appear in R . We denote the partitioned space with $\mathcal{P} = \{(G_i, \tilde{t}_i) | 1 \leq i \leq m\}$. We refer to packages that contain representative tuples as *sketch packages* and packages with only original tuples as *complete packages* (or simply *packages*). We denote a complete package with p and a sketch package with p_s , where $\mathcal{S} \subseteq \mathcal{P}$ is the set of groups that are yet to be refined to transform p_s to a complete answer package p .

4.1. Offline partitioning

SKETCHREFINE relies on an offline partitioning of the input relation R into groups of similar tuples. Partitioning is based on a set of *partitioning attributes* from the input relation R , a *size threshold*, and a set of *diameter bounds*. The size threshold τ , $1 \leq \tau \leq n$, restricts the size of each partitioning group G_i , $1 \leq i \leq m$, to a maximum of τ original tuples, that is, $|G_i| \leq \tau$. The *diameter* $d_{ij} \geq 0$ of a group G_i , $1 \leq i \leq m$, on attribute attr_j , $1 \leq j \leq k$, is the greatest absolute distance between all pairs of tuples within group G_i . The *diameter bounds*, $\omega_{ij} \geq 0$, $1 \leq i \leq m$, $1 \leq j \leq k$, require all diameters to be bounded by $d_{ij} \leq \omega_{ij}$.

Setting the partitioning parameters. The size threshold, τ , affects the number of partitions, m : a lower τ leads to smaller partitions, but more of them (larger m). For best response time of SKETCHREFINE, τ should be set so that both m and τ are small. Our experiments show that a proper

setting can lead to an order of magnitude improvement in query response time.

The diameter bounds, ω_{ij} , are not required, but they can be enforced to ensure a desired approximation guarantee. In general, enforcing the diameter limits may cause the resulting partitions to become excessively small. While still obeying the approximation guarantees, this could increase the number of resulting partitions and thus degrade the running time performance of SKETCHREFINE. This is an important trade-off between running time and quality that we also observe in our experiments, and it is a very common characteristic of most approximation schemes.²⁴

Partitioning method. Our partitioning procedure is based on *k-dimensional quad-tree indexing*.¹¹ The method recursively partitions a relation into groups until all the groups satisfy the size threshold and meet the diameter limits. First, relation R is augmented with an extra group ID column gid , such that $t.\text{gid} = i$ if tuple t is assigned to group G_i . The procedure initially creates a single group G_1 that includes all the original tuples from relation R , by initializing $\text{gid} = 1$ for all tuples. Our method recursively computes the sizes and diameters of the current groups, as well as the *centroid* of each group. It then partitions the groups that violate either the size or the diameter limits, using the centroids as partitioning boundaries. In the last iteration, the centroids for each group become the representative tuples, $\tilde{t}_i$, $1 \leq i \leq m$, and get stored in a new *representative relation* $\tilde{R}(\text{gid}, \text{attr}_1, \dots, \text{attr}_k)$.

One-time cost. Partitioning is an expensive procedure. Partitioning the data in advance avoids this cost at query time. For a known workload, our experiments show that partitioning the dataset on the union of all query attributes provides the best performance in terms of query evaluation time and approximation error for the computed answer package. We also demonstrate that our query evaluation approach is robust to a wide range of partition sizes, and to imperfect partitions that cover more or fewer attributes than those used in a particular query. This means that, even without a known workload, a partitioning performed on all of the data attributes still provides good performance. Note that the same partitioning can be used to support different queries over the same dataset. In our

experiments, we show that a single partitioning performs consistently well across different queries.

4.2. Query evaluation with SKETCHREFINE

During query evaluation, SKETCHREFINE first *sketches* a package solution using the representative tuples (SKETCH), and then it *refines* it by replacing representative tuples with original tuples (REFINE). We describe these steps using the example query Q from Figure 2.

SKETCH. Using the representative relation $\tilde{R}$ produced by the partitioning, the SKETCH procedure constructs and evaluates a *sketch query*, $Q(\tilde{R})$. The result is an initial sketch package, p_s , containing representative tuples that satisfy the same constraints as the original query Q :

```
Q( $\tilde{R}$ ): SELECT      PACKAGE(*) AS  $p_s$ 
FROM               $\tilde{R}$ 
WHERE              $\tilde{R}.\text{gluten} = \text{'free'}$ 
SUCH THAT
  COUNT( $p_s.*$ ) = 3 AND
  SUM( $p_s.\text{kcal}$ ) BETWEEN 2.0 AND 2.5 AND
  (SELECT COUNT(*) FROM  $p_s$  WHERE gid = 1)  $\leq |G_1|$ 
  AND ...
  (SELECT COUNT(*) FROM  $p_s$  WHERE gid =  $m$ )  $\leq |G_m|$ 
MINIMIZE         SUM( $p_s.\text{sat\_fat}$ )
```

The new global constraints (in bold) ensure that every representative tuple does not appear in p_s more times than the size of its group, G_i . This accounts for the repetition constraint REPEAT 0 in the original query. Generalizing, with REPEAT ρ , each $\tilde{t}_i$ can be repeated up to $|G_i|(1 + \rho)$ times. These constraints are omitted from $Q(\tilde{R})$ if the original query does not contain a repetition constraint.

Since the representative relation $\tilde{R}$ contains exactly m representative tuples, the ILP problem corresponding to this query has only m variables. This is typically small enough for the black-box ILP solver to manage directly, and thus we can solve this package query using the DIRECT method. If m is too large, we can solve this query *recursively* with SKETCHREFINE: the set of m representatives is further partitioned into smaller groups until the subproblems reach a size that can be efficiently solved directly.

The SKETCH procedure *fails* if the sketch query $Q(\tilde{R})$ is infeasible, in which case SKETCHREFINE reports the original query Q as infeasible. This may constitute *false infeasibility*, if Q is actually feasible. In Section 4.3, we show that the probability of false infeasibility is low and bounded.

