

# COMMUNICATIONS

CACM.ACM.ORG OF THE ACM 05/2021 VOL.64 NO.05



## Implementing Insider Defenses

Catching the Fakes

The 10 Best Practices for Remote Software Engineering

The Next Phase of Cloud Computing



# THE ACM A. M. TURING AWARD

by the community ♦ from the community ♦ for the community



ACM and Google congratulate

**ALFRED VAINO AHO and JEFFREY DAVID ULLMAN**

For fundamental algorithms and theory underlying programming language implementation and for synthesizing these results and those of others in their highly influential books, which educated generations of computer scientists.



“Aho and Ullman established bedrock ideas about algorithms, formal languages, compilers and databases, which were instrumental in the development of today’s programming and software landscape. They have also illustrated how these various disciplines are closely interconnected. Aho and Ullman introduced key technical concepts, including specific algorithms, that have been essential. In terms of computer science education, their textbooks have been the gold standard for training students, researchers, and practitioners.”

Jeff Dean  
Google Senior Fellow and SVP of Google AI  
Google Inc.

Google™

For more information see <http://research.google.com/>

Financial support for the ACM A. M. Turing Award is provided by Google Inc.

# CALL FOR NOMINATIONS

## 2022 AAI Squirrel AI Award for Artificial Intelligence for the Benefit of Humanity

\$1,000,000 Prize

The AAI Squirrel AI Award for Artificial Intelligence for the Benefit of Humanity recognizes positive impacts of artificial intelligence to protect, enhance, and improve human life in meaningful ways with long-lived effects. The award was given for the first time in 2021.

The award is given annually at the conference for the Association for the Advancement of Artificial Intelligence (AAAI), and is accompanied by a prize of \$1,000,000 plus travel expenses to the conference.

Candidates may be individuals, groups, or organizations that are directly connected with the main contribution stated in the nomination. Qualifications and technical knowledge in artificial intelligence are not requirements for nominations. The emphasis is on the significance and impact of the work.

The award is administered by AAI, with support from the European Artificial Intelligence Association (EurAI). Financial support for the award is provided by Squirrel AI.

**DEADLINE for 2022 Award Nominations: June 15, 2021**

<https://www.aaai.org/Awards/squirrel-ai-award.php>

## Departments

- 4 **Vardi's Insights**  
**The Agency Trilemma and ACM**  
*By Moshe Y. Vardi*
- 
- 5 **In Response to 'Vardi's Insights'**  
*By Cherri M. Pancake,  
Andrew A. Chien*
- 
- 7 **Career Paths in Computing**  
**Promote Sustainability and  
Help Underserved Communities**  
*By Caven Cade Mitchell*
- 
- 8 **BLOG@CACM**  
**Teaching Other Teachers**  
**How to Teach CS Better**  
Mark Guzdial shares how  
he assesses the efforts of other  
computer science teachers.
- 
- 116 **Careers**

## Last Byte

- 120 **Future Tense**  
**Behold the Ch!Ld**  
Opportunity can come calling  
when you least expect it.  
*By P-Ray*

## News



- 10 **A Satisfying Result**  
Formulating a decades-old  
geometric conjecture as  
a satisfiability problem opened  
the door to its final resolution.  
*By Don Monroe*
- 
- 13 **Catching the Fakes**  
Applying neural networks to images  
helps identify counterfeit goods.  
*By Neil Savage*
- 
- 15 **A Traffic Cop for Low Earth Orbit**  
Who will be the space traffic  
controller for orbiting objects?  
*By Keith Kirkpatrick*

## Viewpoints

- 18 **Security**  
**Trustworthy Scientific Computing**  
Addressing the trust issues  
underlying the current limits  
on data sharing.  
*By Sean Peisert*
- 
- 22 **Law and Technology**  
**Software Professionals, Malpractice  
Law, and Codes of Ethics**  
In pursuit of professional status  
for computing professionals.  
*By Bryan H. Choi*
- 
- 25 **Education**  
**CS Unplugged or Coding Classes?**  
Perhaps a more appropriate question  
is 'Why not both?'  
*By Tim Bell*
- 
- 28 **Viewpoint**  
**Understanding Law  
and the Rule of Law: A Plea to  
Augment CS Curricula**  
Why law matters for computer  
scientists and other folk.  
*By Mireille Hildebrandt*
- 
- 32 **Viewpoint**  
**The 10 Best Practices for  
Remote Software Engineering**  
Focusing on the human element  
of remote software engineer  
productivity.  
*By Vanessa Sochat*
- 
- 37 **Viewpoint**  
**Let's Be Honest**  
Seeking to rectify the two mutually  
exclusive ways of comparing  
computational power—encoding  
and simulation.  
*By Nachum Dershowitz*

Practice



42 **Enclaves in the Clouds**  
Legal considerations and broader implications.  
*By Jatinder Singh, Jennifer Cobbe, Do Le Quoc, and Zahra Tarkhani*

52 **Battery Day**  
A closer look at the technology that makes portable electronics possible.  
*By Jessie Frazelle*

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



**About the Cover:** Much has been written about technical protocols for preventing insider cyber-attacks. This month's cover story spotlights some nontechnical solutions that are highly effective and begin with creating a corporate culture of trust. Cover illustration by Peter Crowther Associates, portraits by Shutterstock.

Contributed Articles

60 **Implementing Insider Defenses**  
How to avoid insider cyber-attacks by creating a corporate culture that infuses trust.  
*By Eric Grosse, Fred B. Schneider, and Lynette L. Millett*



Watch the authors discuss this article in the exclusive *Communications* video.  
<https://cacm.acm.org/videos/insider-defenses>

66 **HCDA: From Computational Thinking to a Generalized Thinking Paradigm**  
As a new era in computing emerges, so too must our fundamental thinking patterns.  
*By Yuhang Liu, Xian-He Sun, Yang Wang, and Yungang Bao*



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

76 **What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing**  
The evolution that serverless computing represents, the economic forces that shape it, why it could fail, and how it might fulfill its potential.  
*By Johann Schleier-Smith, Vikram Sreekanti, Anurag Khandehwal, Joao Carreira, Neeraja J. Yadwadkar, Raluca Ada Popa, Joseph E. Gonzalez, Ion Stoica, and David A. Patterson*

Review Articles



86 **Automata Modulo Theories**  
Symbolic automata better balances how automata are implemented in practice.  
*By Loris D'Antoni and Margus Veenes*

Research Highlights

97 **Technical Perspective**  
**A Logical Step Toward the Graph Isomorphism Problem**  
*By Pascal Schweitzer*

98 **Isomorphism, Canonization, and Definability for Graphs of Bounded Rank Width**  
*By Martin Grohe and Daniel Neuen*

106 **Technical Perspective**  
**Robust Statistics Tackle New Problems**  
*By Jacob S. Steinhardt*

107 **Robustness Meets Algorithms**  
*By Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart*



DOI:10.1145/3459084

Moshe Y. Vardi

# The Agency Trilemma and ACM

**T**HE *PRINCIPAL-AGENT PROBLEM*, or the *Agency Dilemma*, arises when one or more persons (the “agents”) are making decisions and taking actions on behalf of other persons (the “principals”). The dilemma occurs in circumstances where agents are motivated to act in their own best interests, which are contrary to those of their principals. A common instance of this dilemma is that of corporate management (agents) and shareholders (principals). A more complicated setting, which can be called the *Agency Trilemma*, is one in which we have two types of agents—for example, elected officials (agents) and civil-service officers (agents)—and citizens (principals). The conflict now is not only between principals and agents, but also between the two different types of agents.

Such a three-way conflict arises also in professional societies, such as ACM, which have members (principals), elected officials (agents), and permanent staff (agents). Aligning the interests of these three groups can be challenging. A prominent failure of such alignment occurred in December 2019, when the American Association of Publishers issued a public letter to U.S. President Trump, endorsed by over 135 publishers, including ACM, urging to abandon a potential executive order that would mandate free release of publications resulting from federally sponsored research. The rationale for the letter is the reality that many scientific and professional societies, including ACM, derive a significant portion their revenues from scholarly publishing. Those revenues help pay for other functions the societies perform, such as producing magazines and newsletters, offering education and outreach programs, and sponsoring awards and conferences.

Yet ACM members broadly believe ACM ought to shift from a subscription-based publishing model to an open access model. The December 2019 letter triggered such an uproar among ACM members that in January 2020 ACM’s President sent a letter to the White House Office of Science and Technology Policy expressing ACM’s regrets having endorsed the letter. In June 2020, ACM Council voted to embark on a five-year transition to an open access publishing model, based on the success of a multiyear pilot.

I was not privy to the deliberations that led to ACM’s President signing the December 2019 letter with the approval of ACM’s Executive Committee (EC), only to retract it a month later. This EC decision, however, cannot be divorced from (valid) concerns by ACM staff, which I have heard many times, regarding the potential impact of open access publishing on ACM revenues. I have worked closely with ACM staff over the past 15 years and I found them to be highly competent, hard-working, and dedicated. With very few exceptions, however, they are not computing professionals, and their perspective is quite different than that of ACM members.

This gap between ACM members and staff is supposed to be bridged by ACM’s elected and appointed officers, who, together with at-large members, comprise ACM Council. The Council has the duty to formulate policies and supervise their execution. But while ACM staff are (by and large) full-time employees, ACM officers are volunteers, and their service is *on top* of their regular jobs. Furthermore, ACM officers have a short tenure, typically serving just two years. Thus, there is a knowledge gap and, consequently, a power gap between the two types of agents. This translates to a growing gap between ACM as an orga-

nization and its membership. This gap has recently led to strong disagreement between ACM staff and an ACM Task Force regarding ACM’s Digital Library.

The gap between members and staff is increasing in significance due to three strategic threats that ACM is facing. First, there is the risk in making the change from subscription-based publishing to open access publishing. ACM has formulated a transition plan, but its successful execution is far from guaranteed. Second, ACM conferences, the second major activity of ACM, may have to completely reinvent themselves in view of the climate-change challenge. The business impact of such a reinvention is far from clear. Thirdly, the younger generation of professionals does not seem to feel the need to join professional associations. While the tech industry, together with computing-degree-program enrollments, have been booming, ACM membership growth has been rather modest.

I believe there is an urgent need for a deep examination of ACM’s structure with the goal of reducing the gaps between ACM and its members. Possible remedies to consider are increasing the length of officers’ tenure, giving elected and appointed officers more influence over ACM operations, and enhancing the authority of ACM’s CEO, who usually is a member of the computing community and is formally responsible for the general administration of the affairs of ACM and its principal office.

As we say in Houston, “ACM, we have a problem!”

Follow me on Facebook and Twitter. 

**Moshe Y. Vardi** ([vardi@cs.rice.edu](mailto:vardi@cs.rice.edu)) is University Professor and the Karen Ostrum George Distinguished Service Professor in Computational Engineering at Rice University, Houston, TX, USA. He is the former Editor-in-Chief of *Communications*.

Copyright held by author.

DOI:10.1145/3459982

# In Response to ‘Vardi’s Insights’

## From ACM’s Past President

**M**OSHE VARDI’S COLUMN ON the preceding page misrepresents some decisions and actions made in regard to ACM’s Open Access (OA) model. As ACM’s President at the time these events took place, he invited my comments but chose not to correct several factual errors I pointed out. That information is included here, by way of clarification.

First, the characterization of ACM’s OA process—which Vardi limits to events between December 2019 and June 2020—as an example of a “gap between ACM members and staff” is contrary to fact. ACM’s progress toward OA began mid-decade. A plan for moving to OA was discussed with ACM Council as early as 2016, and a pilot was instituted soon after. By the end of 2019, the pilot had been evaluated, tweaked, and expanded in a second phase that included more institutions. Each step was discussed and approved by ACM volunteers at meetings with the Publications Board, Executive Committee, and Council; the SIG Governing Board was also provided with regular updates.

Second, the statements about my letter to the Office of Science and Technology are misleading. That letter—which was sent to all ACM members in a Member Bulletin—was not a “retraction.” It stated that ACM indeed opposed the proposed regulation but added our regret that the final wording of the letter emphasized commercial interests rather than those of not-for-profit publishers. I explained ACM’s two concerns with the plan: the aggressiveness of the timeline, and the failure to engage with key stakeholders before setting

the terms. The letter was quite clear that ACM not only supported OA, but that we were already well on the way to achieving it within the next few years.

Finally, it was not the transition to OA that Council approved in June 2020, it was to accelerate the timeline based on member feedback. Our new goal would be to have *complete* OA within five years (that is, required rather than optional, for all SIGs and publications). My farewell Member Bulletin announced that goal, citing the teamwork between volunteers and staff that was involved. Many people responded, applauding the fact that ACM listened to its member community and was “leading the way” for other publishers.

I hope that ACM members will continue to appreciate that there is a unique team relationship between ACM volunteers and staff. It is truly a volunteer-driven organization, and each of us can play a role in shaping future ACM decisions.

**Cherri M. Pancake**, ACM PAST PRESIDENT

**Cherri M. Pancake** is professor emeritus of electrical engineering and computer science at Oregon State University, Corvallis, OR, USA. She served as ACM President from 2018–2020.

Copyright held by author.

## From *Communications’* Editor-in-Chief

**T**HE TWO FOREGOING columns may reflect differing perspectives on ACM and its recent actions. However, readers should understand that Moshe Vardi and Cherri Pancake are both passionate ACM leaders. Each has generously contributed years of service to make our professional society strong. We support open discussion of critical challenges for ACM, so we can meet the challenges of rapid change in the research community, scientific publishing, technical leadership, and computing world.

I find the observation that the interests of membership, staff, and officers are “not always aligned” to be accurate. As *Communications’* Editor-in-Chief, differences in perspective and goals, incentives, and ability to dedicate effort are manifest every day. To strengthen ACM, we must combine these attributes, incentivizing volunteer engagement by making it efficient and impactful, and having clear strategic objectives that enable staff to drive in the direction of ACM’s long-term agenda. We must strengthen this synergy in a world of growing distractions and opportunities that compete for engagement. And we must, if we are to rise and meet the myriad disruptions facing the computing community as opportunity, not disaster.

**Andrew A. Chien**, EDITOR-IN-CHIEF

**Andrew A. Chien** is the William Eckhardt Distinguished Service Professor in the Department of Computer Science at the University of Chicago, Director of the CERES Center for Unstoppable Computing, and a Senior Scientist at Argonne National Laboratory.

Copyright held by author.



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 Cappo  
**Director, Office of Publications**  
Scott E. Delman

**ACM COUNCIL**  
**President**  
Gabriele Kotsis  
**Vice-President**  
Joan Feigenbaum  
**Secretary/Treasurer**  
Elisa Bertino  
**Past President**  
Cherri M. Pancake  
**Chair, SGB Board**  
Jeff Jortner  
**Co-Chairs, Publications Board**  
Jack Davidson and Joseph Konstan  
**Members-at-Large**  
Nancy M. Amato; Tom Crick;  
Susan Dumais; Mehran Sahami;  
Alejandro Saucedo  
**SGB Council Representatives**  
Sarita Adve and 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**  
Wenguang Chen

**PUBLICATIONS BOARD**  
**Co-Chairs**  
Jack Davidson and Joseph Konstan  
**Board Members**  
Jonathan Aldrich; Phoebe Ayers;  
Chris Hankin; Mike Heroux; James Larus;  
Tulika Mitra; Marc Najork;  
Michael L. Nelson; Theo Schlossnagle;  
Eugene H. Spafford; Divesh Srivastava;  
Bhavani Thuraisin; Robert Walker;  
Julie R. Williamson

**ACM U.S. Technology Policy Office**  
Adam Eisgrau  
Director of Global Policy and Public Affairs  
1701 Pennsylvania Ave NW, Suite 200,  
Washington, DC 20006 USA  
T (202) 580-6555; acmpo@acm.org

**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**  
[vacant]  
**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**  
Iliia 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**  
acmhelp@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**  
1601 Broadway, 10<sup>th</sup> Floor  
New York, NY 10019-7434 USA  
T (212) 626-0686  
F (212) 869-0481

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

**Media Kit** [acmm mediasales@acm.org](mailto:acmm mediasales@acm.org)

**Association for Computing Machinery (ACM)**  
1601 Broadway, 10<sup>th</sup> Floor  
New York, NY 10019-7434 USA  
T (212) 869-7440; F (212) 869-0481

**EDITORIAL BOARD**  
**EDITOR-IN-CHIEF**  
Andrew A. Chien  
aic@cacm.acm.org  
**Deputy to the Editor-in-Chief**  
Morgan Denlow  
cacm.deputy.to.aic@gmail.com  
**SENIOR EDITOR**  
Moshe Y. Vardi

**NEWS**  
**Co-Chairs**  
Marc Snir and Alain Chesnais  
**Board Members**  
Tom Conte; Monica Divitini; Mei Kobayashi;  
Rajeev Rastogi; François Sillion

**VIEWPOINTS**  
**Co-Chairs**  
Tim Finin; Susanne E. Hambrusch;  
John Leslie King  
**Board Members**  
Virgilio Almeida; Terry Benzel; Michael L. Best;  
Judith Bishop; Lorrie Cranor; Boi Falting;  
James Grimmelmann; Mark Guzdial;  
Haym B. Hirsch; Anupam Joshi; Richard Ladner;  
Carl Landwehr; Beng Chin Ooi; Francesca Rossi;  
Len Shustek; 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-Neil; Jim Waldo;  
Meredith Whittaker

**CONTRIBUTED ARTICLES**  
**Co-Chairs**  
James Larus and Gail Murphy  
**Board Members**  
Robert Austin; Nathan Baker; Kim Bruce;  
Alan Bundy; Peter Buneman;  
Premkumar T. Devanbu; Jane Cleland-Huang;  
Yannis Ioannidis; Rebecca Isaacs;  
Trent Jaeger; Somesh Jha; Gal A. Kaminka;  
Ben C. Lee; Igor Markov; m.c. schraefel;  
Hannes Werthner; Reinhard Wilhelm;  
Rich Wolski

**RESEARCH HIGHLIGHTS**  
**Co-Chairs**  
Shriram Krishnamurthi  
and Orna Kupferman  
**Board Members**  
Martin Abadi; Amr El Abbadi;  
Animashree Anandkumar; Sanjeev Arora;  
Michael Backes; Maria-Florina Balcan;  
Azer Bestavros; David Brooks; Stuart K. Card;  
Jon Crowcroft; Lieven Eeckhout;  
Alexei Efros; Bryan Ford; Alon Halevy;  
Bernot Heiser; Takeo Igarashi;  
Srinivasan Keshav; Sven Koenig;  
Ran Libeskind-Hadas; Karen Liu;  
Joanna McGrenere; Tim Roughgarden;  
Guy Steele, Jr.; Robert Williamson;  
Margaret H. Wright; Nikolai Zeldovich;  
Andreas Zeller

**SPECIAL SECTIONS**  
**Co-Chairs**  
Jakob Rehof, Haibo Chen, and P. J. Narayanan  
**Board Members**  
Sue Moon; Tao Xie; Kenjiro Taura; David Padua

**ACM Copyright Notice**  
Copyright © 2021 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](mailto: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](http://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 [acmhelp@acm.org](mailto:acmhelp@acm.org).

**COMMUNICATIONS OF THE ACM** (ISSN 0001-0782) is published monthly by ACM Media, 1601 Broadway, 10<sup>th</sup> Floor New York, NY 10019-7434 USA. Periodicals postage paid at New York, NY 10001, and other mailing offices.

**POSTMASTER**  
Please send address changes to *Communications of the ACM*  
1601 Broadway, 10<sup>th</sup> Floor  
New York, NY 10019-7434 USA

Printed in the USA.



Association for Computing Machinery







# CAREER PATHS IN COMPUTING

DOI:10.1145/3453631

Computing enabled me to . . .

## Promote Sustainability and Help Underserved Communities



### NAME

**Caven Cade Mitchell**

### BACKGROUND

**Born in St. Andrew, Jamaica;  
now living in Kawasaki, Japan.**

### CURRENT JOB TITLE/EMPLOYER

**Co-founder and CEO of SIVENTH**

### EDUCATION

**MBA, McGill University, 2020;  
BA, Drexel University, 2005**

**S**USTAINABILITY, OR THE lack of it, is the greatest challenge threatening mankind. I believe protecting natural resources, livelihoods, and the rights of people across the world is a challenge worth fighting for. The United Nations recently issued Sustainable Development Goals to promote solutions for this threat. Computing, through a series of serendipitous events, has enabled me to join this fight, leading me to found my own social enterprise startup, SIVENTH.