REFINE. Using the sketched solution over the representative tuples, the REFINE procedure iteratively replaces the representative tuples with tuples from the original relation R , until no more representatives are present in the package. The algorithm *refines* the sketch package p_s one group at a time. For a group G_i with representative $\tilde{t}_i$, let $\bar{p}_i \subseteq p_s$ be the set of representatives picked from G_i (i.e., $\tilde{t}_i$ with possible duplicates). The algorithm proceeds as follows:

- It derives package $\bar{p}_i$ from p_s , by eliminating all instances of $\tilde{t}_i$ from p_s . That is, $\bar{p}_i = p_s \setminus \tilde{p}_i$. This is a solution to all groups except G_i .
- The algorithm then constructs a *refine query*, $Q_i(p_s)$, which searches for a set of tuples $p_i \subseteq G_i$ to replace the eliminated representatives:

```
Q $_i$ ( $p_s$ ): SELECT      PACKAGE(*) AS  $p_i$ 
FROM               $G_i$  REPEAT 0
WHERE              $G_i.\text{gluten} = \text{'free'}$ 
SUCH THAT
  COUNT( $p_i.*$ ) + COUNT( $\bar{p}_i.*$ ) = 3 AND
  SUM( $p_i.\text{kcal}$ ) + SUM( $\bar{p}_i.\text{kcal}$ ) BETWEEN 2.0 AND 2.5
MINIMIZE         SUM( $p_i.\text{sat\_fat}$ )
```

- The algorithm adds the result of $Q_i(p_s)$, p_i in the current solution, p_s . Now, group G_i is *refined* with actual tuples.

In $Q_i(p_s)$, COUNT($\bar{p}_i.*$) and SUM($\bar{p}_i.\text{kcal}$) are values computed directly on $\bar{p}_i$ before the query is formed. They are used to modify the original constraint bounds to account for tuples and representatives already chosen for all the other groups. The global constraints in $Q_i(p_s)$ ensure that the combination of tuples in p_i and $\bar{p}_i$ satisfy the original query Q . Thus, this step produces the new *refined sketch package* $p'_{s'} = p_i \cup p_{\bar{p}_i}$, where $S' = S \setminus \{(G_i, \tilde{t}_i)\}$.

Since G_i has at most τ tuples, the ILP problem corresponding to $Q_i(p_s)$ has at most τ variables. This is typically small enough for the black-box ILP solver to solve using the DIRECT method. Similar to the sketch query, if τ is too large, SKETCHREFINE can evaluate the query recursively: the tuples in group G_i are further partitioned into smaller groups until the subproblems reach a size that can be efficiently solved directly.

Ideally, the REFINE step will only process each group with representatives in the initial sketch package once. However, the order of refinement matters as each refinement step is greedy: it selects tuples to replace the representatives of a single group, without considering the effects of this choice on other groups. As a result, a particular refinement step may render the query infeasible (no tuples from the remaining groups can satisfy the constraints). When this occurs, REFINE employs a *greedy backtracking* strategy that reconsiders groups in a different order.

Greedy backtracking. REFINE activates backtracking when it encounters an infeasible *refine query*, $Q_i(p_s)$. Backtracking *greedily prioritizes* the infeasible groups. This choice is motivated by a simple heuristic: if the refinement on G_i fails, it is likely due to choices made by previous refinements; therefore, by prioritizing G_i , we reduce the impact of other groups on the feasibility of $Q_i(p_s)$. This heuristic does not affect the approximation guarantees.

The algorithm logically traverses a *search tree* (which is only constructed as new branches are created and new nodes visited), where each node corresponds to a unique sketch package p_s . The traversal starts from the *root*, corresponding to the initial sketch package, where no groups have been refined ($S = \mathcal{P}$), and finishes at the first encountered *leaf*, corresponding to a complete package ($S = \emptyset$). The

algorithm terminates as soon as it encounters a complete package, which it returns. The algorithm assumes a (initially random) refinement order for all groups in $\mathcal{S}$ and places them in a priority queue. During refinement, this group order can change by prioritizing groups with infeasible refinements.

Runtime complexity. In the best case, all refine queries are feasible and the algorithm never backtracks. In this case, the algorithm makes up to m calls to the ILP solver to solve problems of size up to τ , one for each refining group. In the worst case, SKETCHREFINE tries every group ordering leading to an exponential number of calls to the ILP solver. Our experiments show that the best case is the most common and backtracking occurs infrequently.

4.3. Theoretical guarantees

We present two important results on the theoretical guarantees of SKETCHREFINE: (1) it produces packages that closely approximate the objective value of the packages produced by DIRECT; (2) the probability of false negatives (i.e., queries incorrectly deemed infeasible) is low and bounded. The extended version of this work⁴ includes the formal proofs of both results.

For a desired approximation parameter ε , we can derive diameter bounds ω_{ij} for the offline partitioning that guarantee that SKETCHREFINE will produce a package with objective value $(1 \pm \varepsilon)$ -factor close to the objective value of the solution generated by DIRECT for the same query.

THEOREM 1 (APPROXIMATION BOUNDS). *Let $R(\text{attr}_1, \dots, \text{attr}_k)$ be a relation with k attributes, and let $\mathcal{Q}$ be a feasible package query with a maximization (minimization, resp.) objective over R . Let S be an exact solver that produces an answer to $\mathcal{Q}$ with optimal objective value OPT . We denote with ALG the objective value of the package returned by SKETCHREFINE using S as a black-box solver. For any $\varepsilon \in [0, 1)$ ($\varepsilon \in [0, \infty)$, resp.), there exists $\beta \in [0, 1)$ ($\beta \in [1, \infty)$, resp.) that depends on ε , such that if R is partitioned into m groups with diameter limits:*

$$\omega_{ij} = \min_{t \in G_i} \{ |1 - \beta| \cdot |t.\text{attr}_j| \}, \quad \forall i \in [1, m], \forall j \in [1, k] \quad (1)$$

then $ALG \geq (1 - \varepsilon)OPT$ ($ALG \leq (1 + \varepsilon)OPT$, resp.).

For a feasible query $\mathcal{Q}$, false infeasibility may happen in two cases: (1) when the sketch query $\mathcal{Q}(\tilde{R})$ is infeasible; (2) when greedy backtracking fails (possibly due to suboptimal partitioning). In both cases, SKETCHREFINE would (incorrectly) report a feasible package query as infeasible. False negatives are, however, extremely rare, as the following theorem establishes.