The story of how computing brought me to this point begins in Newburgh, once called the “Murder Capital of New York.” I immigrated to America as a child with my parents and became one of the “Dreamers” under DACA. For Jamaicans like me, Newburgh was a place of opportunity, filled with friends and family who

had come before us. Newburgh taught me tenacity and perseverance, which is still a part of my core. I’ve lost a shocking number of friends and family to violence and prison. Those who don’t wind up dead or in jail usually end up in the military or with a local job. A few, like myself, were fortunate enough to make it out.

Through the strength of my parents, in particular my mother, my family bought a computer in 1993 when I was 10 years old. The device sparked my passion for technology with its seemingly infinite capabilities. When I entered high school, I enrolled in every computer class offered: Fortran, QBasic, Visual Basic.

Toward the end of high school, while looking into which military branch to join, a counselor introduced me to the *National Action Council for Minorities in Engineering* Scholarship. I attended Drexel University on this scholarship and received a degree in Information Systems and Technology. Drexel had an amazing co-op program that allowed me to try out roles in three different companies. Because of the program, I began to figure out what I wanted to do long term. After graduating, I explored different roles, always searching for whatever would satisfy my interest in computing and technology.

Eventually, I moved to Japan to work in software development for KDDI Web Communications (KMC). The largest earthquake in Japan’s recorded history occurred just three days after I moved there in 2011. As a result, many local communities were forced to shut down. Wanting to help bring people together, I founded two communities: *DevJapan* for developers and *Design Japan* for designers. My network of contacts grew tremendously, and I eventually became involved

in hosting large tech conferences, such as Nikkei/SUM and Pioneers Asia in Tokyo. I left KWC after six years to join Rakuten as a Tech Evangelist and became even more involved in organizing and speaking at large events and conferences.

These conferences would often put me in the same room as many entrepreneurs, CEOs, and executives from various companies. Conversations with them inspired me to enroll in a Business Administration master’s program at McGill University’s campus in Tokyo. One of my courses focused on how companies could leverage their strategies for developing countries to help the local population while generating greater than expected profits. The professor of this course (Paola Perez-Aleman), along with all my past experiences—working with executives and various startups, growing up in a rough neighborhood, experiencing poverty as a child in Jamaica—drove me to start SIVENTH. While pursuing my MBA, I left Rakuten and founded this social enterprise so the culmination of my experiences could be dedicated to others rather than any one firm’s bottom line.

SIVENTH’s mission is to reduce inequality in underserved communities by leveraging technology to create solutions. Through the development of a sustainability training and simulation tool, we’ve been able to guide corporations toward sustainability by interpreting their industry specific value chains. Additionally, our cloud distribution architecture ensures people everywhere have continuous access to our platform.

I wouldn’t be where I am today if it weren’t for my mother deciding to purchase that first computer so many years ago. Happy Mother’s Day, Mom. **■**

The *Communications* website, <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:10.1145/3453647

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

## Teaching Other Teachers How to Teach CS Better

*Mark Guzdial shares how he assesses the efforts of other computer science teachers.*



**Mark Guzdial**  
**How I Evaluate**  
**a College**  
**Computer Science**  
**Teaching Record**

<http://bit.ly/3kCmzJY>

February 12, 2021

If you spend enough years in academia, you get labelled “senior,” and you find yourself spending a significant amount of time reviewing other academics’ records. Between preparing letters for tenure and promotion cases, and serving on committees at my home institutions, I have read a lot of materials about teaching, research, and service. I evaluate teaching records differently than most of my colleagues. There is a method to what I am doing, and I am sharing it here.

An academic’s teaching recording typically has a personal teaching statement that describes their philosophy of teaching and how they teach their classes. It almost always includes student evaluations of teaching—rankings by students on statements like “This was a good instructor” where 1 might be “strongly disagree” and 5 might be

“strongly agree.” Sometimes the record includes letters from past students, or comments from a colleague who observed classes by the teacher being reviewed. There might also be descriptions of classes created, which is wonderful to do, but it is a different contribution and requires a different skillset than classroom teaching.

I don’t find much value in student evaluations of teaching. First, students do not know what helps them learn the most. From a 2013 study of electrical engineering students using problem-based learning (<https://bit.ly/2NVazYb>) to the 2019 Harvard study of active learning (<https://bit.ly/3uQnXh4>), evidence suggests that students do not know what is best for them. They prefer, and rate more highly, learning opportunities which are easier and less effective. Second, students have implicit bias (<http://bit.ly/3e3UzOe>), such that a white male computer science teacher will tend to get higher student evaluations than colleagues of different races or gender with comparable teaching quality. Students’ written comments in evalu-

ations give me some insights, in the sense of issues to highlight or to be concerned about. But in large classes, the same teaching features may get identified as both positives and negatives. It is hard to find a valuable signal in a large number of anonymous comments. I find student letters to be more valuable. Concerns raised in a signed letter are more likely to be significant.

Some of the common statements in a teaching evaluation have relatively little value for me in identifying teaching quality or skill. A letter writer or peer reviewer might say that the academic “cares about students.” I hope that’s a low bar. It should be the case that everyone teaching cares about students—it is a requisite for the job, not an indication of particular quality. I feel similarly about “has clear and understandable diction.”

Other common statements in teaching statements actually raise concern for me. “Use of active learning” is exactly the kind of statement I’m looking for, unless that’s equated with “is open to student questions” or “encourages class dialogue.” That’s not what I

mean by active learning, and that is not what is meant in the research on active learning methods. I worry when I read that a teacher “is available at any time for students” and “spends many hours meeting students one-on-one in office hours.” Frankly, that suggests to me that the teacher may have poor instructional design skills. Homework in a CS course should be able to be completed based on course content: lectures, discussion, lab, textbook materials, and so on. If the homework problem is so hard that it requires a visit to office hours for most students to complete, then it’s just too hard and that’s unfair to students. First-generation students are less likely to go to office hours (<http://wapo.st/3dYX8kJ>), so it puts them at a disadvantage. If a course is three credit-hours, but students have to go to office hours to finish the assignments, it’s actually a four credit-hour course, and the teacher is demanding too much from the students.

My favorite paper on evaluating undergraduate teaching is Carl Wieman’s article *A better way to evaluate undergraduate teaching* (<https://cwsei.ubc.ca/resources/tools/tpi>). Wieman ([https://en.wikipedia.org/wiki/Carl\\_Wieman](https://en.wikipedia.org/wiki/Carl_Wieman)), a Nobel Prize laureate in Physics and former Associate Director of Science in The White House Office of Science and Technology Policy, is now an Education and Physics Professor at Stanford University. He argues against using student evaluations of teaching for similar reasons to me. He also argues against using peer evaluation of teachers. In general, teachers at the undergraduate level are not taught how to identify high-quality teaching, and likely don’t know what they’re looking for.

He recommends creating an inventory of teaching practices, and computing an “ETP score”—the Extent of use of research-based Teaching Practices. Teachers who use more research-based teaching methods are measurably more effective than teachers who do not. Use of research-based teaching methods makes teaching more effective (for example, the results on peer instruction in computer science classes (<http://peerinstruction4cs.com/>) are amazingly strong), and a teacher who seeks out research-based teaching methods likely has the characteristics of a good teacher—seeking out meth-

## If the homework problem is so hard that it requires a visit to office hours for most students to complete, then it’s just too hard and that’s unfair to students.

ods that serve students’ needs and is humble enough to recognize that their practice could be better.

I don’t compute a score, but I read computer science teaching records actively looking for evidence of seeking and using research-based teaching practices.

- ▶ Any mention of peer instruction, POGIL (Process Oriented Guided Inquiry learning), or even “think-pair-share” counts as active learning for me. I’m looking for activities that engage *all* students (not just those who volunteer) in reflection and collaboration.

- ▶ Use of live coding in class is a good sign. Asking students to make predictions before executing code is a *great* sign.

- ▶ Talking about research-based measures of student success and retention, like self-efficacy, belonging, and intent-to-persist tells me that this is a researcher who is keeping themselves informed about what matters in CS education.

- ▶ I appreciate teachers who recognize how computing education can go wrong, and offer ways to avoid that. Computing classes often have a defensive climate (<https://bit.ly/3qizsdH>), and students get messages that they don’t belong (<https://bit.ly/3sEBsOT>). I look for practices that aim to increase student’s sense of belonging, such as strategies for preventing “show-off” questions in class.

- ▶ Sometimes, it’s the student letters that say things like, “Teacher X asks us a question, and if we don’t get it right, explains it over again.” Use of formative evaluation is a huge positive indica-

tor. (Formative evaluation is hot in the research literature, with a new paper on formative assessment in K–12 CS (<https://bit.ly/3sHAAJd>) at SIGCSE 2021 and a new paper in the *Computer Science Education* journal on effective design of formative feedback (<https://bit.ly/3qaF8Gz>).)

- ▶ Talking about building tools for students and *talking about evaluating them* (especially noting what *did not* work) is another huge positive mark for me. While that might not be using a research-based teaching method, that’s treating teaching innovation as action research (<http://bit.ly/3efbyxl>). (Any *publications* in education conferences or journals is similarly a big sign for me in a positive direction.)

- ▶ Citing any papers from ACM SIGCSE conferences or from IEEE RE-SPECT is an automatic indicator to me of a teacher who looks for better practices and, particularly for RESPECT, is striving to broaden participation in computing.

- ▶ While it’s very rare, I like to see teachers in undergraduate courses adapting practices drawn from K–12 CS. I learn a lot of tips and tricks from AP CS A teachers and from my daughter, a 9<sup>th</sup> grade science teacher. Adaptation from K–12 tells me that the teacher is seeking good practices, and is willing to innovate in their teaching.

In computer science, we often have a model of teaching that is heavily focused on content matter knowledge—if you know your CS, you’ll likely be a good teacher. But all our evidence about quality teaching says that that’s wrong. There’s a lot of knowledge involved in teaching computer science well, and perhaps the most important is *pedagogical content knowledge* (see Chapter 7 in *How People Learn* at <https://www.nap.edu/read/9853/chapter/11>). Teaching is a skill that depends on learned methods. Those who use the best methods tend to teach the best. I value CS teaching that uses the best methods.

My thanks to Leo Porter, Beth Simon, and R. Benjamin Shapiro, who reviewed an early version of this post.

---

**Mark Guzdial** is professor of electrical engineering and computer science in the College of Engineering, and professor of information in the School of Information, of the University of Michigan.

© 2021 ACM 0001-0782/21/5 \$15.00

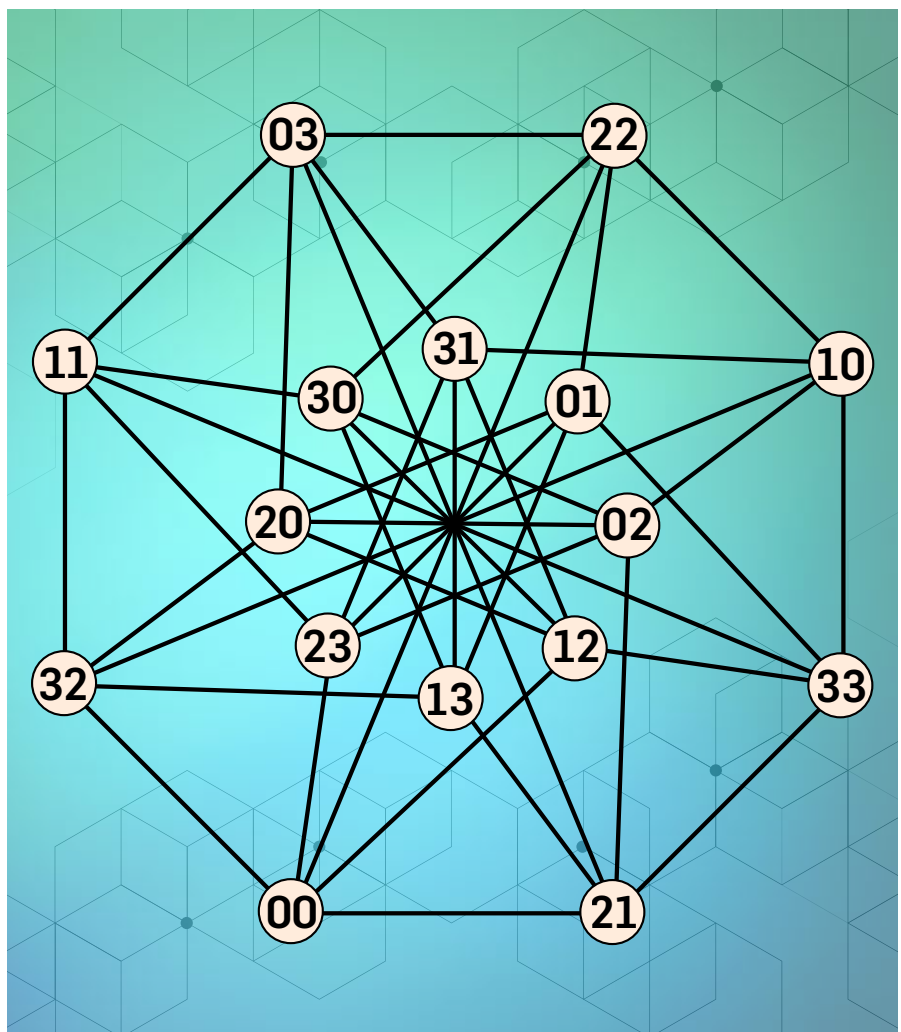
## A Satisfying Result

*Formulating a decades-old geometric conjecture as a satisfiability problem opened the door to its final resolution.*

**A** TEAM OF RESEARCHERS has completed the understanding of the Keller Conjecture, first proposed in 1930, about the packing of squares, cubes, and their higher-dimensional analogues. The conjecture states that any tiling of identical hypercubes that fills space must contain a pair of neighbors that share an entire face.

You might convince yourself it is true for two or three dimensions by toying with squares or blocks. Mathematicians later established the status of the conjecture for all dimensions except seven. The new result, which shared the best paper award at the 2020 International Joint Conference on Automated Reasoning (IJCAR 2020), fills that gap. To do this, the researchers mapped more than 10,324 ways seven-dimensional hypercubes can avoid sharing any six-dimensional face onto a satisfiability problem, which asks whether some Boolean formula can ever be made true.

After exploiting the symmetries of the problem to reduce it to a mere billion or so distinct configurations, the team used extensive computer resources to systematically eliminate them all, implying there must always be a shared face. The resulting proof is enormous, about 200 gigabytes. Obviously, it could not possibly be checked by humans,



A graph representing the two-dimensional case of the Keller Conjecture.

but it produces a certificate that can be checked by other computer programs, and so is almost certainly correct.

“It is satisfying because it’s finishing off an old problem with a venerable history, and math advances like that,” said Jeffrey Lagarias of the University of Michigan, whose work nearly 30 years ago disproved the most general framing of the conjecture. The proof also represents another success for satisfiability-solver algorithms, which are solving problems in many areas of computer science and mathematics.

### A Long History

Mathematician Hermann Minkowski proposed the face-sharing rule for tilings in which every hypercube sits on a periodic lattice. His conjecture was proven in all dimensions in the 1940s. Eduard Ott-Heinrich Keller’s generalization to non-lattice tilings (which includes periodic arrangements of clusters of hypercubes, rather than individual hypercubes) was also proven up to dimension six; higher-dimensional non-lattice tilings, however, remained in play until 1992, when Lagarias and Peter Shor (both then at AT&T Bell Labs) found a non-face-sharing counterexample in 10 dimensions. In a five-page paper, they described a tiling of 10-dimensional hypercubes, none of which share a complete nine-dimensional face.

That result showed that Keller’s original conjecture, which covered all dimensions, was false. Moreover, a non-face-sharing tiling in a particular dimension can be used as a layer in the next-higher dimension, with a simple slide between neighboring layers preventing face-sharing. Thus, once the conjecture was violated in 10 dimensions, it was ruled out for all higher dimensions.

Ten years later, John Mackey, then at Harvard University, found a counterexample for eight dimensions (and thus for nine). Until now, however, mathematicians did not know whether eight was the lowest number of dimensions with a counterexample, or if seven was.

The computer-aided solution built on several intermediate advances. “The interesting question is, how is it that something that appears to be an infinite problem can be reduced to something that can be done by com-

**“It is satisfying because it’s finishing off an old problem with a venerable history, and math advances like that.”**

puter,” said Thomas Hales of the University of Pittsburgh. In 1998, Hales used exhaustive computer checking for a proof of a famous 1611 conjecture by Johannes Kepler concerning the densest packing of spheres.

For Keller’s conjecture, Hales said, “The first reduction that you can do is bring it down to a periodic arrangement.” It turns out that any non-face-sharing example would have to comprise a lattice of clusters that are no larger than 2 in each of the  $d$  dimensions, comprising  $2^d$  cubes.

“That’s not enough to reduce it to a finite problem, because each cube can slide continuously,” noted Lagarias. “You need to do a further reduction so that each cube is just at fixed, discrete positions in space.” It turns out that any non-face-sharing pattern must feature shifts that are integer multiples of  $1/s$ , where  $s$  ranges up to  $d-1$ . “Once there are finitely-many positions, and finitely many cubes, then it’s a finite search problem,” Hales said. Still, he cautioned, “That doesn’t mean that you have a computer that has the capacity to solve the problem.”

For the outstanding seven-dimensional question, Polish mathematician Andrzej Kisielewicz and a coworker, in the past few years, eliminated every possible counterexample except those with  $s=3$ , corresponding to cube shifts by  $1/3$  of their size. To eliminate this final possibility, Mackey, now at Carnegie Mellon University in Pittsburgh, teamed up with a new professor there, Marijn Heule, as well as undergraduate Joshua Brakensiek, now at Stanford University, and visitor David Narváez of the Rochester Institute of Technology. Heule had exten-

# ACM Member News

## CREATING ROBOTS FOR USE IN DISASTERS



Robin Murphy is the Raytheon Professor of Computer Science and Engineering at Texas A&M

University in College Station, TX.

Murphy earned her undergraduate degree in mechanical engineering from the Georgia Institute of Technology (Georgia Tech) in Atlanta, where she also earned her master’s and doctoral degrees, both in computer science. Additionally, she was a Rockwell International Doctoral Fellow.

After receiving her Ph.D. in 1992, Murphy became a professor at the Colorado School of Mines, before moving to the University of South Florida. She joined the faculty of Texas A&M in 2008, where she has remained since.

Murphy’s research is primarily in artificial intelligence for disaster robotics, specializing in human-robot interactions to help them work better together. Her current research is focused on how robots are being used in the response to COVID-19 throughout the world, which she says is telling us a great deal about how robots are adopted for disasters in general.

“Most disaster responses are over in a span of somewhere between three days to two weeks, but the coronavirus timeline is far longer,” Murphy points out. This elongated timeframe provides the opportunity to investigate the maturity of the technology, the influence of trust, and which groups adopt, she adds.

“Instead of just being clinical-care hospitals and the public safety realm, every sector is adopting robots, and using them to cope in some form or fashion,” she says.

Murphy enjoys formalizing what she has learned as she moves from empirical observations to quantitative analysis methods. “It’s been a fun process to develop,” she says.

—John Delaney

sive expertise in large-scale satisfiability proofs, notably the 2016 solution of the “Pythagorean triples problem” which, at 200 terabytes, has the dubious distinction of being the largest proof in history (to date).

### Computer Attack

The 1992 proof by Lagarias and Shor for 10 dimensions was a simple description of a 1,024-cube counterexample. The *non*-existence proof for seven dimensions is vastly harder because it must comprehensively eliminate all of the 128-cube clusters that are not counterexamples.

Lagarias and Shor exploited a graph-theory formulation of the tiling problem that had recently been crafted by Hungarian mathematician Sándor Szabó. He and a colleague described nodes known as “Keller dice,” which bear spots of various colors (depending on  $d$  and  $s$ ) and are connected by an edge in the graph if the colors satisfy certain rules. The existence of a non-face-sharing tiling is equivalent, in seven dimensions, to the existence of a 128-node (27) clique of connected nodes in the corresponding graph.

This graph-theoretic existence question is naturally framed as a satisfiability problem. The required properties of the clique are described by a Boolean formula, like “(die 4 has a red dot OR die 4 has a green dot) AND (die 5 has a black dot),” but much longer. A satisfiability solver, or SAT solver, examines all possible colorings of the dots to determine if any combination can make the Boolean formula true; that is, satisfy it.

Even with today’s powerful computers, it is critical to further reduce the problem by exploiting symmetries; for example, the fact that swapping the labels of two dice does not change anything important. “Many of these graph problems, the main problem of solving them is the symmetries,” Heule stressed. Moreover, experience with SAT-solver competitions has shown that “If there is a bug in a solver, it’s typically because they do some symmetry breaking which is not sound.” For this reason, Heule insisted these arguments be included in the formal published description of the proof.