THEOREM 2 (FALSE-INFEASIBILITY BOUNDS). *For any query $\mathcal{Q}$ and any random package P , if P is feasible for $\mathcal{Q}$, then with high probability: (1) the SKETCH query $\mathcal{Q}(\tilde{R})$ is feasible; (2) all REFINE queries $\mathcal{Q}(p_s)$, $1 \leq i \leq m$, are feasible. Thus, SKETCHREFINE returns a feasible result.*

5. EXPERIMENTAL EVALUATION

This section presents an extensive experimental evaluation of

our techniques for package query execution on real-world data. The results show the following properties of our methods: (1) SKETCHREFINE evaluates package queries an order of magnitude faster than DIRECT; (2) SKETCHREFINE scales up to sizes that DIRECT cannot handle directly; (3) SKETCHREFINE produces packages of high quality (similar objective value as the packages returned by DIRECT). We have also performed extensive experiments on benchmark data that demonstrate the robustness of SKETCHREFINE under imperfect partitioning and different approximation parameters.^{4,5}

5.1. Experimental setup

We implemented our package evaluation system as a layer on top of PostgreSQL.^a The system interacts with the DBMS via SQL and uses IBM's CPLEX¹² as the black-box ILP solver. A package is materialized into the DBMS as a relation, only when necessary (e.g., to compute its objective value). The experiments compare DIRECT with SKETCHREFINE. Both methods use the PaQL to ILP translation presented in Section 3.1: DIRECT translates and solves the original query; SKETCHREFINE translates and solves the subqueries. We demonstrate the performance of our query evaluation methods using a real-world dataset consisting of approximately 5.5 million tuples extracted from the Galaxy view of the SDSS,²² and a workload of seven feasible package queries (Figure 5) constructed by adapting some of the real-world sample SQL queries available directly from the SDSS Website. The experiments use the following efficiency and effectiveness metrics:

Response time. We measure response time as wall-clock time to generate an answer package. This includes the time to translate the PaQL query into one or several ILP problems, the time to load the problems to the solver, and the time the solver takes to produce a solution.

Approximation ratio. We compare the objective value of a package returned by SKETCHREFINE with the objective value of the package returned by DIRECT on the same query. Using Obj_s and Obj_D to denote the objective values of SKETCHREFINE and DIRECT, respectively, we report the empirical *approximation ratio* $\frac{Obj_D}{Obj_s}$ for maximization queries, and $\frac{Obj_s}{Obj_D}$ for minimization queries. An approximation ratio of one indicates that SKETCHREFINE produces a solution with same objective value as the solution produced by the solver on the entire problem. The higher the approximation ratio, the lower the quality of the result package.

5.2. Results and discussion

We evaluate two fundamental aspects of our algorithms: (1)

^a Our code is publicly available on our project Website: <http://packagebuilder.cs.umass.edu>.

Figure 5. Summary of queries in the Galaxy workload. The full PaQL queries appear in the extended version of this work.⁴

Query	$\mathcal{Q}_1$	$\mathcal{Q}_2$	$\mathcal{Q}_3$	$\mathcal{Q}_4$	$\mathcal{Q}_5$	$\mathcal{Q}_6$	$\mathcal{Q}_7$
Objective	max	min	min	min	min	min	max
# of SUM constraints	2	4	2	1	1	5	5
COUNT (*)							

BETWEEN 5 AND 10

their query response time and approximation ratio with increasing dataset sizes; (2) the impact of varying partitioning size thresholds, τ , on SKETCHREFINE's performance.

Query performance as dataset size increases. The first set of experiments evaluates the scalability of our methods on input relations of increasing size. First, we partition each dataset using the union of all package query attributes in the workload: we refer to these partitioning attributes as the *workload attributes*. We do not enforce diameter conditions, ω_{ij} , during partitioning for three reasons: (1) because the diameter conditions may affect the size of the resulting partitions, and we want to tightly control the partition size through the parameter τ ; (2) to show that an offline partitioning can be used to answer efficiently and effectively both maximization and minimization queries, even though they would normally require different diameters; (3) to demonstrate the effectiveness of SKETCHREFINE in practice, even without having theoretical guarantees in place.

We perform offline partitioning with partition size threshold τ set to 10% of the dataset size. We derive the partitionings for the smaller data sizes (less than 100% of the dataset), by randomly removing tuples from the original partitions. This operation is guaranteed to maintain the size condition.

Figure 6 reports our scalability results on the Galaxy workload. The figure displays the query response time in seconds on a logarithmic scale, averaged across 10 runs for each datapoint. At the bottom of each plot, we also report the mean and median approximation ratios across all dataset sizes. The graph for Q2 does not report approximation ratios because DIRECT evaluation fails to produce a solution

for this query across all data sizes. We observe that DIRECT can scale up to millions of tuples in three of the seven queries. Its runtime performance degrades, as expected, when data size increases, but even for very large datasets DIRECT is usually able to answer the package queries in less than a few minutes. However, DIRECT has high failure rate for some of the queries, indicated by the missing data points in some graphs (queries Q2, Q3, Q6, and Q7). This happens when CPLEX uses the entire available main memory while solving the corresponding ILP problems. For some queries, such as Q3 and Q7, this occurs with bigger dataset sizes. However, for queries Q2 and Q6, DIRECT even fails on small data. This is a clear demonstration of one of the major limitations of ILP solvers: they can fail even when the dataset can fit in main memory, due to the complexity of the integer problem. In contrast, our scalable SKETCHREFINE algorithm is able to perform well on all dataset sizes and across all queries. SKETCHREFINE consistently performs about an order of magnitude faster than DIRECT across all queries. Its running time is consistently below one or two minutes, even when constructing packages from millions of tuples.

Both the mean and median approximation ratios are very low, usually all close to one or two. This shows that the substantial gain in running time of SKETCHREFINE over DIRECT does not compromise the quality of the resulting packages. Our results indicate that the overhead of partitioning with diameter limits is often unnecessary in practice. Since the approximation ratio is not enforced, SKETCHREFINE can potentially produce bad solutions, but this happens rarely.

Effect of varying partition size threshold. In the second set of experiments, we vary τ , which is used during

Figure 6. Scalability on the Galaxy workload. SKETCHREFINE uses an offline partitioning computed on the full dataset, using the workload attributes, $\tau = 10\%$ of the dataset size, and no diameter condition. DIRECT scales up to millions of tuples in about half of the queries, but it fails on the other half. SKETCHREFINE scales well in all cases and runs about an order of magnitude faster than DIRECT. Its approximation ratio is always low, even though the partitioning is constructed without diameter conditions.

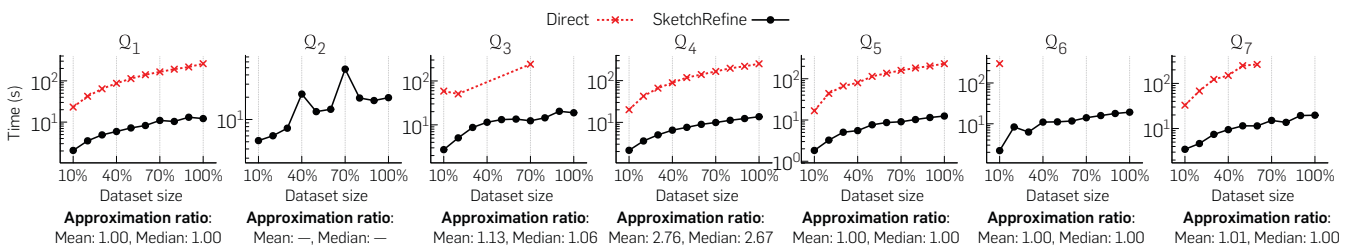
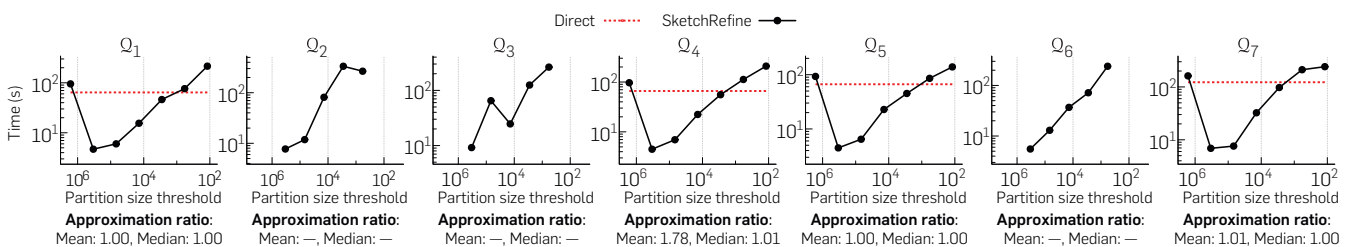


Figure 7. Impact of partition size threshold τ on the Galaxy workload, using 30% of the original dataset. Partitioning is performed at each value of τ using all the workload attributes, and with no diameter condition. The baseline DIRECT and the approximation ratios are only shown when DIRECT is successful. The results show that τ has a major impact on the running time of SKETCHREFINE, but almost no impact on the approximation ratio. DIRECT can be an order of magnitude faster than DIRECT with proper tuning of τ .



partitioning to limit the size of each partition, to study its effects on the query response time and the approximation ratio of SKETCHREFINE. In all cases, along the lines of the previous experiments, we do not enforce diameter conditions. Figure 7 show the results obtained on the Galaxy workload, using 30% of the original data. We vary τ from higher values corresponding to fewer but larger partitions, on the left-hand side of the x -axis, to lower values, corresponding to more but smaller partitions. When DIRECT is able to produce a solution, we also report its running time (horizontal line) as a baseline for comparison.

The results show that the partition size threshold has a major impact on the execution time of SKETCHREFINE, with extreme values of τ (either too low or too high) often resulting in slower running times than DIRECT. With bigger partitions, on the left-hand side of the x -axis, SKETCHREFINE takes about the same time as DIRECT because both algorithms solve problems of comparable size. When the size of each partition starts to decrease, moving from left to right on the x -axis, the response time of SKETCHREFINE decreases rapidly, reaching about an order of magnitude improvement with respect to DIRECT. Most of the queries show that there is a “sweet spot” at which the response time is the lowest: when all partitions are small, and there are not too many of them. This point is consistent across different queries, showing that it only depends on the input data size. After that point, although the partitions become smaller, the number of partitions starts to increase significantly. This increase has two negative effects: it increases the number of representative tuples, and thus the size and complexity of the initial SKETCH query, and it increases the number of groups that REFINE may need to refine to construct the final package. This causes the running time of SKETCHREFINE, on the right-hand side of the x -axis, to increase again and reach or surpass the running time of DIRECT. The mean and median approximation ratios are in all cases very close to one, indicating that SKETCHREFINE retains very good quality regardless of the partition size threshold.

6. CONCLUSION AND FUTURE WORK

We introduced a complete system that supports the declarative specification and efficient evaluation of package queries. We presented PaQL, a declarative extension to SQL, and we developed a flexible approximation method, with strong theoretical guarantees, for the evaluation of PaQL queries on large-scale datasets. Our experiments on real-world data demonstrate that our scalable evaluation strategy is effective and efficient over varied data sizes and queries. We have further extended our techniques and experimental evaluation and placed our research in the context of related work.⁴

Our work so far focused on *deterministic* package queries, but many applications of constrained optimization require support for uncertainty: airline fleet scheduling has uncertain passenger demands, or investment portfolio optimization deals with uncertain returns and risks, etc. We are currently working on extending our system to support optimization of the expected value of an objective function subject to *expectation constraints* of the form $E(\text{SUM}(x)) \geq b$, or *probabilistic constraints* of the form $\text{SUM}(x) \geq b$ WITH PROBABILITY $\geq 95\%$. The challenge is to ensure robust optimal solutions, computed

efficiently, that behave well under the many possible realizations of the uncertain data.