The remaining problem was big, but easily manageable with modern

**“This technology, formal verification, is far more reliable than the level of verification done by referees for a journal—much more reliable by order of magnitude.”**

supercomputing resources, including the Texas Advanced Computing Center at his former institution, the University of Texas at Austin. “I regularly use 5,000 CPUs,” Heule said, although the Keller work only required about 40.

Although he expects he could access greater resources if needed, Heule also strives to be certain he can finish a job before he starts. “I’ve been investing quite a bit in good runtime estimation,” he said. This is especially important if the formula turns out to be unsatisfiable, which demands examining every possible counterexample. In contrast, Heule said, “For satisfiables, you can get lucky.”

“There are lots of problems that you can translate into satisfiability,” including questions in first-order logic and halting problems like termination of term-rewriting algorithms, Heule said. “More and more fields are seeing the potential of using satisfiability, and that satisfiability is more powerful than any of the dedicated techniques they developed.”

### Trust, But Verify

“One really nice thing about SAT solvers now is that they don’t just give you a yes-no answer in the end that you have to trust,” Hales said. “Even if the proof is very large, a proof certificate is produced, and that can be separately checked.” He adds that “This technology, formal verification, is far more reliable than the level of verification done by referees for a journal—much more reliable by orders of magnitude.”

“This resolution of Keller’s conjecture is the absolutely best possible thing,” agreed Lagarias. “You really wanted a computer-aided proof with a proof checker.”

For number theorist Michael Harris of Columbia University, though, “The issue is not so much whether the reasoning is correct as how it works. If you can’t understand the reasoning, it hardly matters whether it’s correct or not, from the point of view of mathematicians.” Proofs “are not generally end-products, they are stages in a historical process,” he stressed. “An impenetrable proof doesn’t provide what mathematicians are looking for,” Harris said, and he is skeptical of “vague claims that at some point computers are going to be better at doing mathematics than human beings, whatever that means.”

Heule emphasizes the ways computers can augment mathematical insight. For one thing, “I think it’s better to have an answer than no answer, although we don’t understand it.” He also described an eminent mathematician, who told Heule “he wastes about 70% of his time trying to prove something which is false,” effort that could be avoided with technology that finds a counterexample automatically, or that finds the smallest counterexample. “That gives so much insight that it can really help mathematicians do their work faster.” ■

### Further Reading

Hartnett, K.

Computer Search Settles 90-Year-Old Math Problem, *Quanta*, August 2020, <https://bit.ly/2Xh1ni4>

Brakensiek, J., Heule, M.J.H.,

Mackey, J., and Narváez, D.

The Resolution of Keller’s Conjecture <http://bit.ly/3olSXBP>

Hales, T., et al.

A Formal Proof of the Kepler Conjecture, *Forum of Mathematics, Pi* 5, e2 (2017). DOI: <https://doi.org/10.1017/fmp.2017.1>

Lagarias, J.C., and Shor, P.W.

Keller’s cube-tiling conjecture is false in high dimensions *Bull. Amer. Math. Soc.* 27, 279-283. (1992). DOI: <https://doi.org/10.1090/S0273-0979-1992-00318-X>

Don Monroe is a science and technology writer based in Boston, MA, USA.

© 2021 ACM 0001-0782/21/5 \$15.00

# Catching the Fakes

*Applying neural networks to images helps identify counterfeit goods.*

**C**OUNTERFEITING IS A big business. Nearly \$509 billion of fake and pirated products were sold internationally in 2016. In that year, the latest for which data was available, counterfeit goods made up 3.3% of international trade, up from 2.5% three years earlier, according to the Organization for Economic Cooperation and Development.

That figure, which does not include domestic trade in fakes, not only means companies are losing revenue and consumers are not getting their money's worth; counterfeiting also helps fund organized crime. It exploits low-wage laborers. Because it skirts safety regulations, makers of counterfeits could use toxic materials or produce unsafe products.

Now, companies are turning to artificial intelligence (AI) to help them identify counterfeit products and stanch the flow of faux goods.

Amazon, for instance, launched Project Zero in 2019. Makers of products provide the company with data about their logos, trademarks, and other features, and Amazon's machine learning algorithm scans listings on the company's website and automatically removes those it deems fake.

Other companies have rolled out tools to allow retailers to identify fakes using smartphones with scanning attachments.

"Counterfeit goods are among the leading causes of a lot of bad things," says Lakshminarayanan Subramanian, a professor of computer science at New York University and co-founder and chief scientist at Entrupy, a New York-based company founded in 2012 that uses AI to verify the authenticity of luxury goods.

The company's system works with microscopic images of the goods in question, looking for features that are common to an authentic product but not to a fake. Those features could be in the texture of the material, the stitching, a zipper, or the way a logo has been stamped into an item. The leather of a



luxury handbag, for instance, will have what appear to be peaks and valleys when viewed at a microscopic scale.

Entrupy trains its AI starting with images of an area of handbags, both authentic and counterfeit, and breaks that image into smaller chunks. It then applies bag-of-words, a technique developed for natural language processing that sorts words by their frequency in a text and uses that to make inferences about the text. In the case of handbags, the "words" are small areas of structural features that might repeat or vary from place to place across the material. The computer creates a histogram showing the frequency of these visual words and how that frequency differs between real and fake goods.

Based on the results, the company then decides which regions of the product are important for identifying whether the product is authentic, and performs Deep Learning on that data, using a convolutional neural network, noted for its efficiency at image recognition.

Selecting the right features on which to train the neural network turned out to be important in order to reduce the number of false positive results, explains Ashlesh Sharma, a former student of Subramanian's who is now chief technology officer at Entrupy. "If you just threw in a bunch of images to a Deep Learning model, the output at times is completely random. Even the best of the Deep Learning models would have an accuracy of 80% or something," Sharma says.

The company offers a money-back

guarantee if it labels an item as authentic and it turns out to be counterfeit, so 80% or so is not a high-enough accuracy rate. To push that up to greater than 99%, Entrupy will test a handbag on a series of models, using 10 to 12 regions of the bag. That creates a set of probability scores as to whether the bag is authentic. If the average score of all the models is high enough, they label the bag as real; if it is too low, they do not declare the bag a fake, but say they are unable to authenticate it.

## Small Differences

Part of the trick of confirming whether a product is real or counterfeit is not simply to focus on just one type of feature, but to check, say, the quality of the stitching or how the logo looks, as well as the texture of the material. That way, even if a counterfeiter managed to steal a shipment of matching leather, there would still be differences that would give away the fake.

The other important consideration is that the features being examined should be smaller than what a counterfeiter could manufacture, so the company usually focuses on structures between 5 and 10 micrometers in size. "If you can nanofabricate a bag, you might be able to produce similar geometries, but luxury goods in particular are not nanofabricated," Subramanian says. "So it's sort of a challenge between what the magnification at which we capture images is, and what the manufacturing inaccuracy of these specific items is."

Users—usually stores selling luxury

brands such as Prada or Louis Vuitton, or warehouses transferring products from manufacturers to retail outlets—can purchase an attachment for smartphones to enable them to take a series of microscopic images, following a set of instructions from an app on the phone. They upload the images to Entrupy, which checks them against their AI models and delivers a verdict. The whole process takes just minutes, but the company is looking for ways to streamline it.

Entrupy also works with sellers of sneakers, a product sector in which limited editions can sell for hundreds of dollars. Entrupy's technology could be used on other goods that have identifiable features, Sharma says, and they have done some work authenticating expensive watches. The system is less effective with reflective surfaces, such as jewelry or porcelain, because stray light adds artifacts to the image that confounds the AI. Sharma says using infrared light may reduce that issue. The system could also be useful for detecting fake pharmaceuticals or counterfeit money, but Subramanian says the company has not yet ventured into those markets because of the complexity of regulations governing them.

### Working with Light

Adding another layer of data allows IBM's Crypto Anchor Verifier to handle a wider range of products. Like Entrupy, it, too, requires a scanner to be attached to a smartphone, so it also can gather images and compare microscopic features to determine what is real and what is not. In addition, though, it looks at the waveforms of light reflecting from or passing through the object being scanned.

One smartphone attachment has a slot into which a vial containing a liquid or powder can be inserted. Donna Dillenberger, a fellow at IBM's Thomas J. Watson Research Center, demonstrated by displaying two vials of olive oil—one pure, one extra virgin. To the human eye, they are nearly identical yellow fluids. She slips one into the holder, and in a few seconds the phone's screen displays a graph describing the wavelengths of passing through it. She replaces it with the other, and a very different graph appears. "Even though to the human eye the liquids look the same, to the Verifier they look different," she says. "We've done the same things to

## Like Entrupy, IBM's product relies on convolutional neural networks to train the computer to distinguish features of actual products from those of fakes.

authenticate wines, between expensive and inexpensive wines."

The Verifier also can distinguish chemotherapy drugs, or discover whether expensive cosmetics have been adulterated with cornstarch. The work on powders was prompted by a scandal in 2008, when powdered milk and baby formula from China was found to have been laced with melamine, which was used to dilute milk to stretch sales, and wound up sickening children.

Like Entrupy, IBM's product relies on convolutional neural networks to train the computer to distinguish features of actual products from those of fakes. Because wines and oils are easy to differentiate, the algorithm needed only a handful of samples, in the "low double-digits," to train on, Dillenberger says. For more complicated tasks, such as grading diamonds by how they refract light, she said, thousands of samples were necessary.

The combination of imaging and AI also can be applied to integrated circuits and circuit boards, says Domenic Forte, a professor of electrical and computer engineering at the University of Florida. Forte trained a neural network to identify physical defects that reveal if a chip has been altered, such as scratches or variations in surface texture that come from resurfacing a chip or changing its marking. "If the counterfeiter changes the marking from commercial to industrial grade, they could sell you a part that's much cheaper and it won't withstand what your application requires," Forte says.

Visual inspection of ICs currently is performed manually, but Forte hopes

to automate the process. Using infrared imaging allows the system to see through a part's packaging and look for alterations underneath. There are electrical tests that can be used to verify parts, but they are often specific to a part, making them more difficult to automate, and they can require expensive equipment, Forte says. Sometimes they even require packaging to be removed, destroying the part in the process.

For all of AI's ability to catch fake goods, Dillenberger wonders whether there might not be a larger societal question that is going unaddressed. "You could build these technologies to catch things, but how could we nudge people, who must be extremely talented to even make counterfeits in the first place, to do something else with their gifts?" she asks. "Why aren't they being rewarded to make something that's not counterfeited? How can technology help at that systemic problem?"

### Further Reading

Asadizanjani, N., Tehranipoor, M., and Forte, D. Counterfeit Electronics Detection Using Image Processing and Machine Learning, *J. of Physics: Conf. Series* 787 (2017) <https://iopscience.iop.org/article/10.1088/1742-6596/787/1/012023>

Sharma, A., Srinivasan, V., Kanchan, V., and Subramanian, L. The Fake vs Real Goods Problem: Microscopy and Machine Learning to the Rescue, *KDD '17*, (2017) <https://dl.acm.org/doi/10.1145/3097983.3098186>

Balagurusamy, V., Siu, V., Kumar, A.D., Dureja, S., Ligan, J., Kudva, P., Tong, M., and Dillenberger, D. Detecting and discriminating between different types of bacteria with a low-cost smartphone based optical device and neural network models, *SPIE 2019 Nanoscience and Engineering Conference Proceedings*, (2019) <https://spie.org/Publications/Proceedings/Paper/10.1117/12.2529829?SSO=1>

Cheung, M., She, J., and Liu, L. Deep learning-based online counterfeit-seller detection, *IEEE Conf. on Computer Communications Workshops*, (2018) <https://ieeexplore.ieee.org/document/8406896>

Entrupy Authentication Demo Video <https://www.youtube.com/watch?v=HMXQJK7v9t8>

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



# A Traffic Cop for Low Earth Orbit

*Who will be the space traffic controller for orbiting objects?*

**O**N EARTH, AVOIDING collisions is a key priority for traffic cops, air traffic controllers, and the parents of toddlers. It is no different in space—and perhaps even more critical—given that objects orbiting the Earth are moving at more than 17,000 m.p.h., which means that even very small objects less than a centimeter in diameter have caused damage to the International Space Station, the Space Shuttle, and satellites.

In fact, the U.S. National Aeronautics and Space Administration (NASA) estimates there are more than 500,000 such objects orbiting the Earth that are larger than a marble, and at least a million smaller pieces of debris that cannot be tracked. Based on the growing number of commercial and government launches of spacecraft, satellites, and even space stations, the number of objects that will need to be catalogued, tracked, and managed is expected to grow significantly in the coming years. And the solutions to this issue are fraught with both technical and political challenges.

A key organization charged with

tracking and notifying the operators of space objects of a potential course collision is the 18<sup>th</sup> Space Control Squadron (SPCS), part of the U.S. Air Force's Space Surveillance Network (SSN), which currently tracks 27,000 objects in low Earth orbit (up to 2,000km above the Earth's surface), medium Earth orbit (2,000km to 3,600km up), and geosynchronous equatorial orbit (located 35,786km above the Earth).

The 18<sup>th</sup> SPCS is co-located with the Combined Space Operations Center at Vandenberg Air Force Base in Santa Barbara County, CA, and is charged with handling conjunction assessments (the close physical encounters of two tracked objects in space), collision avoidance, and reentry assessment for satellites and other objects returning to Earth from space.

Object tracking is handled via a network of sensors and telescopes located around the world and in space, and utilizes two-line element (TLE) position datasets (which incorporate the parameters required to uniquely identify an orbiting element at a given point in time), which can then be fed into models that take into account atmospheric

drag, gravitational drag caused by the Earth's shape, the Earth's spin, and various other factors in order to predict the motion of each orbiting object over time.

The Space Surveillance Network also incorporates the Haystack Ultrawideband Satellite Imaging Radar (HUSIR), which is located in the Lincoln Laboratory of the Massachusetts Institute of Technology (MIT). HUSIR, the highest-resolution, longest-range sensor in the world, simultaneously generates X- and W-band images that can provide valuable information about the size, shape, and orientation of Earth orbiting objects. HUSIR was incorporated into the SSN in 2014.

In March 2020, additional object tracking capability known as the Space Fence, designed to provide higher resolution for tracked objects, was declared operational. Located on Kwajalein Island in the Republic of the Marshall Islands near the Equator in the Pacific Ocean, the Space Fence uses an S-band radar system to track objects primarily in low Earth orbit (though it is also capable of tracking objects in higher orbits), and allows the tracking of ob-



IMAGE BY DOTTED YETI. USING ELEMENTS FURNISHED BY NASA

jects below the previous size limitation of 10cm.

While the Space Fence technology permits greater visibility of smaller objects orbiting the Earth, it requires a significant amount of compute power to project the paths of these objects, according to Mike Gruntman, a professor of Astronautics at the University of Southern California's School of Engineering.

"When the number of catalogued objects increases by a factor of four, you need an increase in computational power by a factor of 16," Gruntman says, noting that commercial operators increasingly are launching a greater number of objects into space, and they're not necessarily building them to a high standard of reliability, due to cost. This can create a situation where satellites malfunction, and then either fall out of orbit back to Earth, or are no longer able to be maneuvered, increasing the risk of a collision with another object in space. "So, we'll have many more satellites presenting a challenge for many decades, until we figure out how to manage it, and the management is not easy because you cannot tell a certain country, 'you cannot launch a satellite.'"

Gruntman adds that simply adding more compute power, while helpful, is not yet able to overcome the accuracy constraints of existing sensors which cannot account for atmospheric drag

and solar radiation, introducing a high degree of variability in an orbiting object's path, as well as the time and trajectory of an object that is expected to fall out of orbit and return to earth. Ultimately, "more frequent observations by multiple sites will help to continuously correct the predictions," according to Gruntman. However, he adds that no amount of compute power can address the increasing number of owner/operators who launch objects into space, but choose not to report or share their trajectories to tracking authorities.

This is exacerbated by the lack of a single overriding international authority that would serve as space's equivalent of air traffic control. While the United Nations Office for Outer Space Affairs (UNOOSA) promotes international cooperation in outer space, it does not have any recognized authority to enforce regulations at the international level.

Within the U.S., a 2018 White House directive appointed the Commerce Department to serve as the traffic cop for space, though other agencies are likely to continue to have some role in the management of space traffic. The Federal Aviation Administration (FAA) regulates launches and reentries, and the Federal Communications Commission (FCC) is responsible for regulating satellite transmissions.

The U.S. Department of Defense,

meanwhile, perhaps has the most active role in tracking and managing objects in orbit through the 18<sup>th</sup> SPCS, though it relies heavily on cooperation between parties, including other space authorities such as NASA, the European Space Agency, and Russia's Roscosmos State Corporation for Space Activities, as well as cooperating with commercial space companies, rather than specific inter-country or worldwide authorities.

"The 18<sup>th</sup> Space Control Squadron provides collision avoidance services, which is a free service provided to satellite owner/operators on behalf of the United States Government through the United States Space Command," explains Lt. Col. Justin Sorice, the commander of the 18<sup>th</sup> SPCS. "The 18<sup>th</sup> Space Control Squadron sent out almost 20 million conjunction data messages this year, informing satellite owner/operators how close their systems are approaching other space objects in order to make a maneuver decision."

The 18<sup>th</sup> SPCS currently shares Space Domain Awareness information with more than 100 governmental, academic, and commercial partner organizations from 25 nations through formal Space Situational Awareness data-sharing agreements, and via a publicly accessible website, [www.Space-Track.org](http://www.Space-Track.org). It is this data sharing and cooperation that is enabling U.S. government

## ACM News

# AI vs. AI: The Race to Detect Deepfakes

Distinguishing between fact and fakery has become an everyday part of our online lives.

Artificial intelligence (AI) can generate astonishingly realistic false images and videos, and it increasingly is being used to detect them.

Researchers from Intel and the Graphics and Image Computing (GAIC) laboratory of Binghamton University developed a tool called FakeCatcher that identifies fake portrait videos via biological signals. The team reported 99.39% accuracy in pairwise separation (pairs of videos in which one is real and one is fake), and 96% accuracy in deepfake detection.

"When people experience

different emotions, they experience different physiological signals, those signals are captured in our lab," said GAIC director Lijun Yin.

Ilke Demir, a senior research scientist at Intel, realized the potential of the resource for deepfake detection and collaborated with one of Yin's Ph.D. students, Umur A. Ciftci. As Yin explained, "People can synthesize a facial video, an animated facial expression, but it's very hard to synthesize their physiological signals."

When blood flow and heart rate changes are triggered by emotions, they manifest on video as color changes in pixels.

The changes are invisible to the human eye, but can be detected using photoplethysmography (PPG). Under Yin's supervision, Ciftci and Demir developed an algorithm that identifies PPG signals and converts them into a PPG map that can be used to detect deepfakes.

In Germany, a team at Horst Görtz Institute for IT Security at Ruhr-Universität Bochum developed a technique for detecting deepfakes using frequency analysis. The researchers wrote a program that automates discrete cosine transform (DCT) analysis on images sourced from the Which Face is Real? website, and a

dataset of bedroom images collated specifically for the project.

The team showed that unlike their real counterparts, GAN-generated deepfakes display information traces in high-frequency areas. These traces can now be easily identified and used to flag an image as a fake.

"In the high frequency, it's very prominent because real pictures don't have much information there and they [deepfakes] actually have quite a bit," said research assistant Joel Frank.

—Karen Emslie is a freelance journalist and essayist.

agencies tasked with tracking objects to work with foreign governments and commercial operators to identify and track satellites, rockets, and other items being launched into space.

“One of the most valuable things we can do is what we call early engagement, making contact with new owner-operators, whether they’re commercial, civil, or whatnot, and really understanding what their mission objectives are,” says Diana McKissock, SSA Sharing & Spaceflight Safety Lead for the 18th SPCS. “The DoD does not have the authority to manage space traffic; rather, we provide spaceflight safety data that helps satellite owners/operators make responsible decisions to mitigate risk to their spacecraft and the space environment. A significant challenge for the space community is the lack of agreed-upon norms of behavior or enforceable global regulations, which could provide a true space traffic management construct.”

Communication and cooperation with commercial and government owner-operators of space objects will be key to limiting future space collisions between objects, given the growing number of conjunctions each year. “In 2019, there were about 328 close conjunctions with the International Space Station (ISS),” Sorice said, adding that during 2020 through the end of November, “we’ve had 460. So again, the space domain is becoming a lot more congested as we go forward.”

McKissock says for objects in low Earth orbit, emergency reportable criteria, which allows the 18<sup>th</sup> SPCS to share information with the owner or operator of any active satellite, specifies the time of closest approach between two objects within a 72-hour period, with a miss distance of one meter, and a probability of collision of greater than 1 in 10,000. The criteria for deeper orbits expands the miss distance to less than 500 meters. In the absence of an international space regulator, cooperation between commercial and government operators is key to eliminating potentially catastrophic collisions.