Another open problem is to efficiently handle *incremental* package queries to enable user-facing, interactive constrained optimization applications such as vacation planning. Rather than calling the solver for each incremental query variation from scratch, we are exploring the use of efficient database techniques, such as top-k querying, to provide faster, albeit approximate, solutions for interactive applications.

Acknowledgments

This research is supported by the National Science Foundation under grants IIS-1420941, IIS-1421322, and IIS-1453543. 

References

- Alagoz, O., Schaefer, A.J., Roberts, M.S. *Optimizing Organ Allocation and Acceptance*. Springer, Boston, MA, 2009, 1–24.
- Baykasoglu, A., Dereli, T., Das, S. Project team selection using fuzzy optimization approach. *Cybern. Syst.* 38, 2 (2007), 155–185.
- Bisschop, J. *AIMMS Optimization Modeling*. Paragon Decision Technology, 2006.
- Brucato, M., Abouzied, A., Meliou, A. Package queries: efficient and scalable computation of high-order constraints. *VLDB J.* (Oct. 2017).
- Brucato, M., Beltran, J.F., Abouzied, A., Meliou, A. Scalable package queries in relational database systems. *PVLDB* 9, 7 (2016), 576–587.
- Brucato, M., Ramakrishna, R., Abouzied, A., Meliou, A. PackageBuilder: From tuples to packages. *PVLDB* 7, 13 (2014), 1593–1596.
- Chen, D.-S., Batson, R.G., Dang, Y. *Applied Integer Programming: Modeling and Solution*. John Wiley & Sons, 2011.
- Cook, W., Hartmann, M. On the complexity of branch and cut methods for the traveling salesman problem. *Polyhedral Comb.* 1 (1990), 75–82.
- De Choudhury, M., Feldman, M., Amer-Yahia, S., Golbandi, N., Lempel, R., Yu, C. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia* (Toronto, Ontario, Canada, June 13–16, 2010) ACM, NY, 35–44.
- Deng, T., Fan, W., Geerts, F. On the complexity of package recommendation problems. In *PODS '12 Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Scottsdale, Arizona, USA, May 21–23, 2012) ACM, NY, 261–272.
- Finkel, R.A., Bentley, J.L. Quad trees a data structure for retrieval on composite keys. *Acta Inf.* 4, 1 (1974), 1–9.
- IBM CPLEX Optimization Studio. <http://www.ibm.com/software/commerce/optimization/cplex-optimizer/>.
- Lappas, T., Liu, K., Terzi, E. Finding a team of experts in social networks. In *KDD '09 Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Paris, France, June 28–July 01, 2009) ACM, NY, 467–476.
- Makuch, W.M., Dodge, J.L., Ecker, J.G., Granfors, D.C., Hahn, G.J. Managing consumer credit delinquency in the us economy: A multi-billion dollar management science application. *Interfaces* 22, 1 (1992), 90–109.
- Meliou, A., Suciu, D. Tiresias: The database oracle for how-to queries. In *SIGMOD '12 Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (Scottsdale, Arizona, USA, May 20–24, 2012) ACM, NY, 337–348.
- Padberg, M., Rinaldi, G. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.* 33, 1 (1991), 60–100.
- Parameswaran, A.G., Venetis, P., Garcia-Molina, H. Recommendation systems with complex constraints: A course recommendation perspective. *ACM TOIS* 29, 4 (2011), 1–33.
- Pinel, F., Varshney, L.R. Computational creativity for culinary recipes. In *CHI EA '14 CHI '14 Extended Abstracts on Human Factors in Computing Systems* (Toronto, Ontario, Canada, April 26–May 01, 2014) ACM, NY, 439–442.
- Rushmeier, R.A., Kontogiorgis, S.A. Advances in the optimization of airline fleet assignment. *Transp. Sci.* 31, 2 (1997), 159–169.
- Sauer, O.A., Shepard, D.M., Mackie, T.R. Application of constrained optimization to radiotherapy planning. *Med. Phys.* 26, 11 (1999), 2359–2366.
- Terrer, J.M.A., Benede, M.A.N., del Rio, E.B., Llanas, S.C. A feasible application of constrained optimization in the IMRT system. *IEEE Trans. Biomed. Eng.* 54, 3 (2007), 370–379.
- The Sloan Digital Sky Survey. <http://www.sdss.org/>.
- Wang, X., Dong, X.L., Meliou, A. In *SIGMOD '15 Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data* (Melbourne, Victoria, Australia, May 31–June 04, 2015) ACM, NY, 1231–1245.
- Williamson, D.P., Shmoys, D.B. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

Matteo Brucato and Alexandra Meliou ([matteo,ameli]@cs.umass.edu), College of Information and Computer Sciences, University of Massachusetts, Amherst, MA, USA.

Azza Abouzied (azza@nyu.edu), Computer Science, New York University, Abu Dhabi, UAE.

Southern University of Science and Technology (SUSTech)

Tenure-Track Faculty Positions

The Department of Computer Science and Engineering (CSE, <http://cse.sustc.edu.cn/en/>), Southern University of Science and Technology (SUSTech) has multiple Tenure-track faculty openings at all ranks, including Professor/Associate Professor/Assistant Professor. We are looking for outstanding candidates with demonstrated research achievements and keen interest in teaching, in the following areas (but are not restricted to):

- Data Science
- Artificial Intelligence
- Computer Systems (including Networks, Cloud Computing, IoT, Software Engineering, etc.)
- Cognitive Robotics and Autonomous Systems
- Cybersecurity (including Cryptography)

Applicants should have an earned Ph.D. degree and demonstrated achievements in both research and teaching. The teaching language at SUSTech is bilingual, either English or Putonghua. It is perfectly acceptable to use English in all lectures, assignments, exams. In fact, our existing faculty members include several non-Chinese speaking professors.

Established in 2012, the Southern University of Science and Technology (SUSTech) is a public institution funded by the municipal of Shenzhen, a special economic zone city in China. Shenzhen is a major city located in Southern China, situated immediately north to Hong Kong Special Administrative Region. As one of China's major gateways to the world, Shenzhen is the country's fastest-growing city in the past two decades. The city is the high-tech and manufacturing hub of southern China, home to the world's third-busiest container port, and the fourth-busiest airport on the Chinese mainland. As a picturesque coastal city, Shenzhen is also a popular tourist destination and was named one of the world's 31 must-see tourist destinations in 2010 by The New York Times. Shenzhen ranks the 66th place on the 2017 Global City Competitiveness List, released by the National Academy of Economic Strategy, the Chinese Academy of Social Sciences and United Nations Habitat. By the end of 2016, there were around 20 million residents in Shenzhen.

SUSTech is committed to increase the diversity of its faculty, and has a range of family-friendly policies in place. The university offers competitive salaries and fringe benefits including medical insurance, retirement and housing subsidy, which are among the best in China. Salary and rank will commensurate with qualifications and experience.

We provide some of the best start-up packages in the sector to our faculty members, including one PhD studentship per year, in addition to a significant amount of start-up funding (which can be used to fund additional PhD students and postdocs, research travels, and research equipments).

To apply, please provide a cover letter identifying the primary area of research, curriculum vitae, and research and teaching statements, and forward them to cshire@sustc.edu.cn.

Stevens Institute of Technology ECE Department

Assistant/Associate/Full Professor

The Department of Electrical and Computer at Stevens Institute of Technology invites applications for several tenure-track/tenured faculty positions at the rank of Assistant/Associate/Full Professors, starting on August 16, 2019 or later. Qualified candidates can also be considered for an endowed chair professor position.

Applicants should have earned a Ph.D. in Electrical or Computer Engineering or a related discipline. The department is looking for researchers with a strong funding and publication record in key areas of interest: artificial intelligence, computer architecture, smart and automated systems, electronics and digital system design. Successful applicants are expected to develop a strong externally funded, globally recognized research program. They should also possess a passion for and be committed to excellence in both undergraduate and graduate education.

Stevens Institute of Technology is a private university located in Hoboken, New Jersey. Stevens is an Equal Opportunity Employer that is building a diverse faculty, staff and student body and strongly encourages applications from female and minority candidates as well as veterans and individuals with disabilities. Stevens is an NSF ADVANCE institution committed to equitable practices and policies.

Applications will be accepted until the positions are filled. All applications must be submitted electronically through the HR website at https://stevens.wd5.myworkdayjobs.com/en-US/External/job/Hoboken-NJ---Main-Campus/Assistant-Associate-Professor-Electrical-and-Computer-Engineering_RQ22188. Applicants should submit their curriculum vitae, a research plan (3-5 pages), teaching interests and philosophy, and contact information including at least three references to the HR system. For any inquiries, please contact the Search Committee Chair, Prof. Hong Man (hong.man@stevens.edu).

University of Alabama

Computer Science Faculty Position – Cybersecurity

The University of Alabama is accepting applications for an Associate or Full Professor in the area of Cybersecurity to begin August 2019. A Ph.D. in Computer Science or a closely related field is required. Applicants must demonstrate a strong external funding record, publication record, and Ph.D. graduation rate commensurate with this

level of appointment. Successful applicants are expected to show evidence of a quality research program, effective collaboration with other faculty, and excellence in teaching at both the graduate and undergraduate levels.

The Computer Science Department has 25 faculty members (17 tenured/tenure-track faculty), over 700 undergraduates in an ABET-accredited program, and approximately 40 graduate students. Current faculty members are funded by agencies such as NSF, Google, Departments of Education and Commerce, various Defense agencies, multiple State agencies and other sponsors.

Applicants should apply online at <https://facultyjobs.ua.edu>. For additional details, please contact Dr. Yang Xiao (yangxiao@cs.ua.edu) or visit <http://cs.ua.edu>.

The University of Alabama is an Equal Employment/Equal Educational Opportunity Institution. All qualified applicants will receive consideration for employment without regard to race, color, religion, national origin, sex, sexual orientation, gender identity, gender expression, pregnancy, age, genetic or family medical history information, disability, or protected veteran status, or any other legally protected basis, and will not be discriminated against because of their protected status. Applicants to and employees of this institution are protected under Federal law from discrimination on several bases.

University of South Carolina

Director of Artificial Intelligence Institute

The University of South Carolina is initiating a search for the Director of the new Artificial Intelligence Institute. The pan-University Institute is expected to engage core and affiliated faculty from a range of disciplines. The College of Engineering and Computing is well positioned to support this University-wide Institute and is in the midst of expanding its tenured and tenure-track ranks by over 40 faculty members. The Director will be expected to create the vision for the Institute and lead it to international prominence in several areas of research, real-world applications, work-force preparation, and job creation in intelligent systems.

The new Director will have the opportunity to grow strategic areas of research and oversee innovation of curricula, as well as hire the core faculty and attract as affiliates several dozen faculty members across the university, and spanning all fields (medicine, pharmacy, public health, education, journalism, social work, nursing, business, humanities, physical sciences, engineering, and computing). The Institute will be housed centrally in the University, and the Director will have significant input into design and function of the space.

The Director will be expected to:

- Conduct convergent, team-oriented, high impact research, with a substantial portfolio of competitive and institute-scale research funds from

ACM Journal of Data and Information Quality



*Providing Research and Tools
for Better Data*

ACM JDIQ is a multi-disciplinary journal that attracts papers ranging from theoretical research to algorithmic solutions to empirical research to experiential evaluations. Its mission is to publish high impact articles contributing to the field of data and information quality (IQ).



For further information
or to submit your
manuscript,
visit jdiq.acm.org

external sponsors.

- ▶ Engage with key industries/services in the region and foster an entrepreneurial ecosystem with joint projects, technology transfer, and start-up formation.
- ▶ Advance AI education and training programs across the University and the State.
- ▶ Position the Institute for national prominence in niche areas within 5 years.
- ▶ Lead multidisciplinary project teams
- ▶ Serve as a mentor to junior faculty and students.