“The policies that we work under allow us to share information with the owners and operators of any active satellite, which allows us to work closely with a lot of foreign civil and commercial entities who are willing to com-

**“One of the most valuable things we can do is what we call early engagement, making contact with new owner-operators ... and really understanding what their mission objectives are.”**

municate with us,” McKissock says. “Right now, of the nearly 3,500 active satellites, I think we have positive communication with about 95% of those owner-operators.”

The desire to reduce or eliminate collisions cannot be overstated. While there is being work done to improve the accuracy of calculating the physical impact of objects colliding in space, Gruntman says that experimental databases are limited, and major uncertainties are likely to remain because in the event of a collision between a piece of space debris and a satellite or spacecraft, “the debris/fragment field significantly depends on the design of particular spacecraft.” McKissock adds that while increasing the available compute power also may aid in predicting where and when satellites may return to Earth, “having accurate information on the reentering spacecraft, as well as a reliable atmospheric density model, are also key contributors to accurate reentry predictions.”

To assist in the analyses of the potential conjunctions of such a large (and growing) number of objects, the 18<sup>th</sup> SPCS increasingly is deploying machine learning technology to automate the tracking of objects, as well as the prediction of conjunctions between objects, so conjunction assessments can be managed without overwhelming human resources. Orbital altitude, speed, and course trajectory data from satellites and other objects orbiting

the Earth (or falling out of orbit) are fed into an algorithm that can learn and predict future close conjunctions.

However, it is not just governmental agencies that may play a role in managing space traffic. Canadian company NorthStar Earth & Space is planning to launch a constellation of 12 satellites equipped with optical sensors to monitor space objects in low Earth orbit, with the initial launch of three satellites next year, and the remainder by 2024. The company likely will use a subscription-based revenue model, charging organizations to access traffic and collision data. However, as it is a commercial entity, it is unlikely any operator that did not wish to move its equipment in orbit could be compelled to do so.

Still, the biggest issue for any organization tracking objects in space is not managing active satellites or equipment in orbit; there’s no benefit to having something smashed to bits if it’s at all possible to reposition a satellite or space station. Space debris or junk that cannot be controlled is the primary concern, given the potential chain-reaction of a collision, Sorice says.

“When we’re talking about thousands of pieces of debris moving at 17,000 miles per hour and potentially creating chain effects within that orbital regime, that’s what keeps me up at night, and making sure that we do our best to prevent this scenario,” Sorice says. “Because those pieces of debris don’t have political agendas, they’re just simply following the laws of physics at that point in time. You need to keep up calculational power to be able to predict where those pieces may be going, and what assets are at threat.”

#### Further Reading

Space Debris and Human Spacecraft, National Aeronautics and Space Administration, Sept. 26, 2013, [https://www.nasa.gov/mission\\_pages/station/news/orbital\\_debris.html](https://www.nasa.gov/mission_pages/station/news/orbital_debris.html)

Real-time object tracking data [www.Space-track.org](http://www.Space-track.org)

The Truth About Space Debris, Real Engineering, <https://www.youtube.com/watch?v=itdYS9XF4a0>

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

© 2021 ACM 0001-0782/21/5 \$15.00

► Terry Benzel, Column Editor

# Security

## Trustworthy Scientific Computing

*Addressing the trust issues underlying the current limits on data sharing.*

**D**ATA USEFUL TO science is not shared as much as it should or could be, particularly when that data contains sensitivities of some kind. In this column, I advocate the use of hardware trusted execution environments (TEEs) as a means to significantly change approaches to and trust relationships involved in secure, scientific data management. There are many reasons why data may not be shared, including laws and regulations related to personal privacy or national security, or because data is considered a proprietary trade secret. Examples of this include electronic health records, containing protected health information (PHI); IP addresses or data representing the locations or movements of individuals, containing personally identifiable information (PII); the properties of chemicals or materials, and more. Two drivers for this reluctance to share, which are duals of each other, are concerns of data owners about

**Hardware trusted execution environments can form the basis for platforms that provide strong security benefits while maintaining computational performance.**

the risks of sharing sensitive data, and concerns of providers of computing systems about the risks of hosting such data. As barriers to data sharing are imposed, data-driven results are

hindered, because data is not made available and used in ways that maximize its value.

And yet, as emphasized widely in scientific communities,<sup>3,5</sup> by the National Academies, and via the U.S. government's initiatives for "responsible liberation of Federal data," finding ways to make sensitive data available is vital for advancing scientific discovery and public policy. When data is not shared, certain research may be prevented entirely, be significantly more costly, take much longer, or might simply not be as accurate because it is based on smaller, potentially more biased datasets.

Scientific computing refers to the computing elements used in scientific discovery. Historically, this has emphasized modeling and simulation, but with the proliferation of instruments that produce and collect data, now significantly also includes data analysis. Computing systems used in science include desktop systems and clusters run by individual



investigators, institutional computing resources, commercial clouds, and supercomputers such as those present in high-performance computing (HPC) centers sponsored by U.S. Department of Energy's Office of Science and the U.S. National Science Foundation. Not all scientific computing is large, but at the largest scale, scientific computing is characterized by massive datasets and distributed, international collaborations. However, when sensitive data is used, computing options available are much more limited in computing scale and access.<sup>8</sup>

### Current Secure Computing Environments

Today, where remote access to data is permitted at all, significant technical and procedural constraints may be put in place, such as instituting ingress/egress "airlocks," requiring "two-person" rules to move software and data in or out, and requiring the use "remote desktop" systems. Archi-

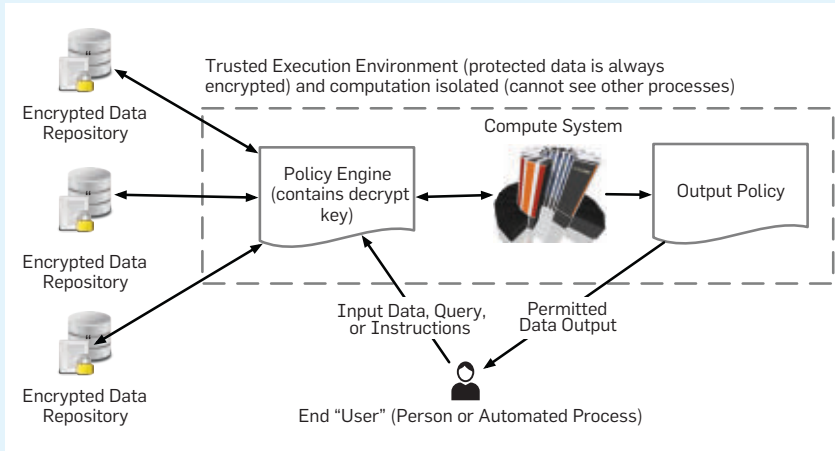
tectures like this are becoming more and more common as means for scientific computing involving sensitive data.<sup>8</sup> However, even with these security protections, traditional enclaves still require implicitly trusting system administrators and anyone with physical access to the system containing the sensitive data, thereby increasing the risk to and liability of an institution for accepting responsibility for hosting data. This security limitation can significantly weaken the trust relationships involved in sharing data, particularly when groups are large and distributed. These concerns can be partially mitigated by requiring data analysts to be physically present in a facility owned by the data provider in order to access data. However, in all these cases, analysis is hindered for the scientific community whose abilities and tools are optimized for working in open, collaborative, and distributed environments. Further, consider the current pandemic in which a requirement of

physical presence in a particular facility for analysis would be a public health risk.

### Reducing Data Sensitivity Using "Anonymization" Techniques

Sometimes attempts are made to avoid security requirements by making data less sensitive by applying "anonymization" processes in which data is masked or made more general. Examples of this approach remove distinctive elements from datasets such as birthdates, geographical locations, or IP network addresses. Indeed, removing 18 specific identifiers from electronic health records satisfies the HIPAA Privacy Rule's "Safe Harbor" provisions to provide legal de-identification. However, on a technical level, these techniques have repeatedly been shown to fail to preserve privacy, typically by merging external information containing identifiable information with quasi-identifiers in the dataset to re-identify "anonymized" records.<sup>6</sup> Therefore,

A portion of a system leveraging a trusted execution environment in which data is stored encrypted on disk; a policy engine, controlled by the data owner, and running in the TEE, contains the mapping for what data is to be made available for computing by each authenticated user; and an output policy, also specified by the data owner, dictates what information is permitted to be returned to the user. An output policy might be based on differential privacy, or be access-control based, or be some combination of these or other functions.



de-identification does not necessarily address the risk and trust issues involved in data sharing because re-identification attacks can still result in significant embarrassment, if not legal sanctions. In addition, the same masking used in these processes also removes data that is critical to the analysis.<sup>6</sup> Consider public health research for which the last two digits of a ZIP code, or the two least significant figures of a geographic coordinate are vital to tracking viral spread.

### Confidential Scientific Computing

Hardware TEEs can form the basis for platforms that provide strong security benefits while maintaining computational performance (see the accompanying figure). TEEs are portions of certain modern microprocessors that enforce strong separation from other processes on the CPU, and some can even encrypt memory and computation. TEEs have roots in the concepts of Trusted Platform Modules (TPMs) and Secure Boot, but have evolved to have significantly greater functionality. Common commercial TEEs today include ARM's TrustZone, introduced in 2013; Intel's Secure Guard Extensions (SGX), introduced in 2015; and AMD's Secure Encrypted Virtualization (SEV), introduced in 2016 and revamped several times since then to include SEV-ES (Encrypted State) in 2017 and SEV SEV-SNP (Secure

Nested Paging) in 2020. All three vendors take extremely different approaches and have extremely different strengths, weaknesses, use cases, and threat models.

TEEs can be used to maintain or even increase security over traditional enclaves, at minimal cost to performance in comparison to computing over plaintext. TEEs can isolate computation, preventing even system administrators of the machine in which the computation is running from observing the computation or data being

**Trusted execution environments can be used to maintain or even increase security over traditional enclaves, at a minimal cost to performance in comparison to computing over plaintext.**

used or generated in the computation, including even from certain "physical attacks" against the computing system. They can implement similar functionality as software-based homomorphic and multiparty computation<sup>2</sup> approaches, but without the usability issues and with dramatically smaller performance penalties.

The use of TEEs to protect against untrustworthy data centers is not a novel idea, as seen by the creation of the Linux Foundation's Confidential Computing Consortium<sup>10</sup> and Google's recent "Move to Secure the Cloud From Itself."<sup>7</sup> Google has comparing the importance of the use of TEEs in its cloud platform to the invention of email.<sup>9</sup> However, TEEs have not yet seen broad interest and adoption by scientists or scientific computing facilities.

The envisioned approach is to leverage TEEs when data processing environments are out of the direct control of the data owner, such as in third-party (including DOE or NSF) HPC facilities or commercial cloud environments, in order to prevent exposure of sensitive data to other users of those systems or even the administrators of those systems. Data providers can specify the configuration of the system, even if they are not directly the hosts of the computing environment, to specify access control policies, a permitted list of software or analyses that can be performed, and output policies to prevent data exfiltration by the user. The notion of being able to leverage community HPC and cloud environments also enables the use of data from multiple providers simultaneously while protecting the raw data from all simultaneously, each potentially with their own distinct policies.

Researchers at the Berkeley Lab and UC Davis have been empirically evaluating Intel SGX and AMD SEV TEEs for their performance under typical HPC workloads. Our results<sup>1</sup> show that AMD's SEV generally imposes minimal performance degradation for single-node computation and represents a performant solution for scientific computing with lower ratios of communication to computation. However, Intel's SGX is not performant at all for HPC due to TEE

memory size limitations. Importantly, NERSC-9 and a number of other modern HPC centers will contain AMD processors that support the SEV TEE, and thus it is our hope that our results will provide some of the evidence needed to justify the use of TEEs in scientific computing.

### Looking to the Future

Although numerous commercial TEEs exist, no TEEs yet exist in processors other than CPUs, such as in GPUs and accelerators, although Google has indicated that it plans to expand “Confidential Computing” to GPUs, TPUs, and FPGAs.<sup>9</sup> There are also issues with low-latency communication between TEEs, and also the cost of virtualization, that must be addressed to enable HPC at scale.<sup>1</sup> In addition, promising RISC-V efforts such as Keystone<sup>4</sup> exist that carry both the promise of broadening the scope of processors that contain TEEs, while also being open source and possible to formally verify. However, RISC-V based TEEs have not yet been developed that target scientific computing. Most likely, an entirely new TEE architecture tailored for scientific computing and data analysis applications will be needed.

Output policies are another area that deserve investigation. While TEEs protect against untrusted computing providers, and can provide certain measures of protection from malicious users, output policies determine what data is returned to the user. Differential privacy is a particularly interesting approach to providing strong privacy protection of data output. Differential privacy is a statistical technique that can guarantee the bounds on the amount of information about a dataset that can be leaked to a data analyst as a result of a query or computation by adding “noise” and enforcing a “privacy budget” that bounds information leakage. It is now a mainstream solution, with production use by Apple, Google, and the U.S. Census Bureau, the existence of several open source distributions, and successful application to a diverse range of data types. However, differential privacy is not appropriate everywhere, and applying it is currently challenging, requiring a high degree of expertise and effort. Thus, differential privacy is


## Trusted execution environments enable sensitive data to be leveraged without having to trust system administrators and computing providers.

highly useful today, albeit in a limited set of situations for datasets that have sufficiently wide use to justify the time and expense required. Work is needed to advance the usability of differential privacy so it can more easily be broadly leveraged.

### Summary and Next Steps

In contrast to traditional secure enclaves, TEEs enable sensitive data to be leveraged without having to trust system administrators and computing providers. However, while the application of TEEs has now been widely heralded in cloud environments, TEEs have not been discussed for use in scientific computing environments, despite the significant concerns frequently expressed by both data providers and computing facilities about hosting sensitive data. Operators of scientific computing facilities are notoriously conservative for good reason—they are frequently evaluated on the degree of utilization and amount of uptime of the systems they run, and so the margin for error is low. But TEEs are here, they are available, and until we start making use of them in scientific computing, data is not shared as much as it should or could be by leveraging TEEs to address the trust issues underlying current limits on data sharing.

What is missing is a connection to the particular infrastructure used in scientific computing, including identity, access, and authentication systems; remote direct memory access (RDMA);

batch scheduling systems in HPC; HPC I/O subsystems; custom scientific workflows; highly specialized scientific instruments; community data repositories, and so on. Therefore what is needed is a conversation between processor manufacturers, system vendors (for example, Cray, HPE), and scientific computing operators regarding enabling the TEE functionality already present in the AMD EPYC processors—and presumably in other, future processors—into scientific computing environments. However, the path forward is not solely technical. It requires the community to build infrastructure around TEE technology and integrate that infrastructure into scientific computing facilities and workflows, and into the mind-set of operators of such facilities. I hope this column helps to start that conversation. For more on TEEs, see the Singh et al. article on p. 42. —Ed. 

### References

1. Akram, A. et al. Performance analysis of scientific computing workloads on trusted execution environments. In *Proceedings of the 35th IEEE International Parallel & Distributed Processing Symposium*. (2021).
2. Choi, J.I. and Butler, K. Secure multiparty computation and trusted hardware: Examining adoption challenges and opportunities. *Security and Communication Networks* 1368905 (2019).
3. Hastings, J.S. Unlocking data to improve public policy. *Commun. ACM* 62, 10 (Sept. 2019), 48–53.
4. Lee, D. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys)* (Heraklion, Greece, 2020), Article 38.
5. Macfarlane, J. When apps rule the road: The proliferation of navigation apps is causing traffic chaos. It's time to restore order. *IEEE Spectrum* 56, 10 (2019), 22–27.
6. Narayanan, A. and Felten, E.W. No Silver Bullet: De-identification Still Doesn't Work. (2014); <https://bit.ly/3loBvMd>
7. Newman, L.H. Google moves to secure the Cloud from itself. *Wired* (July 14, 2020); <https://bit.ly/2NnLLru>
8. Peisert, S. An examination and survey of data confidentiality issues and solutions in academic research computing. *Trusted CI Report* (2020); <https://bit.ly/300ajx9>
9. Potti, S. and Manor, E. Expanding Google Cloud's Confidential Computing portfolio. (2020); <https://bit.ly/38Nozuo>
10. Rashid, F.Y. The rise of confidential computing. *IEEE Spectrum* 57, 6 (2020), 8–9.

**Sean Peisert** (speisert@lbl.gov) leads computer security R&D at Lawrence Berkeley National Laboratory and is an Associate Adjunct Professor at University of California, Davis, USA.

The author thanks Venkatesh Akella, Ayaz Akram, Jim Basney, Jason Lowe-Power, and Von Welch for their valuable feedback on this Viewpoint and the ideas in it. This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy, and by Contractor Supported Research (CSR) funds provided by Lawrence Berkeley National Laboratory, operated for the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors of this work.

Copyright held by author.

► James Grimmelmann, Column Editor

## Law and Technology

# Software Professionals, Malpractice Law, and Codes of Ethics

*In pursuit of professional status for computing professionals.*

**W**E ALL KNOW what a professional is—or do we? For years, ACM has proclaimed that its members are part of a computing *profession*. But is it really a profession? Many people describe themselves as “professionals” in the colloquial sense of being paid to perform some specialized skill. Yet, only a few occupations are regarded as professions in the *legal* sense. Courts do not consider athletes or chefs to be professionals the way doctors and lawyers are. Likewise, courts have consistently excluded software developers from that select group.

To understand why U.S. law does not recognize computing as a profession—and whether that classification could be changed—calls for a fresh look at the law of professions. Why does the law distinguish professionals from *non*professionals such as mechanics or pilots? What would happen if courts treated software developers like doctors or lawyers? What are professionals’ legal duties of care and how do they differ from ethical codes of conduct? Can one bootstrap the other?

U.S. tort law draws a clear distinction between professionals and nonprofessionals. When nonprofessionals cause injury to others, they can be sued for ordinary negligence. Their conduct is compared to that of the



“ordinary reasonable person.” For example, when a homebuyer sues a home inspector for failing to disclose a material defect, the jury must decide whether the inspector acted with reasonable care. Defendants who fail that common-sense standard must pay damages for the harm they caused.

Evidence of industry custom can be considered, but the jury is free to override it based on the jurors’ own personal understanding of how reasonable people are expected to behave. Thus, even if the inspector followed

an industry-standard checklist, the jury could find the checklist itself to be deficient. That rule prevents an entire industry from hiding behind an obviously negligent custom.

By contrast, professionals like doctors and lawyers are subject to malpractice claims. The relevant metric is the custom of the profession—not the ordinary reasonable person or even the ordinary reasonable professional. Under this “customary care” standard, the jury is restricted to evaluating whether the defendant complied with



the profession's internal customs and norms. In a medical malpractice case, what matters is whether the doctor followed practices accepted by other peers in the same field, not whether the jury feels those customary practices were reasonable. Juries are not free to substitute their own views on how the professional should have acted.

Most legal scholars agree the list of professions includes medicine and law. Beyond that there is little consensus as to who else qualifies. A scattering of court decisions have allowed accountants, architects, engineers, social workers, or even sports coaches. One leading legal authority, the Second Restatement of Torts, recommends extension to "skilled trades" such as airplane pilots, precision machinists, electricians, carpenters, blacksmiths, and plumbers.<sup>a</sup> A more recent version, the Third Restatement of Torts, argues in favor of adding insurance agents but not construction contractors.<sup>b</sup> Despite the many candidates, there is still great uncertainty as to which occupations should be treated as professions, and why.

The computing industry is not on any of those lists. On the contrary, every court to consider this question has refused to recognize software developers as professionals. The leading case is *Hospital Computer Systems, Inc. v. Staten Island Hospital*,<sup>c</sup> in which the court recited a long list of traits that the professional possesses but that the software developer lacks. The judge wrote: "A profession is not a business. It is distinguished by the requirements of extensive formal training and learning, admission to practice by a qualifying licensure, a code of ethics imposing standards qualitatively and extensively beyond those that prevail or are tolerated in the marketplace, a system for discipline of its members for violation of the code of ethics, a duty to subordinate financial reward to social responsibility, and, notably, an obligation on its members, even in nonprofessional matters, to conduct themselves as members of a learned, disciplined, and honorable occupation."<sup>d</sup>

a Restatement (Second) of Torts § 299A (1965).  
 b Restatement (Third) of Torts: Liability for Economic Harm § 4 (2020).  
 c 788 F. Supp. 1351 (D.N.J. 1992).  
 d Id. at 1361.

## Most legal scholars agree the list of professions includes medicine and law. Beyond that there is little consensus as to who else qualifies.