Applicants must be of international stature with an exceptional record of published research in high-quality journals, demonstrated ability to attract significant funding from multiple sources, outstanding leadership and administrative skills, and a history of successful graduate student supervision. Their record (including an earned Ph.D. degree in computer science or a closely-re-

lated field) must be commensurate with appointment as a full professor with tenure. The applicant must also show clear evidence of commitment to diversity, equity, and inclusion through research, teaching, and/or service efforts.

Review of applications will begin immediately and continue until the position is filled. Expected start date is August 16, 2019. Interested applicants will apply online at <http://uscjobs.sc.edu/postings/46728> with: (1) a letter of intent, (2) curriculum vitae, (3) a concise description of research plans, and (4) names and contact information of 5 references.

Questions about the search may be directed to: DirectorAIsearch@cec.sc.edu.

The University of South Carolina does not discriminate in educational or employment opportunities on the basis of race, color, religion, national origin, sex, sexual orientation, gender, age, disability, protected veteran status, or genetics.



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.

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

[CONTINUED FROM P. 120] I had to compete with only a few other candidates to get this six-month detail. Do you think there are beings on Prox Cen b who can receive the message, as modulated in the infrared beam, and actually respond?”

Shekhov grinned, “Now you are testing my faith in the mission and the skills of its managers in mission control on Earth. I would not be here long if I doubted the mission’s scientific value and ultimate success.”

Caruthers said, “Stephen Hawking, the English professor, warned that if we contacted space aliens, we would inevitably risk some kind of attack. We’d be at such a disadvantage technologically and intellectually. We’d be overwhelmed in no time and decimated like the indigenous natives of the Americas at the time of the conquistadors. Our command of all the resources on our ancient but familiar Earth and Moon wouldn’t be enough to protect us.”

“Hawking was projecting his guilt for the sins of the European empires in the colonies, along with his own infirmities and impending mortality. Just a timid old professor, he was. I think aliens really might respond to the message in the laser beam. They might greet us as fellow intelligent beings in the endless Universe, but there is no chance they could harm us. For one thing, they are probably simply too far away, not only in distance but in the technological advancement we can expect from future human generations as they come and go many times over.”

Caruthers said, “Do you think they could decode the message in the laser modulation?”

“It’s complex,” Shekhov admitted. “If aliens really do exist, we probably will have to wait for them to decipher it and compose an intelligent reply we would be able to interpret.”

...

Even as Shekhov gave Caruthers a tour of the habitat, the alien mother ship from the Prox Cen b node used its titanic antimatter engines to decelerate into an orbit 200 km above Earth’s equator and scanned the now-terrified population centers below. The primitives, in their view, were still using vulnerable electromagnetic technology so would be easily subdued into harmless members of the processor

Power consumption skyrocketed as the networked humans on Earth and aliens aboard the mother ship burdened each transaction with cryptographic virtual ledger updates.

swarm in the Earth’s arm of the Milky Way Galaxy, compliant with the hive imperative to exploit Solar System resources.

The mother ship now disgorged several thermonuclear devices, detonating them in low orbits above major cities where the electromagnetic pulses would render human power and computer grids nonfunctional. The descriptions of human culture and technology the swarm received in the laser transmission back on Prox Cen b, and now on their mother ship, made the task straightforward. A few retaliatory ICBMs were launched by Russia, the U.K., and the U.S. but were quickly disabled by the mother ship’s high-energy particle cannons, doing no damage other than alert the Earth’s human population to the aliens’ overwhelming force.

It seemed Hawking would be proved right.

Pods filled with billions of minidrones entered the atmosphere and aerobraked until they reached the troposphere, then dispersed on wings like a swarm of attacking hornets. They flew through the night air, identifying humans and their structures through their infrared profiles and attached themselves with neural-connection electrodes to their brainstems, reducing them to compliant zombies. An alien global Wi-Fi network of neural commands and control quickly incor-

porated the now-networked humans into a new node of the swarm.

Once the alien swarm had assimilated them into its hive mind, it quickly assessed and identified the most advanced human technology it could exploit—blockchain.¹ The alien hive quivered with delight at the prospect of adding ironclad reliability to each transaction among network members in its Wi-Fi network by implementing distributed virtual ledgers. With blockchain, radio interference would never corrupt the network’s transactions, ensuring perfect command and control of its members and their collective will. The swarm thus converted *all* of its transactions to blockchain and sent a radio transmission back to Prox Cen b mission control propagating the blockchain technology to neighboring nodes in the Earthly galactic arm.

Power consumption skyrocketed as the networked humans on Earth and aliens aboard the mother ship burdened each transaction with cryptographic virtual ledger updates. Unsustainable heat built up in the circuits, the mother ship’s processors were overwhelmed and exploded, and the neural connectors to the humans shorted out, leaving only an eerie electric blue glow that briefly filled the Solar System, before winking out.

Shekhov, Caruthers, and the other humans still on the Moon heard panicked messages from the last free humans on Earth via the comsat. They feared for all humankind, but the exponential blockchain wave of networked destruction made short work of the aliens and their threat. The crew of the Hawking’s Nightmare facility then received another message from mission control in Moscow via the comsat: Shut the laser. It had fulfilled its purpose—establishing we are not alone but would probably prefer to be. C

Reference

1. Church, Z. *Blockchain, explained*. MIT Sloan School of Management, May 25, 2017; <http://mitsloan.mit.edu/ideas-made-to-matter/blockchain-explained>

David Allen Batchelor (batchelor@alum.mit.edu) is a scientist and computer engineer for data systems at NASA Goddard Space Flight Center, Greenbelt, MD. His first science fiction novel, *The Metalmark Contract*, was published in 2011 by Black Rose Writing, Castroville, TX.

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/3303769

David Allen Batchelor

Future Tense

Hawking's Nightmare

Stephen Hawking warned us not to contact E.T.

YURI SHEKHOV WAS outside the lunar habitat in his space suit, preparing to watch the supply shuttle from Earth fire its retro rockets and land. The sun glinted off the windows of the boxy crew module, attached to its strange collection of spherical pressurized fuel tanks, rocket nozzles, and articulated cushioned footpads, as it hovered suspended atop its rocket exhaust, carefully lowering itself onto the landing pad. In the airless lunar environment, the shuttle did not need to obey any aerodynamic forms or compensate for more than lunar gravity.