As one leading computer law treatise summarized: "Computer programmers commonly define themselves as 'professionals.' Yet, despite the complexity of the work, computer programming and consultation lack the indicia associated with professional status."<sup>e</sup>

At first sight, those cases seem to imply that meeting the professional indicia will provide an automatic pathway to professional status. If one accepts that premise, then it makes sense to push the computing industry to check the appropriate boxes. Thus, many universities have sought to formalize accreditation standards for computer science and engineering degrees. Some members of the community have advocated for state licensure and professional exams for software engineering.

That pursuit of professional status also explains why ACM approved major revisions to its Code of Professional Conduct in 1992—the same year that *Hospital Computer Systems* was decided. The authors who led the revisions explained that ACM was "tak[ing] a new direction" in order to achieve greater "consensus and commitment of its members to ethical behavior."<sup>1</sup> Those efforts, in turn, would "help persuade the public that professionals are deserving of its confidence and respect, and of increased social and economic rewards."<sup>e</sup> With the latest revisions in 2018, ACM has again proclaimed that "computing is a profession," and that

e See Anderson et al.,<sup>1</sup> quoting Mark S. Frankel, Professional Codes: Why, How, and with What Impact? *J. Bus. Ethics* 8, 109 (1989).

members should uphold "the responsibilities the profession has to the larger society that it serves."<sup>2</sup>

But who generated this list of professional traits? Are we sure that formal education, licensure bars, and codes of ethics properly distinguish professionals from nonprofessionals? After all, although countless occupations have faithfully pursued this formula, courts remain reluctant to extend the professional designation much beyond the traditional domains of medicine and law.

In fact, this list of professional traits can be traced back to early work in organizational sociology. Early 20<sup>th</sup>-century sociologists sought to identify the key traits that separated the "gentlemanly" professions—such as medicine, law, and clergy—from other, common-class occupations. That work produced lists of traits closely resembling the indicia described in the *Hospital Computer Systems* case. Yet, later critics condemned the trait-based approach as being "essentially atheoretical"<sup>7</sup> and suffering from status quo bias and confirmation bias.<sup>4</sup> They accused earlier thinkers of "becom[ing] the dupe of established professions"<sup>6</sup> and inventing arbitrary factors that serve only to lock in elitist class distinctions. That lock-in effect remains evident in the judicial case law today.

Is there a better way to distinguish professionals from nonprofessionals? In short, yes. The key distinction between professionals and nonprofessionals is the legal substitution of "customary care" for the ordinary "reasonable care" standard. That substitution is an important safeguard when the practice is not a precise science but an inexact art, and thus there is a great need for the exercise of professional judgment. Arguably, this need for professional judgment arises when three conditions are met: bad outcomes are inevitable in the practice; those bad outcomes are attributable to inherent uncertainties in the science of the field; and the practice is socially vital even where bad outcomes are especially likely to occur.<sup>3</sup>

The first factor is important because it means run-of-the-mill practitioners will face questions of tort liability. Doctors and lawyers, for example, are lightning

rods for tort claims because they are expected to intercede in fraught areas with a high likelihood of failure. Doctors are encouraged to treat patients who are unlikely to survive, just as lawyers are encouraged to represent defendants who are unlikely to be acquitted. By contrast, barbers and cosmetologists are not expected to injure their clients in the ordinary course of performing their services. The professional designation is unnecessary when one is unlikely to be sued.

The second factor gets to the heart of the distinction between professional *judgment* and professional *expertise*. When the science of the field is inherently uncertain in light of current knowledge, even experts can reasonably disagree about the appropriate course of action in a particular instance. Medicine and law are inexact sciences. That uncertainty makes it problematic to allow a judge or jury to second-guess the judgment calls that the professional does make. In other words, it is important to judge a professional's actions in light of the full *range* of choices that would be acceptable in the field.

This explanation also provides a crisp justification for why courts have refused—and will likely continue to refuse—to extend the professional designation to skilled trades such as electricians and plumbers. Although skilled artisans possess a high degree of specialized expertise, there is also high uniformity in best practices. When electricians cause fires or plumbers cause water damage, there is less uncertainty about which safety principles should have been followed.

The third factor reserves the professional designation for only those activities that are truly vital to societal functioning. A reduction in medical or legal services would diminish care for the neediest cases. Doctors and lawyers provide essential services for the community and their disappearance can be a crippling loss. Similar reasoning could be extended to other occupations such as clergy and teachers, as well as to firefighters, police, and other critical personnel. By contrast, many dangerous activities can be deterred or even prohibited without ill effect on society. For example, tiger keepers are ineligible for professional status even if their work involves a high degree of risk and uncertainty.

## The key distinction between professionals and nonprofessionals is the legal substitution of “customary care” for the ordinary “reasonable care” standard.

This alternate framework offers a clearer, more compelling case for legal recognition of software workers as a profession. Unlike the trait-based test, which rests on antiquated notions of class nobility, the judgment-based test addresses a problem of modern science. Despite decades of maturation, the practice of software development—like medicine and law—remains deeply uncertain. Any software system of nontrivial complexity is expected to have undetected faults and errors despite careful design and testing practices. Yet, once an error is exposed, it is too easy for outside observers to second-guess one's coding practices and characterize them as careless. Under the ordinary “reasonable care” standard, software liability looks essentially random and unpredictable. Instead, requiring judges and juries to use the “customary care” standard would allow software experts to define a range of acceptable practices, as well as a minimum floor of competence. Ultimately, the professional malpractice framework would improve software quality by offering more sensible legal oversight.

The professional malpractice framework is a nuanced tool capable of addressing a broad range of practice arrangements. For example, both medicine and law can be performed by solo practitioners or large teams, by generalists or specialists, and in lo-

cal or national regions. Moreover, who counts as a “member” of a profession can be determined on a case-by-case basis, just as courts have considered whether the medical profession should include dentists, nurse practitioners, orthopedists, and other related occupations. That heterogeneity should allay concerns that the practice of software development does not look enough like the practice of medicine or law. Customary practices for building cyber-physical systems can and do differ from those for developing database systems, just as they do for enterprise applications versus mobile apps.

Finally, a word should be said about the role of ethics in governing professional conduct. For nonprofessionals, ethical codes of conduct have little bite in determining legal duties of care. If a jury finds one's conduct to be unreasonable, it is no defense to argue that one's conduct was ethical. For professionals, however, evidence of customary practice is decisive. Accordingly, a code of ethics can be deeply influential in shaping the legal duties of professionals, as long as that code reflects and guides actual norms among the professional community.

Much of the computing community has assumed that a more robust commitment to ethics is a prerequisite for legal recognition as a profession. That assumption is exactly backward. Professional malpractice law is needed to catalyze a robust code of ethics. The lesson is this: the best way for ACM's Code of Ethics to make a meaningful difference in changing software development practices is for courts to recognize software as a profession. **□**

### References

- Anderson, R.E. et al. Using the new ACM Code of Ethics in decision making. *Commun. ACM* (Feb. 1993).
- Chien, A.A. Computing is a profession. *Commun. ACM* (Oct. 2017).
- Choi, B.H. Software as a profession. *Harv. J.L. & Tech.* 33, 557 (2020).
- Millerson, G. *The Qualifying Associations*. (1964), 3–4.
- Nimmer, R.T. *The Law of Computer Tech.* § 9.30 (4<sup>th</sup> ed. 2012).
- Roth, J.A. Professionalism: The sociologist's decoy. *Soc. Work & Occupations* 6, 1 (1974).
- Saks, M. A review of theories of professions, organizations and society: The case for neo-weberianism, neo-institutionalism, and eclecticism. *J. Professions & Org.* (2016).

**Bryan H. Choi** (choi.1399@osu.edu) is Assistant Professor of Law and Computer Science and Engineering at The Ohio State University in Columbus, OH, USA.

Copyright held by author.

▶ Mark Guzdial, Column Editor

## Education

# CS Unplugged or Coding Classes?

*Perhaps a more appropriate question is ‘Why not both?’*

**C**OMPUTER SCIENCE UNPLUGGED (CS Unplugged, or just “Unplugged”) is a pedagogy for teaching computational ideas to grade-school students *without* using a computer.<sup>a</sup> It was developed in the early 1990s as a necessity when working with computers in the classroom was not usually practical, but it still finds widespread adoption as a supplement to computer-based lessons, even where devices are readily available. This appears as a contradiction to some (if you are teaching computer science, why not spend as much time as possible on a computer?), while for others it provides a chance to reduce screen time, get physical exercise, and engage with students in a kinesthetic way. Unfortunately, Unplugged can also be used to justify poor decisions by treating it as a complete curriculum in itself—a teacher who does not have the time or support to extend themselves in new curriculum content might rely on Unplugged as “enough,” or administrators might justify a lack of funding by suggesting that schools use Unplugged teaching instead of buying devices.

The Unplugged approach is widely used, mentioned in dozens of research papers about CS education, has been translated into many languages, and is widely used in teacher professional development.<sup>1</sup> A number of studies have



reported positively on its use. And yet there are those who have raised concerns about teaching computer science without computers (for example, Stager and Martinez<sup>7</sup>) since it appears antithetical to learning programming.

So why does it work for some, and not for others?

### The Intention of CS Unplugged

Unplugged is based on a *constructivist* approach: students construct their own knowledge by engaging with computational challenges based on concisely described rules, where the process of solving the challenge gives

them understanding of significant principles.<sup>2</sup> They work with games and puzzles that have constraints that occur in computing, and this enables students to discover important ideas from computer science, such as the insight that an algorithm does not necessarily take twice as long to process twice as much data; or to understand concepts in everyday life, such as error detection that makes barcodes reliable to scan; or to grapple with the limits of what can be computed.

Unplugged was developed by computer scientists working with young children to show them what computer

a See <http://www.csunplugged.org> for examples of activities, including videos of it in action.



## 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



scientists are passionate about. It focused on getting students to experience the kind of thinking that computer scientists engage in, such as solving graph problems, working with digital representations, and grappling with computational complexity. Later, the phrase “thinking like a computer scientist”<sup>8</sup> would turn up as a description of computational thinking (CT), and hence Unplugged activities resonated with movements to introduce CT in schools.

As CT has become part of formal curricula, and devices are readily available in many schools, students now have the opportunity to engage with computer programming. Unplugged never sought to replace computer programming, but to provide impactful experiences with computer science without depending on programming. I am a strong advocate of students engaging with programming, as the experience of using thought to create something out of nothing can be life-changing for some students. However, we should not assume access to devices is the only barrier to learning programming, and it has emerged that using Unplugged activities with programming can improve students' self-efficacy and engage a more diverse audience.

A turning point was a study by Hermans and Aivaloglou,<sup>5</sup> who worked with two groups of students: one group spent four out of 10 weeks doing Unplugged activities before learning programming, and the other group spent the entire 10 weeks on programming. Between the groups they found “no difference in performance on the understanding of programming concepts. However, the unplugged first group shows more self-efficacy and ... a wider vocabulary [of commands].” Here, Unplugged activities seemed to act as a catalyst when combined with programming to bring about a better result.

To encourage this link, the CS Unplugged site now includes a “plugging it in” section that explicitly links Unplugged activities to programming, rather than leaving it for teachers to make the connection.

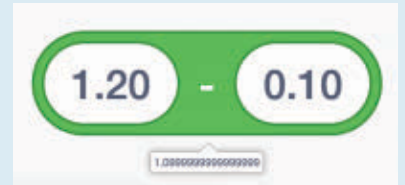
### Goals of Introducing CS to Younger Learners

Sometimes CS education in schools is seen primarily as developing compe-

### Examples of Computer Science Unplugged activities:



(a) A student using cards to engage with how data can be represented using binary digits.



(b) Confusing arithmetic in a programming environment that can be explained by the binary card activity.



(c) An outdoor activity where students experience the power of using simple “if” statements to sort values into order.

tent programmers to address a labor shortage, but there are deeper social goals at play that must be considered if we want to avoid focussing on a subset of students.

**Not just knowledge and skills.** We must address the lack of diversity in the computing industry, and to do this it is important to find gateways to spark the interest of those who might not otherwise see themselves in this role. Plutarch noted that “education is not the filling of a pail, but the lighting of a fire”—teaching should go beyond knowledge and skills, and needs to value qualities like self-efficacy.<sup>6</sup>

How best to “light a fire” for students depends on the context. An example of this is discussed in Guzdial’s recent article,<sup>4</sup> where a curriculum that is light on programming and includes Unplugged teaching increased participation in advanced CS courses for underrepresented students.

**Not everyone has access.** In many countries students have the privilege of ready access to devices for education, but not all have enough access to develop programming skills well. Sometimes the issue is economic—some schools simply do not have computers—but the reason may also be philosophical (for example, Waldorf schools discourage the use of computers at lower grade levels) or legal (for example, in many jurisdictions, prison inmates are not allowed access to computers<sup>b</sup>).

**CS is more than programming.** There is more to CS than just cutting code. We do not write programs for computers, we write them for people. This means we need to develop a culture of knowing how to make programs respond quickly, have interfaces that take account of the user, and treat data with respect. Stepping back from the computer provides a chance to think about how to address these issues, rather than rushing in to write a program, only to find that it seems to take forever to finish a computation.

### Why Does Unplugged Work for Some and Not Others?

Some people hold strong ideas about a range of aspects of teaching computer science, and these are partially explained because individuals can become very focused on how a particular approach has worked in a specific context. Teachers can develop extreme views: some are very experienced and risk overwhelming students with their aspirations, while others are fearful of teaching topics that they have no experience in, and revert to the familiar.

**Context is important.** Busuttil and Formosa<sup>3</sup> found students were more engaged using Unplugged in a gym than in a computer lab, which reflects my experience—any non-computer activity in a lab may seem like a wasted opportunity to access (possibly limited) computing resources, and students can tune out. It is also important to take account of the learners' values, experience and expectations, which can vary considerably!

**CS Unplugged is not a curriculum.** Unplugged is sometimes treated as a curriculum (without connecting it to computers) rather than a pedagogy. In-

## Unplugged was developed by computer scientists working with young children to show them what computer scientists are passionate about.


deed, interventions that attempted to use CS Unplugged as a self-contained curriculum instead of weaving it into an existing one have been found to be ineffective. Sometimes it may be necessary to teach regular classes without computers, but in a well resourced education environment the CS Unplugged activities are intended to be used in conjunction with conventional programming lessons—and any other effective tools for teaching CS.

**Teachers need support.** We must acknowledge that as new CS content is introduced to school curricula, in many countries there is limited time for teachers to get up to speed. An underinvestment in teachers forces them to either invest their own time (which they may not have the luxury of being able to donate), or else they quite understandably may latch onto the first ideas that make sense to them, and make do with that. Developing self-efficacy in students is the goal, but teachers themselves need to be supported to develop their own self-efficacy in this discipline so that they can teach with confidence and competence!

### Conclusion

Working out how to teach CS is not simple, and it is not surprising we see different outcomes as a particular tool is used in different contexts, and with varying support for the teachers using it. When Unplugged was created 30 years ago it was designed for quite a different context to the curricula now appearing around the world, and

yet it has emerged as a useful tool as part of the many approaches available for engaging a variety of students in computer science. A combination of approaches is needed if we wish to engage a diverse range of students, and we must support teachers to adopt an appropriate selection of teaching methods. We also must be aware they are working in highly constrained environments. Computing professionals who have the opportunity to interact with teachers can often become focussed on one particular aspect that they personally find engaging, but we need to be aware of the range of tools to choose from and think about what will work in each teacher's context.

Unplugged can be understood as a catalyst that supports learning to program. While sometimes learning offline is a necessity because of resource constraints, even if these did not exist it is emerging that Unplugged can be used to improve self-efficacy in students without increasing teaching time or decreasing programming skills. 

### References

1. Bell, T. and Vahrenhold, J. CS Unplugged—How Is It Used, and Does It Work?. In *Adventures between Lower Bounds and Higher Altitudes*. Springer, Cham. (2018), 497–521; doi:10.1007/978-3-319-98355-4\_29
2. Bell, T. and Lodi, M. Constructing computational thinking without using computers. *Vrije Universiteit Brussel, Special Issue "Constructivism and Computational Thinking", *Constructivist Foundations* 14, 3 (2019), 342–351; https://constructivist.info/14/3/342bell*
3. Busuttil, L. and Formosa, M. Teaching computing without computers: Unplugged computing as a pedagogical strategy. *Informatics in Education* 19, 4 (2020), 569–587; https://doi.org/10.15388/INFEDU.2020.25
4. Guzdial, M. The goal of the first CS course should be to promote confidence if we're going to increase diversity in CS: Paying off on a bet. *Computer Education Research Blog* (Dec. 29, 2020); https://bit.ly/3bSEN7c
5. Hermans, F. and Aivaloglou, E. To Scratch or not to Scratch?: A controlled experiment comparing plugged first and unplugged first programming lessons. In *Proceedings of the 12<sup>th</sup> Workshop on Primary and Secondary Computing Education*. ACM, New York, NY, USA, 2017, 49–56; https://doi.org/10.1145/3137065.3137072
6. Malmi, L. et al. Theories and Models of Emotions, Attitudes, and Self-Efficacy in the Context of Programming Education. In *ICER '20 Virtual event, New Zealand*. (2020), 36–47; https://doi.org/10.1145/3372782.3406279
7. Stager, G. S. and Martinez, S. Thirteen considerations for teaching coding to children. In Humble, S., Ed., *Creating the Coding Generation in Primary Schools: A Practical Guide for Cross-Curricular Teaching*. Routledge, 2017; https://doi.org/10.4324/9781315545813
8. Wing, J.M. Computational thinking. *Commun. ACM* 49, 3 (Mar. 2006), 33–35; http://doi.acm.org/10.1145/1118178.1118215

**Tim Bell** (tim.bell@canterbury.ac.nz) is a Professor at the University of Canterbury, NZ.

Copyright held by author.

<sup>b</sup> For example, see <https://learnlevel.org/>

## Viewpoint

# Understanding Law and the Rule of Law: A Plea to Augment CS Curricula

*Why law matters for computer scientists and other folk.*

**S**OME PEOPLE THINK they are above the law. In a constitutional democracy this cannot be the case. Neither the head of state nor the doctor or the police are above the law. They should all be enabled to do their work, but we do not buy the claim that they could act as they wish. In 18<sup>th</sup> century Europe we replaced the authoritarian rule *by* law with a rule *of* law, to mitigate uninhibited power, and to ensure that those in power can be held to account in a court of law. Whereas *rule by law* is *rule by persons* (law as an instrument of control), *rule of law* implies a division of powers where those who enact the rules do not get the last word on their interpretation.<sup>13</sup>

This also refers to the difference between law and ethics. Replacing rule *by* law with rule *of* law means we do not want to depend on the ethical inclinations of those who rule us. Instead, we can send them home if we don't agree with the rules they impose (democracy) and we can contest their interpretation of those rules in court (rule of law). As a thought experiment I ask the reader how this would apply to the rules computing systems impose: Can we send home the developers (and/or those who implement these systems to gain a profit or to engage in public administra-



tion)? Can we contest their rules in a court of law when they impact our choice architecture?

Law and the rule of law have been implemented by way of intricate checks and balances that safeguard the contestability of legally relevant decision making, thus preparing the ground for robust, legitimate, and binding decisions. This is how we create and sustain societal trust: not by cherishing the illusion of an ideal world where

power plays no role, but by creating and sustaining countervailing powers. Simultaneously, law is about coordinating human interaction, making sure that governments treat their citizens with equal respect and concern,<sup>5</sup> thus providing for legal certainty and justice. That is why it is imperative that nobody is above the law.

This also goes for the architects of our computational environments, who increasingly design and engineer

the space we inhabit. Computer scientists, Web developers, roboticists, and software engineers must understand both when and how the law applies to them, and insofar as they develop modules, systems or applications for specific use cases, they should be sensitized about how and when the law may apply. This goes for issues of privacy and data protection, cybercrime, intellectual property rights and private law liability (for example, tort), but also for issues of jurisdiction (what law applies) and international law (how national legal systems interact at the global level). It goes even more for the idea of the rule of law that should inform our understanding of the law.

Based on many years of teaching law to master's students of computer science,<sup>8</sup> I have come to believe that by teaching them about law I am not only helping them to comply with current law, but also offering them a unique opportunity to engage with the foundations and implications of their own 'trade' (precisely because computing systems also produce rules that affect human behavior).

I have noticed students' intuitive understanding of complexity (developed while studying the behavior of computing systems) provides them with unique analytical skills that are surprisingly relevant for the study of law and the rule of law. Obviously, computer scientists are more aware of the limitations of formalized systems than others, making them open to what law has on offer as a complex, multidimensional, adaptive operating system.

In this Viewpoint I hope to explain why law and computer science have much in common, as well as much to learn from each other. Basically, I will argue that an introduction to law *and to the rule of law* should be integrated into the curriculum of computer science, considering the huge impact of computational systems on our shared world.

### Legal Research and Computer Science

Computer science is about the design and behavior of computing systems, including their verification, grounded in mathematics, information theory,

## Computer scientists are more aware of the limitations of formalized systems than others, making them open to what law has on offer as a complex, multidimensional, adaptive operating system.

statistics, electrical engineering and often including the study of computational game theory, complexity theory, and cognitive theory, as well as cybernetics, the theory of programming languages, and much more. Hopefully it is also about the falsification of testable theory that contributes to scientific research.<sup>11</sup>

Legal research is about the interpretation and development of positive law; that is, the architecture of binding legal norms that define a specific jurisdiction, thus ensuring alternative interpretations can be argued, and safeguarding the integrity of law and the rule of law.<sup>5,8</sup>

Computer science concerns code that decides the behavior of a system, depending on specified inputs; legal research concerns a specific type of norms that decide what legal effect follows when certain conditions apply. Legal norms can be both written or unwritten, but in both cases they are expressed in natural language. Law is text-driven. Computing systems are code- (and possibly data-) driven.

### Law and Computing Systems

This is where things become interesting. For computing systems it is crucial to remove ambiguity, for law it is

crucial to sustain the open texture of human language while still ensuring closure.<sup>6,7</sup> The beauty of 'natural' language is that it simultaneously opens a space for multiple interpretations of the same word, sentence, paragraph, or larger text body, and provides the means for the closure that is necessary to achieve mutual understanding.

This closure, however, is never final as it can always be called into question, for instance, by using language in a different manner, by connecting terms with previously unconnected terms or by creating references to new events or situations. The latter is interesting, because human language not only refers to objects such as chairs and tables, bridges and highways, but also to objects such as 'marriage', 'religion', or 'economic markets' that are neither given nor tangible in the way that, for example, rivers and canals are. Those objects are the result of performative speech acts that 'do what they say'.<sup>1,10</sup> When a civil servant declares a couple husband and wife, they are not describing a situation but calling it into existence, with all the (legal) consequences this entails.<sup>9</sup>

This type of performative speech act appears in the law, where lawyers are trained to determine what factual circumstances 'count as' a specific legal fact, and subsequently, to determine the specific legal effects of the relevant legal fact. To achieve such determinations, a lawyer will first investigate under what jurisdiction the problem falls.

### Jurisdiction

For instance, a lawyer may have to determine whether a specific action 'counts' as 'unfair', meaning that they have to elicit the relevant legal conditions and check whether they apply, based on the relevant facts. The legal requirement of 'fairness' returns in many different jurisdictions and is part of many different legal domains (decisions of public administration must be fair, punishment must be fair, compensation for breach of contract must be fair, recruitment of employees must be fair). Lawyers are accustomed to the fact that fairness is not a predefined notion and has dif-

## Distinguished Speakers Program

**A great speaker can make the difference between a good event and a WOW event!**

Students and faculty can take advantage of ACM's Distinguished Speakers Program to invite renowned thought leaders in academia, industry and government to deliver compelling and insightful talks on the most important topics in computing and IT today. ACM covers the cost of transportation for the speaker to travel to your event.

**speakers.acm.org**



Association for  
Computing Machinery

ferent meanings, depending on the legal domain and the case at hand. Deciding its meaning requires them to answer a series of very specific questions, such as 'fair compared to what' (what type of cases should be considered the point of reference), 'fair compared to whom?' (which others should be treated equally), and 'fair in respect of what?' (of a private or a public interest). The answers to those questions will have legal effect, they are not part of a thought experiment or a model building exercise. They will affect real people in the real world, depending on the relevant jurisdiction.

Take, for instance, the judgment of the Court of Justice of the European Union (CJEU),<sup>a</sup> which concerned unisex rules on insurance premiums and benefits, and was based on the legal prohibition to use 'sex as an actuarial factor'. This legal prohibition meant that such usage 'must not result in differences in premiums and benefits for insured individuals': even though the driving behaviors of men was statistically more risky, the Court concluded the insurance company was not allowed to charge men a higher premium. Considering the fact that many other factors may function as a proxy for 'sex', this case is highly relevant for risk assessments based on machine learning. The decision, however, is only applicable within the jurisdiction of the EU. We are not in the realm of universal rules on fairness, which actually do not 'exist' in the real world. Neither in ethics (where disagreement abounds), nor in law (whose validity is restricted to a particular jurisdiction).

Other than the envy-free cake cutting algorithm, discussed in a previous issue of *Communications*, lawyers cannot abstract from real life implications based on invalid assumptions. For instance, we cannot assume agent 1 will be able to find a division with equally preferred slices of cake, nor can we assume that agent 2 would not have preferred another initial division (noting that many different initial divisions are possible). On top of that lawyers, economists and

politicians may observe that once an envy-free division has been realized, the incentive to enlarge the cake could be lost—diminishing the appetite for innovation. Finally, the assumption that envy rules the world is simply one way of framing the problem. The assumption is utilitarian (preference based), and grounded in a particular kind of methodological individualism that may not hold. Utilitarianism does not necessarily provide for the best way of understanding the notion of a fair division (and fairness is not per se about division). Law is pragmatic and refrains from determining what is 'fair' in any universally valid sense, limiting itself to the question whether a given state of affairs should be qualified as 'unfair', depending on the relevant jurisdiction.

The latter refers to the fact that the legislature and the courts of a particular state or supranational organization determine what counts as unfair within that jurisdiction. What counts as unfair also depends on the legal domain (criminal law is about desert,

**Computer science concerns code that decides the behavior of a system, depending on specified inputs; legal research concerns a specific type of norms that decide what legal effect follows when certain conditions apply.**

<sup>a</sup> CJEU, 1 March 2011, Case C 236/09 (Test-Achats), at 30.



prevention, deterrence; private law is about compensation, autonomy, property; administrative law is about the public interest), and on the particular circumstances of the case at hand. There is no algorithm for deciding what counts as fair, it requires judgment rather than calculation. An algorithm would forever lag behind what is relevant, trained on historical data or framed by code written on the basis of previous events. The situated nature of (un)fairness implies that lawyers are familiar with many of the issues that ‘fair computing’ struggles with, they are used to the fact that deciding the meaning of (un)fairness is an act of interpretation rather than the outcome of a calculation.

However, lawyers could learn lots about their own blind spots by taking note of the myriad interpretations of fairness that have been detected by machine learning experts,<sup>3</sup> while the latter could learn lots about the myriad considerations that lawyers take into account when determining what ‘counts as’ (un)fair. Legislation and case law provide for an especially rich resource of granular decisions of the meaning of ‘unfair’ or ‘fair’. Those decisions have force of law within a specific jurisdiction, which may be national but also international (for example, human rights treaties). This highlights the fact that concepts such as fairness do not lend themselves to universal interpretation, as a computer scientist may be tempted to believe. Law, instead, takes into account the history, values, and cultural background that grounds a jurisdiction. Simultaneously, even at the level of transnational jurisdiction, there are limits to what qualifies as fairness. Taking cultural settings into account does not mean that anything goes. This suggests a family resemblance between different interpretations rather than a set of necessary and/or sufficient conditions.

### If You Cannot Disobey a Law, It Cannot Be Law

In the meantime, the fact that law can be disobeyed is often presented as a drawback. This is, however, not a bug but a feature. The law appeals to the agency of those under its jurisdiction, it does not self-execute or

## It is pivotal that computer scientists come to grips with the architecture, the multidimensionality and the constitutive force of law.

twist their arm. In the latter case, we would not speak of law but rather of administration, discipline, or brute force.<sup>4</sup> Clearly, those who disobey the law may find themselves up against law enforcement: they may have to pay damages, or fines, or they may even be punished. This is where law differs from ethics. Following the law is not a matter of individual preference or personal taste: nobody is above the law. Nevertheless, in a constitutional democracy, law enforcement leaves room for contestation (due process, a fair trial), treating those under its rule as agents capable of giving reasons for their actions, thus respecting their dignity.<sup>12</sup>

### The Binding Force of Law, Code, and Ethics

Defining law is like nailing a pudding to the wall, legal historian Uwe Wesel once wrote.<sup>14</sup> The same goes for all human institutions, such as ‘marriage’, ‘universities’, ‘corporations’, ‘religion’, or ‘economy’, ‘artificial intelligence’, and even ‘computer science’. This, again, is not a bug but a feature, as it enables us to navigate our shared institutional world in a reasonably smooth way— institutions are adaptive without being altogether undefined (one could say they are underdetermined). What differentiates law from ethics is neither brute enforcement nor mechanistic application of rules that speak for themselves, but a complex web of binding norms, enforceable decisions and dedicated institutions that

empower and protect human agency as well as societal trust.

It is pivotal that computer scientists come to grips with the architecture, the multidimensionality, and the constitutive force of law. Designing computational systems ‘to do good’ is not enough. The end users of those systems should not be dependent on the ethical inclinations of individual developers, when confronted with an environment that is soaked in computational infrastructure. Before engaging with the ethical implications of such infrastructure, computer scientists must urgently be trained in both the theoretical underpinnings and the practical affordances of both law and the rule of law. ■

#### References

1. Austin, J.L. *How to Do Things with Words*. Oxford University Press, Oxford, 1962; <https://bit.ly/3ttf8rH>
2. Aziz, H. and Mackenzie, S. 2020. A bounded and envy-free cake cutting algorithm. *Commun. ACM* 63, 4 (Apr. 2020), 119–126; <https://bit.ly/38NJe1g>
3. Barocas, S., Hardt, M. and Narayanan, A. *Fairness and Machine Learning*. 2019; <https://bit.ly/3cNQ0f8>
4. Brownsword, R. *Rights, Regulation, and the Technological Revolution*. Oxford University Press, Oxford, 2008.
5. Dworkin, R. *Law's Empire*. Belknap Press, Cambridge, MA, 1986.
6. Hart, H.L.A. *The Concept of Law*. Clarendon Press, Oxford, 1994.
7. Hildebrandt, M. Radbruch's Rechtsstaat and Schmitt's Legal Order: Legalism, legality, and the institution of law. *Critical Analysis of Law* 2, 1 (2015); <https://bit.ly/30QQyEX>
8. Hildebrandt, M. *Law for Computer Scientists and Other Folk*. Oxford University Press, Oxford, New York, 2020.
9. MacCormick, N. *Law as Institutional Fact*. University of Edinburgh Press, Edinburgh, 1973.
10. Searle, J.S. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, U.K., 2011.
11. Van Rooij, I. and Baggio, G. Theory before the test: How to build high-verisimilitude explanatory theories in psychological science. (2020); <https://bit.ly/2Nvapqm>
12. Waldron, J. The rule of law and the importance of procedure. *Nomos* 50, (2011), 3–31.
13. Waldron, J. The Rule of Law. In E.N. Zalta, Ed. *The Stanford Encyclopedia of Philosophy* (Summer 2020), Metaphysics Research Lab, Stanford University (2020); <https://stanford.io/3157zeZ>
14. Wesel, U. *Frühformen des Rechts in vorstaatlichen Gesellschaften. Umriss einer Frühgeschichte des Rechts bei Sammlern und Jägern und akephalen Ackerbauern und Hirten*. Suhrkamp, Frankfurt am Main, 1985.

**Mireille Hildebrandt** (M.Hildebrandt@cs.ru.nl) is a tenured Research Professor on ‘Interfacing Law and Technology’ at Vrije Universiteit Brussels, and holds the part-time chair of Smart Environments, Data Protection and the Rule of the Law at Radboud University in the Netherlands. She is author of the recently released *Law for Computer Scientists and Other Folk* (Oxford University Press) available free of charge in .pdf format, no DRM, no Adobe Digital Editions, at <https://bit.ly/3eOJshp>

Copyright held by author.

## Viewpoint

# The 10 Best Practices for Remote Software Engineering

*Focusing on the human element of remote software engineer productivity.*

**A**T FACE VALUE, when we think of developer productivity we might think of effectiveness in time management, communication, and task completion.<sup>14</sup> Although we are drawn to personal workflow or time management tools, and learning secrets to improving our productivity, ironically this quest for the holy grail can sometimes take us off course and be a detriment to our productivity. The problem is that accomplishing tasks or having a filled up schedule does not necessarily equate to productivity. Creating a formulaic working strategy, as was common in the last century, does not either.<sup>a,8,13</sup> Productivity is less a quality that can be easily measured,<sup>b</sup> controlled, or improved directly with tools, but instead is a human element that manifests from developer happiness.

This Viewpoint is intended for remote software engineers who are facing new challenges to thinking about routine, responsibility, and goal setting. As a developer of scientific software, and one who has transitioned to working remotely before any stay at home orders,<sup>24</sup> I have slowly learned to optimize my own productivity by focusing exclusively on well-being. In this Viewpoint, I summarize what I have learned.



### Best Practices

**1. Work on Things You Care About.** It goes without saying that a core ingredient to happiness, and thus productivity, is working on projects or software that you care about.<sup>31</sup> In fact, this has been experimentally shown: happier workers are more productive, and lower happiness associated with adverse life events is a detriment to productivity.<sup>7,28</sup> Thus, we might step back to say that before we discuss any best practice criteria, you should enjoy the practices of software engineering, or participation in a community, enough to warrant having it a part of your daily life. If you are not sure if you are in the right role, then assess it based on the

small tasks that it encompasses. You can break your day down into small events or tasks, and take a mental note about whether or not you enjoyed each one. For example, a software engineer might realize they enjoy working publicly on open source on GitHub, and writing documentation, but are not so keen on reanswering the same questions on a private user forum. Figuring this out would help the software engineer to push for projects and responsibilities that are better catered to their interests. If you cannot identify any items, or if all the items identified are considered negative experiences, then it is time to look more critically at your role. There is no list of best prac-

a *The Scrum Guide*. Scrum.org (2020): <https://bit.ly/3ePRiT6>

b The myth of developer productivity; <https://bit.ly/3m3uvF7>

tices that you could reasonably follow to make you happy if you are not in the right line of work to begin with. If this is true for you, you will have a difficult time being regularly productive.

**2. Define Goals for Yourself.** As software engineers, it is sometimes the case that we don't have a career ladder.<sup>21</sup> Many of us might not sit within established groups where we have a manager who is aware of supporting our personal and professional development and growth. This can be problematic because goal setting can generally have a positive effect on performance.<sup>35</sup> Under these circumstances, and especially in the present moment when we are more disconnected from our teams, we have to take our personal and professional development into our own hands. This means thinking about the kind of work that you want to do, or more generally, the goals you want to accomplish in the short and long term. For example, if you program a lot, your goal might be to learn a new language that would be useful to know. If you are feeling disconnected, your goal might be to pursue interacting with a small set of new communities, or creating structure into your schedule for discussing projects or work with colleagues. If you are involved with research, your goal might be to submit or publish a paper. Short-term goals might be in the context of a week or month. You might want to solve a particular challenge, add a feature to your software, or write up an idea you have. Having these goals is important for your productivity because you can better understand how your work fits into the overall story of you, as a software engineer. If you share your goals with your colleagues or supervisor, they might learn they are important to you, and be interested to support you or otherwise keep an eye out for you.

**3. Define Productivity for Yourself.** If we were in a role that explicitly produced some kind of output (for example, a farmer might grow corn) then we might easily measure and define productivity<sup>c</sup> as these units of output. However, in context of a rich role that involves not just writing code but also

## It goes without saying that a core ingredient to happiness, and thus productivity, is working on projects or software that you care about.

interacting with people and ideas, this simple representation does not work very well. While we could define productivity based on lines of code or developer outputs,<sup>26</sup> this definition fails to capture individual differences. There is huge variance in software engineering work, and it would be a daunting task to quantify or qualify what productivity or happiness means globally. However, even though we lack definition, we still are cognizant of what a productive day feels like. Although there are arguably different kinds of happiness,<sup>5</sup> we do not need to have a holistic definition to know if we feel good at the end of a day. A productive day is one that you finish and feel satisfaction that your work was meaningful. Thus, it follows that we might be able to define productivity for ourselves by assessing our daily tasks and deciding if they contribute to that feeling of satisfaction. A software engineer might create a physical list, and add items to it whenever something feels done or accomplished. These items can either be items of work that are mention-worthy, or softer goals like fostering a collaboration or growing a community. This list can be hugely useful for deriving a better understanding of yourself, and what makes you happy or productive. Although happiness and productivity are not exactly the same thing, they seem to be intimately related.<sup>17</sup>

**4. Establish Routine and Environment.** Small details about your working environment, or lack of a routine, can hugely throw off your workday, and thus your productivity. You should generally pay attention to the lighting, noise level, and comfort of a work

space. If you find yourself distracted by anything, you might consider changing your environment. Habitual repetition by way of thinking about your environment or making explicit choices can lead to the establishment of longer-term habits.<sup>25</sup> The more that you are able to repeat a behavior in a specific environment, the more likely it is to become effortless.<sup>12</sup> For example, if you immensely enjoy browsing the Internet for pleasure, you might teach yourself to look forward to this activity after dinner, and choose to browse at a different location with a different device to not associate either with work. If you are the kind of person who enjoys the routine of working in a coffee shop, you might try to emulate this experience at home by creating several different working spots, and taking a walk in the middle of the day to mimic moving between them. In the case of a maladaptive habit you want to get rid of, you might consider changing your environment.<sup>37</sup> For example, if you typically work alone and you do not like it, you might consider trying an environment with more regular interactions with another person. If you do not feel productive, you might consider testing a different working location, morning ritual, or lunch or exercise routine. An important aspect of this routine is creating a clear separation between times when you are working, and times when you are not. This is especially challenging and important for working at home, because any routine of going to and from work is gone, and home becomes two in the same. Finally, a routine that includes more than work, meaning activities that are for your personal well-being (exercise, time with family, relaxation), are essential to consider. The remarkable features of a healthy routine and environment are that they work so well that you do not notice them.

**5. Take Responsibility for Your Work.** If you are in a role that provides user-support, work on a team, or otherwise have a list of tasks set out for you, there is a tendency to be passive. While you might check off items from a to-do list on a daily basis, this does not necessarily indicate that you have taken ownership or responsibility for your work. Having this ownership can be a driving factor for caring more about your work,

<sup>c</sup> Definition of *productivity*—see <https://bit.ly/38M7HEc>



2018 JOURNAL IMPACT  
FACTOR: 6.131

## ACM Computing Surveys (CSUR)

*ACM Computing Surveys* (CSUR) publishes comprehensive, readable tutorials and survey papers that give guided tours through the literature and explain topics to those who seek to learn the basics of areas outside their specialties. These carefully planned and presented introductions are also an excellent way for professionals to develop perspectives on, and identify trends in, complex technologies.



For further information  
and to submit your  
manuscript,  
visit [csur.acm.org](http://csur.acm.org)

and thus your productivity.<sup>30</sup> It is a challenging but important skill to learn how to independently recognize and address challenges in your community or space. For example, a challenge might be anything from a small annoyance that your group faces on a regular basis, to a significant computational problem. The more capable you are of deciding something is important to do, creating a plan to do it, and executing that plan, the more you can establish independence, learn new skills, and feel you have ownership over your work. If you share your ideas or projects with your supervisor or team, they will also see these positive qualities in you, and this can help to establish trust, which is hugely important in a remote working environment. In fact, participation in organizational decisions can give a sense of psychological ownership that also improves general well-being.<sup>20</sup> Taking ownership on both of these levels, in that it establishes trust, can help to solidify how your team thinks about and consequently treats you. Your supervisor should not require tracking software to keep abreast of your every move because they have confidence and trust in you.

**6. Take Responsibility for Human Connection.** When newly remote, it is common to place the burden of connection on your team. If your team is not remote first,<sup>11</sup> it is even more unlikely that there are established protocols and best practices for how to make you feel included and happy. Many of us learn this the hard way right at the offset of going remote. If we become disconnected from our immediate or even open source communities, it can feel terribly lonely and have direct impact on our well-being.<sup>9,34</sup> In fact, social-isolation or exhaustion alone can trigger this same loneliness.<sup>32</sup> In this situation, it is important to identify that there is a problem, namely lack of human connection, and figure out the kinds of human connection that are meaningful to you. You might first find comfort in the fact that feeling lonely is more common than you might think.<sup>36</sup> You might then seek out human connection. Seeking human connection means attending virtual workshops, conference calls, or looking around for regular meetings that you could participate in. If you are looking for more one-on-one human interaction,

you might see if anyone would be interested in having a virtual coffee a few times a month, or a virtual “happy hour” to involve others. The choice of group is up to you. While it is reasonable to consider your colleagues at work, what you are missing might be connection to friends or family, and so a regularly scheduled family meeting or even meeting online to play a game could be a regular event. In the context of work, you can also use the trick of sharing your work to pursue human connection. If you share progress on your projects on social media or Slack, it likely will open lines of communication with others interested in your work. Whatever your strategy, by changing your environment you can feel less lonely.<sup>38</sup> As a remote worker you might live in isolation, but you do not need to work in it.