Shekhov had talked with the pilot, who reported a nominal status during the shuttle's orbit and braking maneuvers just above the east edge of the lunar hemisphere that was visible from Earth. It looked to be a flawless landing near his optical-beacon habitat, located at 98 degrees east longitude, eight degrees around to the lunar far side in the crater named for American rocketry genius James H. Wyld. The pilot had deftly avoided the structure behind him that itself embodied the purpose of the billion-dollar lunar base—a giant 45-meter telescope financed and built by wealthy Russian fracking tycoon Oleg Volkov. The telescope pointed approximately 45 degrees southward, toward the nearest star, Proxima Centauri, and its planetary consort, Prox Cen b. A nuclear power plant buried 20 feet below the lunar surface nearby supplied a two-megawatt laser that pulsed with infrared light, round the clock, directed by the telescope with milli-arc-second accuracy, toward the exoplanet four light years away. It also supplied enough direct heat to make the human



quarters as comfortable as a Caribbean villa in tourist season.

When the shuttle was secured to the pad and its engines safely deactivated, Shekhov bounced over to it in the light gravity (one-sixth Earth equivalent) and pulled the latch that released the supply capsule from the shuttle. The capsule deployed its wheels and started to roll on a 100-meter roadway to the habitat. The process was automated, leaving him to turn his attention to the space-suited figure of a passenger exiting the airlock of the crew module. Giving a friendly wave, he radioed, "Welcome to Hawking's Nightmare. You're in time for lunch." After living here practically alone for six months to manage the base, he was glad to welcome a new crew member, any new crew member.

The supply capsule docked with the habitat, and the shuttle ignited its engines to propel it back to lunar orbit and where it was scheduled to

rendezvous with an orbiting booster for its return to Earth. Shekhov and the newcomer cycled through the habitat airlock and removed their helmets inside the habitat.

The newcomer's helmet came off and freed a glorious halo of curly red hair that expanded into the low-gravity environment. "Andrea Caruthers reporting for duty sir," she said.

"Welcome," said Shekhov. "Today we are having borscht and roast beef. Enjoy."

The food was surprisingly savory considering it included no naturally raised animal protein but was as nutritious as an Earthly steak and potato, along with an extra-nutritious dessert. As they dug into the dessert, with the taste and consistency of sherbet, chilled, as it was, in a sunless crater beneath the far side's Earthless skies, she said, "It was spectacular orbiting the Moon. Descending over the Neper and Jansky craters, the view was awesome."

"Awesome, indeed," Shekhov agreed. "But the crater walls keep me from seeing Earth. We always keep the laser from pointing directly toward Earth, but it has been a lonely six months. I am able to exchange messages with home only when the comsat flies over, but that is not at all the same as being in Moscow."

"I'm amazed the laser has been operating continuously for eight years! Most people on Earth have dismissed the project as Volkov's folly. Few know the light-travel-time for our messages to Prox Cen b has passed, plus enough time for a reply message to arrive at the speed of light. It's no wonder [CONTINUED ON P. 119]



Let the good talks roll!

SIGUCCS 47th Annual Conference

November 3-6, 2019 | New Orleans, LA

Call for Proposals

The ACM SIGUCCS annual conference brings together IT support professionals from academic institutions around the world to share ideas and experiences delivering information technology in aid of teaching, research, and administration. Join them by proposing a paper, poster, panel, or lightning talk to be delivered at this year's conference in New Orleans on November 3-6, 2019.

Proposals may be accepted in any area of IT support, including:

- Strategy and governance
- Infrastructure and operations
- Instructional technology and design
- Leadership and career development
- Service management
- Lab management and desktop support

Submit an abstract of your proposed presentation by visiting <http://bit.ly/siguccs2019cfp> no later than **March 8th**. Then join us in New Orleans in November to enjoy the conference and the city from ASCII to zydeco!



Learn more about the conference here:

<http://bit.ly/siguccsNOLA>

ACM SIGUCCS is the Special Interest Group on University and College Computing Services

The Pragmatic Wisdom of Michael Stonebraker

Making Databases Work

This book celebrates Michael Stonebraker's accomplishments that led to his 2014 ACM A.M. Turing Award "for fundamental contributions to the concepts and practices underlying modern database systems."

The book describes, for the broad computing community, the unique nature, significance, and impact of Mike's achievements in advancing modern database systems over more than forty years. Today, data is considered the world's most valuable resource, whether it is in the tens of millions of databases used to manage the world's businesses and governments, in the billions of databases in our smartphones and watches, or residing elsewhere, as yet unmanaged, awaiting the elusive next generation of database systems. Every one of the millions or billions of databases includes features that are celebrated by the 2014 Turing Award and are described in this book.

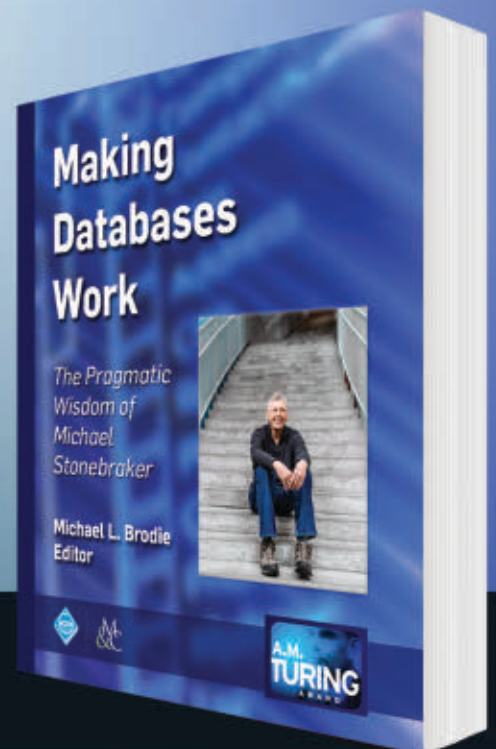
Edited by Michael L. Brodie

ISBN: 978-1-94748-719-2

DOI: 10.1145/3226595

<http://books.acm.org>

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



ACM BOOKS