**7. Practice Empathetic Review.** Software engineering requires a lot of technical communication. Whether you are having discussion in a chat window, on video, or a code review, you are undoubtedly going to encounter a lot of different kinds of people, all with different expectations and incentive structures. Empathy is essential. Communication can lead to good productivity,<sup>2,27</sup> and having good productivity can further make you a better, less adversarial communicator.<sup>19</sup> For example, if you encounter a maintainer of a repository seeming curt or mean, you might consider the responsibility on their shoulders to maintain the software, the number of other issues or pull requests they need to review, and that they are working in free time. The maintainer is likely to be in a stressful role. You might also consider cultural differences in communication—it could be that

**Although happiness  
and productivity  
are not exactly  
the same thing,  
they seem to be  
intimately related.**

a succinct response is more common in their community or culture. When you take these factors into account, a response that you might have reacted with that is defensive, confrontational, or otherwise aggressive, might be adjusted to have kindness, and further, be productive by asking how you might help. Arguably, the quality of communication on the level of an organization comes down to the quality of the individuals' communication within it. Better communication leads to positive outcomes that range from addressing an issue to solving a problem, or simply feeling involved with decision making.<sup>15</sup> No interaction is too small to practice good communication. Practicing empathetic development means introducing yourself, describing the issue or change clearly, and making it easy for the other party to reproduce. By setting an example, you will not only influence the others involved in the work at hand, but also set an example for the immediate community and larger community of software engineers.

**8. Have Self-Compassion.** You can do your best to define personal goals, establish routine, and do meaningful work that feels productive, but this does not always mean that you finish the day and feel that you were successful. This is where self-compassion<sup>d</sup> and mindfulness<sup>4</sup> are important, as they have been shown to have positive impact on mental health and well-being.<sup>3</sup> Self-compassion can broadly be defined as being mindful and forgiving to yourself.<sup>16</sup> Having self-compassion means looking back on the experience of a day, and not egging yourself for everything that was not perfect. It is just a given that every day will not feel productive. You might try new routines that do not work for you. Instead of thinking negatively, you might consider that establishing your routine, or figuring out a hard problem, is a work in progress, and there is no real state of failure because you wake up and try again. I have found that it helps to try to imagine myself as another person. If someone else approached me with the same experiences and negative thoughts, how would I comfort them? It is very likely that I would react with

d The Five Myths of Self-Compassion; <https://bit.ly/3bVDepm>

## As a remote worker you might live in isolation, but you do not need to work in it.

compassion, and that is the same compassion that I must find for myself. If the source of criticism comes from an external source, although you cannot control other people, you can choose to respond thoughtfully, with kindness, and try to find humor in the situation. However, obviously if someone is in violation of some code of conduct, you should report it, and not absorb it quietly. Thus, self-compassion also means knowing when to stand up for yourself and say something. This advice is especially pertinent for women and other minorities in the work force who might experience subtle inequality or microaggressions.<sup>e</sup>

**9. Learn to Say Yes, No, and Not Anymore.** As a software engineer, engaging with projects can be akin to going to a candy store. Each one offers a rich set of problems to work on, people to work with, and new languages or technologies to learn. You might be someone who very easily says “yes” to contributing to a project, or generally providing help.<sup>10</sup> While this is a really great way to grow as a developer and person, there is also a time to say no: when you do not have the bandwidth, do not share the vision of the project, or otherwise have a gut feeling telling you to do so.<sup>13</sup> It is also okay to say “yes” but then scope your contribution to an amount that you have time for. Finally, what if you originally say “yes” and then change your mind? This is okay too. To maximize your productivity, maintaining awareness about what is on your plate, and when it is time to ramp up or taper down engagement is essential for focusing on the projects that are most valuable and important to you or your community.<sup>18</sup>

e Almost two thirds of women face everyday sexism and racism at work; <https://bit.ly/3eOB64l>

**10. Choose Correct Communication Channels.** When working remotely, there are many ways to communicate with others. In that people have different working environments and schedules, ideally the communication is asynchronous, meaning that you use messaging services like Slack, or issue boards on version control services like GitHub.<sup>f</sup> However, you should be careful to choose the method based on the needs of the communication.<sup>g</sup> Discussion that should be open and linked to code would be better on GitHub issues or Gitter<sup>h</sup> than Slack. A discussion that moves into a document might first do really well using a collaborative document tool like Google Docs, but after some hardening you might want to move it into version control. Sometimes a quick call is more efficient than trying to write out a verbose email.<sup>1</sup> The important detail is that, whether you choose a video call, an email, or a Slack message, your choice of channel is appropriate for the needs of the problem that needs to be discussed, the degree to which the communication should be transparent and open, and your own level of comfort.<sup>23,29</sup> If you are less comfortable with video or voice chat, you might section off specific time slots or days for it to make it a predictable part of your routine.

### Discussion

I have summarized 10 suggested best practices to optimize remote developer happiness, and thus remote developer productivity:

- ▶ Work on things that you care about;
- ▶ Define goals for yourself;
- ▶ Define productivity for yourself;
- ▶ Establish routine and environment;
- ▶ Take responsibility for your work;
- ▶ Take responsibility for human connection;
- ▶ Practice empathetic review;
- ▶ Have self-compassion;
- ▶ Learn to say yes, no, and not anymore; and
- ▶ Choose correct communication channels

I can personally attest that by discovering and subsequently following these

f Build software better, together.

g *Journal of Knowledge Management Practice*. (Oct. 2001); <https://bit.ly/30UB56O>

h Gitter; <https://gitter.im/>



## Digital Threats: Research and Practice

*Digital Threats: Research and Practice* (DTRAP) is a peer-reviewed journal that targets the prevention, identification, mitigation, and elimination of digital threats. DTRAP aims to bridge the gap between academic research and industry practice. Accordingly, the journal welcomes manuscripts that address extant digital threats, rather than laboratory models of potential threats, and presents reproducible results pertaining to real-world threats.



For further information  
and to submit your  
manuscript,  
visit [dtrap.acm.org](http://dtrap.acm.org)

guidelines over half a decade, I have felt more productive, more guided in my work, and can more easily take on additional responsibility without detriment to well-being. You might have noticed that most of these points come down to identifying a locus of control. The software engineer who feels in control of his or her work and has mental tricks for handling uncertainty and stress is more prepared to deal with said uncertainty, and over time is more productive and happy. Even in the case where happiness is related to disposition,<sup>22</sup> by way of being mindful of these mental strategies we can change the way that we think, and arguably change our disposition.<sup>6</sup> It should also be noted that although these points of discussion are especially relevant for remote software engineering, they can easily be extended beyond this domain of work. These points offer a refreshing idea that success and productivity does not happen to us, but is something that we choose to create. □

### References

- Bélangier, F. and Watson-Manheim, M.B. Virtual teams and multiple media: Structuring media use to attain strategic goals. *Group Decision and Negotiation* 15, 4 (July 2006), 299–321.
- Clampitt, P.G. and Downs, C.W. Employee perceptions of the relationship between communication and productivity: A field study. *The Journal of Business Communication* 30, 1 (Jan. 1993), 5–28.
- Coaston, S.C. Self-care through self-compassion: A balm for burnout. *Professional Counselor* 7, 3 (2017), 285–297.
- Coo, C. and Salanova, M. Mindfulness can make you happy-and-productive: A mindfulness controlled trial and its effects on happiness, work engagement and performance. *J. Happiness Stud.* 19, 6 (Aug. 2018), 1691–1711.
- Cropanzano, R. and Wright, T.A. When a “happy” worker is really a “productive” worker: A review and further refinement of the happy productive worker thesis. *Consulting Psychology Journal: Practice and Research* 53, 3 (2001), 182–199.
- Davidson, R.J. and Begley, S. *The Emotional Life of Your Brain: How Its Unique Patterns Affect the Way You Think, Feel, and Live—and How You Can Change Them*. Penguin, 2013.
- DiMaria, C.H., Peroni, C. and Sarracino, F. Happiness matters: Productivity gains from subjective well-being. *J. Happiness Stud.* 21, 1 (Jan. 2020), 139–160.
- Dingsøyr, T. et al. A decade of agile methodologies: Towards explaining agile software development. *J. Syst. Softw.* 85, 6 (June 2012), 1213–1221.
- Erdil, O. and Gülen Ertoşun, O. The relationship between social climate and loneliness in the workplace and effects on employee well-being. *Procedia—Social and Behavioral Sciences* 24 (Jan. 2011), 505–525.
- Freedman, J.L. and Fraser, S.C. Compliance without pressure: The foot-in-the-door technique. *J. Pers. Soc. Psychol.* 4, 2 (Aug. 1966), 195–202.
- Gant, W. Remote-First: A Guide for Organizations—Simple Programmer. (2019); <https://bit.ly/3vzWCjr>.
- Gardner, B., Lally, P. and Wardle, J. Making health habitual: The psychology of ‘habit-formation’ and general practice. *Br. J. Gen. Pract.* 62, 605 (Dec. 2012), 664–666.
- Guadagno, R.E. et al. When saying yes leads to saying no: Preference for consistency and the reverse foot-in-the-door effect. *Pers. Soc. Psychol. Bull.* 27, 7 (July 2001), 859–867.
- Hakes, T. How to Measure Developer Productivity. (2019); <https://bit.ly/3qWH5q8>
- Hellweg, S.A. and Phillips, S.L. Communication and productivity in organizations. *Public Productivity Review* 6, 4 (1982), 276–288.
- Igic, I. et al. Daily Self-Compassion during work: A daily diary study. (2017).
- Irwin, P. Knowmail. (Jan. 2018).
- Izraeli, D.M. and Jick, T.D. The art of saying no: Linking power to culture. *Organization Studies* 7, 2 (Apr. 1986), 171–192.
- Jabljn, F.M. and Sussman, L. An exploration of communication and productivity in real brainstorming groups. *Hum. Commun. Res.* 4, 4 (June 1978), 329–337.
- Javed, T. Impact of employee ownership on an organizational productivity: A mediating role of psychological ownership. *Academy of Accounting and Financial Studies Journal* 22, 2 (Jan. 2018), 1–12.
- Katz, D.S. et al. Community organizations: Changing the culture in which research software is developed and sustained. *Computing in Science Engineering* 21, 2 (2019), 8–24.
- Ledford, G.E. Comment: Happiness and productivity revisited. *Journal of Organizational Behavior* 20, 1 (1999), 25–30; <https://bit.ly/3vDtNCS>
- Lurey, J.S. and Raisinghani, M.S. An empirical study of best practices in virtual teams. *Information & Management* 38, 8 (Oct. 2001), 523–544.
- Mervosh, S., Lu, D. and Swales, V. See which states and cities have told residents to stay at home. *The New York Times* (Mar. 2020).
- Neal, D.T., Wood, W. and Quinn, J. Habits—A Repeat Performance. *Curr. Dir. Psychol. Sci.* 15, 4 (Aug. 2006), 198–202.
- Oliveira, E. et al. How have software engineering researchers been measuring software productivity?—A systematic mapping study. In *Proceedings of the 19th International Conference on Enterprise Information Systems* (Porto, Portugal). SCITEPRESS—Science and Technology Publications, (2017), 76–87.
- Opitz, I. and Hinner, M.B. Good internal communication increases productivity. Technical Report. Freiburger Arbeitspapiere. 2003.
- Oswald, A.J., Proto, E. and Sgroi, D. Happiness and productivity. *Journal of Labor Economics* 33, 4 (2015), 789–822. <https://doi.org/10.1086/681096> arXiv:<https://doi.org/10.1086/681096>
- Pauleen, D.J. and Yoong, P. Facilitating virtual team relationships via Internet and conventional communication channels. *Internet Research* 11, 3 (Jan. 2001), 190–202.
- Pierce, J.L. and Rodgers, L. The psychology of ownership and worker-owner productivity. *Group & Organization Management* 29, 5 (Oct. 2004), 588–613.
- Robertson, I. and Cooper, C. *Well-Being: Productivity and Happiness at Work*. Palgrave Macmillan, London.
- Seppälä, E. and King, M. Burnout at work isn't just about exhaustion. It's also about loneliness. *Harvard Business Review* (June 2017).
- Siniaalto, M. and Abrahamsson, P. A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage. (Nov. 2017); arXiv:1711.05082 [cs.SE]
- Sochat, V. The Sadness of the Open Source Developer. (2017); <https://bit.ly/3vvtRX>
- Umstot, D.D., Bell, C.H. and Mitchell, T.R. Effects of job enrichment and task goals on satisfaction and productivity: Implications for job design. *J. Appl. Psychol.* 61, 4 (Aug. 1976), 379–394.
- Williams, S.E. and Braun, B. Loneliness and social isolation—A private problem, a public issue. *Fam. Consum. Sci. Res. J.* 111, 1 (Feb. 2019), 7–14.
- Wood, W. and R niger, D. Psychology of habit. *Annu. Rev. Psychol.* 67 (2016), 289–314.
- Wright, S.L. *Loneliness in the Workplace*. (2005).

**Vanessa Sochat** ([sochat1@llnl.gov](mailto:sochat1@llnl.gov)) is a Computer Scientist at Lawrence Livermore National Lab (LLNL), Livermore, CA, USA. She was previously a Research Software Engineer at the Stanford University Research Computing Center, Stanford, CA, USA.

The author thanks Ruth Marinshaw and colleagues in the Stanford Research Computing Center, her new team at LLNL, along with the many open source communities that she cares deeply about, for their humor and support during these times.

Copyright held by author.

## Viewpoint

# Let's Be Honest

*Seeking to rectify the two mutually exclusive ways of comparing computational power—encoding and simulation.*

**W**E HAVE A serious problem with how we have been teaching computability theory, a central component of the ACM/IEEE computer science curriculum.

Let me explain. For a fair number of years, I taught a computability course. Following the standard curriculum (such as described by Hopcroft and Ullman<sup>14</sup>), and in concert with my colleagues in the field, I made claims on countless occasions that one model of computation is more powerful than another or that two models have the same power of computation. In some cases the argument appealed to ordinary set inclusion, while at other times it involved a notion of simulation via encodings. Imagine my chagrin when I came to realize these two methods of comparison are in fact incompatible!

When two models work with the same entities, simple set inclusion of formal languages or sets of functions is employed naturally by everyone. We teach that finite-state automata recognize the same languages as defined by regular expressions but are strictly weaker than pushdown automata, and we bring palindromes or non-square words as proof positive.<sup>13</sup> Similarly, we assert that primitive recursion (or, equivalently, looping via bounded **for** loops only) is weaker than general recursion (with **while** loops, too) because of the two models, only the latter can compute the Ackermann function.<sup>14</sup>

When, on the other hand, the domains of the models under consideration differ, encodings are required



before they can be compared. For example, Alan Turing, in an appendix to his profound landmark 1936 paper, showed that the lambda-computable functions and the functions that can be computed using his Turing machines are of equivalent computational power. “Standard” machine descriptions (lists of quintuples) were turned into decimal numbers, which in turn were expressed in the lambda calculus as Church numerals. Turing also proved that his machines and general recursion are equipotent.<sup>24</sup> To show that Turing machines can compute all general recursive functions, numbers are normally (and wastefully) represented on the machine tape as a sequence of tally marks in unary.<sup>14</sup>

Unfortunately, the preceding two

methods of comparison, namely inclusion and simulation, can yield mutually exclusive outcomes. The simplest example is the counter machine (a.k.a. Minsky machine, abacus model). Each counter holds a natural number that can be incremented, decremented, and tested for zero (see the sidebar). With only two counters, the model is not even powerful enough to square its input<sup>2,20</sup> or recognize primes.<sup>15</sup> However, if we agree to represent a number  $n$  as the exponential  $2^n$  (a simpler encoding than Gödel numbering of expressions), then, courtesy an ingenious proof by the late Marvin Minsky,<sup>14,17</sup> we find that two counters suffice to compute every computable function. This is why one encounters statements such as:

► It is well known that a finite-state



## ACM Transactions on Computing for Healthcare

*ACM Transactions on Computing for Healthcare* (HEALTH) is a multi-disciplinary journal for the publication of high-quality original research papers, survey papers, and challenge papers that have scientific and technological results pertaining to how computing is improving healthcare.



For further information and to submit your manuscript, visit [health.acm.org](http://health.acm.org)

automaton equipped with two counters is Turing-complete.<sup>9</sup>

► [Minsky proved that a] two-counter machine is universal, and hence has an undecidable halting problem.<sup>16</sup>

Such claims of completeness or universality would be blatantly false were one to subscribe to the set-inclusion sense, whereas they are manifestly true in the simulation sense, which is indeed what Minsky proved. As Rich Schroepel expressed it: “Any counter machine can be simulated by a 2CM, provided an obscure [sic!] coding is accepted for the input and output.”<sup>20</sup> So, I take issue with a statement like this: “The surprising result about counter machines is that two counters are enough to simulate a Turing machine and therefore to accept every recursively enumerable language.”<sup>13</sup> Two-counter machines do simulate Turing machines, but they do not “accept” all recursively enumerable languages in the usual “as-is,” unencoded sense, primes being a prime example.

The point is it behooves teachers to be forthright and forthcoming and to address this inconsistency. We cannot carry on oblivious to the fact that by one of the methods of comparison that we use in our lectures 2-counter machines are strictly weaker than (the complete) 3-counter machines, while by a second method that we also endorse the two are to be deemed equivalent.

### The Solution

All is not lost, thankfully. We can eat our proverbial cake and still have it, provided we invest extra effort.

To begin with, it would be an unmitigated disaster to abandon simulations, since the idea that all our traditional unrestrained models are of equivalent power, despite operating with different entities, stands at the core of computability theory, as enshrined in the Church-Turing thesis. Consequently, as painful as it may seem, we are obliged to give up the inclusion notion for paradigms that compute functions, such as general recursion and primitive recursion—though not for formal languages, as I will explain later.

By “simulation” one usually means there is an (injective, 1-1) encoding  $c$  from the domain of the simulated model  $M$  into the domain of the simu-

lating model  $M'$  such that every function  $f$  computed by the former is mirrored by a function  $f'$  computed by the simulator  $M'$  such that  $f'(c(x_1), \dots, c(x_n)) = c(f(x_1, \dots, x_n))$  for all inputs  $x_1, \dots, x_n$  coming from the domain of  $M$ . The following are textbook quotations:

► To show two models are equivalent, we simply need to show that we can simulate one by the other ... Any two computational models that satisfy certain reasonable requirements can simulate one another and hence are equivalent in power.<sup>22</sup>

► Computability relative to a coding is the basic concept in comparing the power of computation models ... Thus, we can compare the power of computation models using the concept “incorporation relative to some suitable coding.”<sup>23</sup>

But what should be deemed “reasonable” or “suitable” lies in the eyes of the beholder. If we do take this simulation route, and I believe we must, and if we are to have a mathematically satisfying theory of computation, then we are in dire need of a formal definition of allowable encodings  $c$ .

Hartley Rogers elucidated, “The coding is chosen so that it is itself given by an informal algorithm in the unrestricted sense.”<sup>19</sup> This requirement, however valid, is at the same time too informal and potentially too generous. And it is circular, since our goal is to demarcate the limits of effective computation. As Richard Montague complained: “The natural procedure is to restrict consideration to those correspondences which are in some sense ‘effective’ ... But the notion of effectiveness remains to be analyzed, and would indeed seem to coincide with computability.”<sup>18</sup> The only way around its informality would be to agree somehow on a uniform, formal notion of “effective” algorithm that crosses domains (such as Boker and Dershowitz<sup>4</sup>). Still, an “unrestricted” encoding could conceivably enlarge the set of functions that can be computed by the simulating model, as we saw with counter machines and an effective exponential encoding.

What other restrictions, then, should be imposed on encodings? Obviously, we need one and the same encoding  $c$  to work for all simulated functions  $f$ . Were one to examine lone



# Counter Machines

Counter machines are one of the very simplest models of computation.

Think of a collection of bowls of marbles, alongside a heap containing an unlimited supply of more marbles.

An  $n$ -counter machine comes with  $n$  bowls.

A program consists of a list of instructions of the following five simple types:

(1) Place a marble taken from the pile into bowl  $X$ , where  $X$  is a particular bowl.

(2) Remove a marble from bowl  $X$ , and return it to the pile; do nothing if there is nothing in the bowl.

(3) Check if there are no marbles in bowl  $X$ ; if so, continue with instruction  $K$ , where  $K$  is the number or label of one of the instructions in the program.

(4) Continue with instruction  $K$ , unconditionally.

(5) Halt.

The colors and sizes of the marbles do not matter; only the quantity does.

Initially, the bowls have some given number of marbles as input. When and if a program halts, the number of marbles in a designated bowl is the program's output.

For example, the following is a 4-counter program for multiplying the quantities initially in bowls A and B. Bowls C and D start out empty. The product of A and B will be in D at the end. Bowl C serves as a holding area.

**S:** If A is empty, continue at H.

Remove a marble from A.

**L:** If B is empty, continue at R.

Remove a marble from B.

Place a marble in C.

Place a marble in D.

Continue at L.

**R:** If C is empty, continue at S.

Remove a marble from C.

Place a marble in B.

Continue at R.

**H:** Halt.

It is a fact that three bowls suffice to compute any computable single-argument function over the natural numbers, but to compute them all with only two bowls is only possible with an encoding such as  $2^i$  for  $i$ .



itive) encoding that allows one to simulate all the computable functions plus an incomputable one like halting.<sup>3</sup> It turns out, in fact, that simulating the successor function effectively is necessary and sufficient to guarantee that Turing machines cannot simulate (under a single-valued encoding) anything unexpected.<sup>4</sup> And this is all we need for the big picture to remain intact.

On the other hand, with just a bit of effort, one can devise a recursive function (a modification of Ackermann's function) that cannot be simulated by primitive recursion regardless of the encoding.<sup>3</sup> So it thankfully remains true that primitive recursion is strictly weaker than recursion—in the very strong sense that no (injective) encoding whatsoever would endow primitive recursion with full Turing power. Were it not likewise provable that 1-counter machines cannot simulate all recursive functions, statements like “Combining these simulations, we see that two-counter machines are as powerful as arbitrary Turing machines (one-counter machines are strictly less powerful)”<sup>12</sup> would be indefensible.

Turning to formal languages (sets of words over some alphabet), the situation is reversed. Encodings are bad; inclusion is good. Homomorphic mappings may preserve the relative power of most language models (with their purely local impact on the structure of strings), but more general injections or bijections do not. In fact, there is a nefarious bijection between the words of any (nonsingular) alphabet with the disconcerting property that all the regular *plus* all the context-free languages can be recognized by mere finite-state automata. The situation is actually infinitely more intolerable: one can at the same time also recognize countably many arbitrary undecidable languages with vanilla finite automata via such a mischievous bijection.<sup>10</sup>

In the case of languages, then, we are compelled to adhere to straightforward inclusion and ban (even computable) mappings of input strings when comparing the power of language models. Earlier, when dealing with (all) the computable functions, we did have the flexibility of simulating via mappings, but that was because the same mapping is also applied to the full range of possible function outputs.<sup>5</sup>

functions, then it would be easy to come up with a bespoke “deviant encoding” that makes a single uncomputable function appear computable. For another thing, we must insist that the same encoding  $c$  be used both for the inputs  $x_i$  as well as for the output of  $f$ , or else everything can easily go belly-up (pace Butterfield et al.<sup>8</sup>). Ideally, the restrictions would ensure that (unlike for counter machines, or the lambda calculus, for that matter) no allowed encoding can expand the class of computed functions. Specifically, we must preclude the endowing of Turing's ma-

chines or the recursive functions with superpowers (what is termed “hypercomputation”). How can we guarantee this? Shapiro<sup>21</sup> has submitted that for an encoding of (Platonic) numbers to be “acceptable,” the encoded successor function should also be computable by  $M'$ . In other words,  $M'$  must include a function  $s': c(n) \mapsto c(n + 1)$  simulating successor. I agree. But why successor?

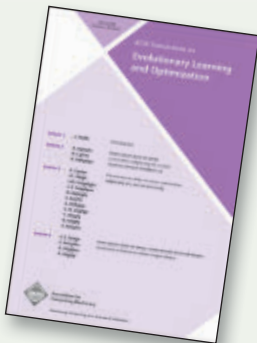
Mercifully, encoding turns out not to be a problem for the usual use cases. Indeed, no encoding whatsoever can break the Turing barrier. Specifically, one can prove that there is no (injec-



Association for  
Computing Machinery

## ACM Transactions on Evolutionary Learning and Optimization (TELO)

*ACM Transactions on Evolutionary Learning and Optimization (TELO)* publishes high-quality, original papers in all areas of evolutionary computation and related areas such as population-based methods, Bayesian optimization, or swarm intelligence. We welcome papers that make solid contributions to theory, method and applications. Relevant domains include continuous, combinatorial or multi-objective optimization.



For further information  
and to submit your  
manuscript,  
visit [telo.acm.org](http://telo.acm.org)

### Related Concerns

There is another ubiquitous use of encodings that similarly requires extra caution. Oftentimes, one wishes to compute a function on objects other than strings or numbers, such as graphs, logical formulae, or computer programs. For that purpose, one must somehow represent those objects in the input/output language of the computational model that is to manipulate them, typically strings or numerals. To quote: “A necessary preliminary to applying our work on computability ... is to code expressions by numbers .... There are many reasonable ways to code finite sequences, and it does not really matter which one we choose.”<sup>6</sup>

To be “reasonable,” however, one needs to be sure that the encoding does not do anything beyond faithfully representing the input.

For example, it is a trivial matter to concoct an encoding of Turing machines that turns an undecidable problem about machines into a readily computable one. Let  $w_0, w_1, \dots$  be an enumeration of all binary strings (over the alphabet  $\{0, 1\}$ ), and let  $M_0, M_1, \dots$  be some enumeration of all Turing machines (over that input alphabet). The following are four typical decidability questions:

$T(i, j)$ : machine  $M_j$  halts on input  $w_i$ .

$H(i)$ : machine  $M_i$  halts on input  $w_0$ .

$U(i)$ : machine  $M_i$  halts on all inputs  $w_0, w_1, \dots$

$D(i)$ : machine  $M_i$  halts on input  $w_i$ .

Halting on a single particular input (like the empty word) is just the parity problem if one reorders a standard enumeration of machines so that the odd-numbered ones halt on that input while the even ones do not. The snag with such an encoding of Turing machines is that it also makes ordinary tasks incomputable. Specifically, once could not modify the code of a given machine to act in some related but different way, because one would need to ascertain the termination behavior of the modified machine. So whether problem  $H$  is decidable or not actually depends on exactly how machines are encoded.

Consider an assertion such as the following:

► One of Turing’s key insights was the Halting Problem  $H$  (which takes an integer  $n$  and outputs  $H(n) = 1$  if and

**It would be an unmitigated disaster to abandon simulations, since the idea that all our traditional unrestrained models are of equivalent power, despite operating with different entities, stands at the core of computability theory, as enshrined in the Church-Turing thesis.**

only if  $n = \langle P \rangle$  is an encoding of a valid [self-contained] program  $P$  and  $P$  terminates) is “undecidable.”<sup>7</sup>

For your usual, straightforward encodings of machine descriptions, the problem is indeed undecidable, but for any number of alternative encodings it becomes decidable. The same goes for the “universal halting” problem  $U$ .

On the other hand, the more basic halting problem,  $T(i, j)$ , which asks about the behavior of the  $i^{\text{th}}$  machine on the  $j^{\text{th}}$  possible input, is undecidable regardless of how machines are encoded or how inputs are enumerated. Nevertheless, one must be careful how pairs  $\langle i, j \rangle$  are encoded for models of computation—such as run-of-the-mill Turing machines—that allow only a single input representing both  $i$  and  $j$ . Similarly, the “diagonal” language  $D$ , consisting of the indices of those machines that halt when the input string has the same index in its enumeration as does the machine in its encoding, is not computable for any and all machine encodings and string enumerations. So it is true that “no encoding [of all Turing machines as numbers] can represent a TM  $M$  such that  $L(M) = L_d$  [the diagonal language],” as claimed in

Hopcroft et al.,<sup>13</sup> but the same immunity to encoding does not hold true for the collection of machines that accept nothing ( $L_c$ ).<sup>13</sup> Regrettably, no textbook I have seen clarifies which encodings of machines are valid and for what purpose and why. Nothing in the following remark, for example, precludes a representation from incorporating a finite amount of uncomputable information about the represented machine, such as whether it always terminates or halts on a specific input:

► The details of the representation scheme of Turing machines as strings are immaterial [as long as]: (1) We can represent every Turing machine as a string. (2) Given the string representation of a Turing machine  $M$  and an input  $x$ , we can simulate  $M$ 's execution on the input  $x$ .<sup>1</sup>

The standard part of a malicious string encoding would allow one to simulate execution as usual, while tacked-on extras can allow an algorithm to decide otherwise undecidable questions about them. Overzealous encoding is not, however, a problem in programming languages that pass unadulterated programs as arguments, sans encoding.

As a final comment, when it comes to complexity comparisons, everyone realizes that representation is an issue to be taken into account, but the requirements remain vague: “The intractability of a problem turns out to be essentially independent of the particular encoding scheme ... used for determining time complexity ... It would be difficult to imagine a ‘reasonable’ encoding scheme for a problem that differs more than polynomially from the standard ones ... What we mean here by ‘reasonable’ cannot be formalized ...”<sup>11</sup>

It would seem to me that standard string and image compression schemes are perfectly reasonable encodings, despite reducing size exponentially in many cases. In any event, a formal, principled definition of “reasonableness” is still sorely lacking for the theory of complexity. (But see Boker and Dershowitz<sup>5</sup> for one proposal.)

## Takeaway

To recapitulate the main points of the problem raised here:

► Every single course in automata or

computability utilizes set inclusion as the means of comparing the computational power of different formalisms for language definition.

► Virtually every such course claims equivalence of a wide variety of models of computation in support of the Church-Turing thesis, an equivalence that is based on mutual simulations.

► These two notions are logically incompatible as we have witnessed.

► No textbook nor any instructor I have encountered recognizes, let alone addresses, this fundamental inconsistency.

At a bare minimum, then, we must make the following changes in the manner this subject is traditionally taught:


► One should use set inclusion only as a means to compare classes of formal languages, such as in the demonstration that context-free grammars are a strictly more inclusive formalism than are regular expressions.

► We should never use set inclusion to compare the power of primitive recursion with general recursion, or **for**-loop programs with **while**-loop ones, or one-counter machines with two counters, without mentioning that it has in fact been demonstrated that the one can also not *simulate* all of the other.

► Instructors ought to emphasize that one must always be careful with encodings, as they easily alter computational power, while pointing out it has been proved this is not an issue for the usual use case of Turing-level computability.

► One should definitely avoid using halting-on-empty-tape, or empty-language acceptance, or similar problems as fundamental examples of undecidability, as their decidability is encoding-dependent. Instead, we need to explicate the subtle role of input encodings when reducing the standard two-input halting problem to those other problems.

► We should be cautious to never say or imply that two-counter machines recognize all recursively enumerable languages (they do not), nor that they *compute* (as opposed to *simulate*) all Turing-computable functions.

► One should not choose the lambda calculus as a primary exemplar of a fully empowered computational model (since it simulates *more* than it computes). 

## References

- Barak, B. *Introduction to Theoretical Computer Science*. 2020. Online text (version of Dec. 25, 2020); <https://bit.ly/3bZnLvi>
- Barzdins, J.I. Ob odnom klasse machin turinga (machiny minskogo) [On one class of Turing machines (Minsky machines)]. *Algebra i Logika [Algebra and Logic]* 1, 6 (1963), 42–51. In Russian.
- Boker, U. and Dershowitz, N. Comparing computational power. *Logic Journal of the IGPL* 14, 5 (2006), 633–648; <https://bit.ly/3cEzd99>
- Boker, U. and Dershowitz, N. The Church-Turing thesis over arbitrary domains. In A. Avron, N. Dershowitz, and A. Rabinovich, Eds., *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85<sup>th</sup> Birthday*, volume 4800 of Lecture Notes in Computer Science, Springer, Berlin, 2008, 199–229; <https://bit.ly/3eUun99>
- Boker, U. and Dershowitz, N. Honest computability and complexity. In E. Omodeo and A. Policriti, Eds., *Martin Davis on Computability, Computational Logic & Mathematical Foundations*, volume 10 of Outstanding Contributions to Logic Series, Springer, Cham, Switzerland, 2017, 153–175; <https://bit.ly/3cH8589>
- Boolos, G.S., Burgess, J.P., and Jeffrey, R.C. *Computability and Logic*. Cambridge University Press, Cambridge, U.K., 4<sup>th</sup> edition, 2002.
- Braverman, M. Computing with real numbers, from Archimedes to Turing and beyond. *Commun. ACM* 56, 9 (Sept. 2013), 74–83.
- Butterfield, A., Ngondi, G.E., and Kerr, A., Eds. Machine simulation entry. *A Dictionary of Computer Science*. Oxford University Press, Oxford, U.K., 7<sup>th</sup> edition, 2016.
- D'Souza, D. and Shankar, P. *Modern Applications of Automata Theory*. World Scientific, River Edge, NJ, 2011.
- Endrullis, J., Grabmayer, C., and Hendriks, D. Regularity preserving but not reflecting encodings. In *30<sup>th</sup> Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (Kyoto, Japan, July 2015. IEEE Computer Society), 535–546; <https://bit.ly/3cDSR3M>
- Garey, M.R. and Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York, NY, 1979.
- Harel, D., Kozen, D., and Tiuryn, J. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.
- Hopcroft, J.E., Motwani, R., and Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation*. Pearson Education, Boston, MA, 3<sup>rd</sup> edition, 2007.
- Hopcroft, J.E. and Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.
- Ibarra, O.S. and Trân, N.Q. A note on simple programs with two variables. *Theor. Comput. Sci.* 112, 2 (1993), 391–397.
- Lipton, R.J. and Regan, K.W. Minsky the theorist. (Jan. 27, 2016); <https://bit.ly/2P45zRn>
- Minsky, M.L. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, NJ, 1967.
- Montague, R. Towards a general theory of computability. *Synthese* 12, 4 (1960), 429–438.
- Rogers, Jr., H. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York, NY, 1966.
- Schroepfel, R. A two counter machine cannot calculate  $2^n$ . Technical report, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA, 1972; <ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-257.pdf>
- Shapiro, S. Acceptable notation. *Notre Dame Journal of Formal Logic* 23, 1 (1982), 14–20.
- Sipser, M. *Introduction to the Theory of Computation*. Thomson, Boston, MA, 2<sup>nd</sup> edition, 2006.
- Sommerhalder, R. and van Westrheden, S.C. *The Theory of Computability: Programs, Machines, Effectiveness and Feasibility*. Addison-Wesley, Workingham, England, 1988.
- Turing, A.M. Computability and  $\lambda$ -definability. *The Journal of Symbolic Logic* 2, 4 (1937), 153–163; <https://bit.ly/3eOUEbF>

Nachum Dershowitz (nachum@tau.ac.il) is a Professor (Emeritus) in the School of Computer Science, Tel Aviv University, Ramat Aviv, Israel.

The author thanks Jörg Endrullis for his perceptive comments and Centrum Wiskunde and Informatica (CWI) for its hospitality.

Copyright held by author.

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

## Legal considerations and broader implications.

BY JATINDER SINGH, JENNIFER COBBE,  
DO LE QUOC, AND ZAHRA TARKHANI

# Enclaves in the Clouds

WITH ORGANIZATIONAL DATA practices coming under increasing scrutiny, demand is growing for mechanisms that can assist organizations in meeting their data-management obligations. In this context, trusted execution environments (TEEs), also known as *enclaves* or *secure enclaves* (the terms are used interchangeably here), are the subject of much interest.

TEEs provide hardware-based mechanisms (enclaves) with various security properties for assisting computation and data management. Broadly, TEEs are concerned with the confidentiality and integrity of data, code, and the corresponding computation. Because the main security properties come from hardware,

certain protections and guarantees can be offered even if the host privileged software stack (for example, the operating system or hypervisor) is curious, vulnerable, or compromised.<sup>3</sup>

The technology is growing in prominence. There are a range of hardware offerings, and the major cloud providers are beginning to feature enclaves in their services.

### A Technology of Current Interest

TEEs have a rich history, based on long-established security principles. In the early 2000s the use of such technology gained some attention by way of “Trusted Computing” initiatives, which proponents argued were advantageous because they provided the means to enforce particular systems operations. These initiatives were met with controversy, given that the technology effected a loss of user control in the interests of vendors (see the “Application Contexts” section later in this article).<sup>4</sup>

Today the growing awareness of the likelihood and costs (financial, reputational, legal) of security and privacy breaches, along with the increased legal and regulatory attention to data and technology more generally, has resulted in more interest in TEEs’ potential to assist organizations in managing issues of data and its processing.

One key driver for this interest is the dominance of the cloud (that is, service-oriented computing in multiparty environments). This is because enclaves can provide a technical means for enabling isolation and assurance against the “untrusted” host (that is, the provider), other *tenants* (customers of cloud services), and other parties with which interactions occur. The industry has recognized this potential for enclaves to shield a tenant’s data and processing from others, including the cloud provider, by effectively removing the provider from the *trust chain*.<sup>16,22</sup> Reflecting this, terms such as *confidential computing* are used to highlight cloud-related business cases for such technology.<sup>13,16</sup> Enclaves can also work in a cloud context to assure



the integrity of systems, that certain processing occurred, that configuration and management policies are applied,<sup>8</sup> and that billing accurately reflects the resources consumed.<sup>1</sup>

Directly related to this interest in TEEs is the attention being paid to the privacy and security considerations of data analytics and ML (machine learning), which is increasingly occurring in the cloud. Work has been ongoing on enclaves for enabling secure multiparty computation on untrusted platforms,<sup>19</sup> as well as for privacy in large-scale data

analytics (for example, SGX-PySpark<sup>12</sup>) and enabling more privacy-aware machine learning as a service.<sup>11</sup>

This article focuses on TEEs for the cloud, because the cloud represents the dominant form of providing applications and data processing infrastructure.

### Enclaves and the Law

The legal requirements and obligations of those designing, deploying, or using technical systems for processing personal data stem from a variety of sources. A prominent example is the

European Union's GDPR (General Data Protection Regulation), which is both high profile, and, though an EU regulation, also globally relevant—it not only operates across territories, but also serves as a model for the data protection regimes of many other jurisdictions, including the CCPA (California Consumer Privacy Act). Various aspects of GDPR have direct technical implications, including in relation to security, organizational accountability, and data protection by design.

GDPR has received much attention,

given the potentially significant penalties for noncompliance. Still, it is only one example—legal obligations with technical implications can flow from other sources, such as contractual agreements or product-liability regimes.

It follows that demand is growing for mechanisms that assist organizations in meeting their legal responsibilities. TEEs are seen as one technology to help with such concerns. Indeed, vendors have explicitly characterized the technology as enabling compliance with various regulations (see the “Cloud Computing” section later in this article). Despite these assertions, however, the extent to which enclaves assist with legal obligations requires exploration. So far, neither the implications of such protections in a service-provider context nor the potential for this technology to reinforce the dominance of the large providers have been given much consideration.

Accordingly, this article provides a cross-disciplinary analysis of how the functionality of TEEs relates to legal obligations in the context of the cloud. The article presents a high-level overview of the common security properties provided by enclaves and introduces some of the relevant legal landscape. The focus is on data protection (specifically the GDPR), as that represents a key area of law relating to data concerns.

The article then explores how enclaves can assist an organization’s legal obligations, responsibilities, liabilities, and compliance concerns in a service-provider context, considering the benefits for the parties involved. It shows

that although TEEs might assist in managing some legal obligations, they will not resolve issues of data protection compliance—despite what some marketing materials might imply.

Further, despite some benefits for tenants, the technology is shown to favor the interests of the service provider (similar to how previous instances of the technology were seen as protecting vendor interests), and the implications of this are considered. Finally, the technology’s potential as a foundation for enabling compliance and accountability regimes is considered.

### Technology Overview

At a high level, a TEE (or *secure enclave* or just *enclave*) can be described as an execution environment that isolates the code and data inside it from the host’s software. The security guarantees are backed by hardware (for example, CPUs), protecting the confidentiality and integrity of the data and code within. Cryptography often plays a key role in the security guarantees.

Conceptually, Trusted Computing is not new. In the 1980s the U.S. Department of Defense Trusted Computer System Evaluation Criteria,<sup>28</sup> known as the “Orange Book”, provided the basic principles of trust for sensitive applications. This included a definition for a TCB (trusted computing base) and generally established the need for minimalism, compartmentalization, and the verification of systems-critical hardware/software modules. Efforts to develop trusted computing technology continue today, both in terms of defin-

ing new requirements and the practicalities of their implementation. Many hardware vendors and service providers now incorporate such technology into their offerings (see the “Application Contexts” section later in this article).

**Security properties.** TEEs reflect a broad set of security features and concerns, as outlined here. The accompanying figure presents the purported features of some prominent implementations.

*In-enclave protection.* TEEs can provide a shielded execution environment that protects (to various degrees) data and code against unauthorized access and modification by external software. This includes protections from the host operating system or hypervisor that otherwise would generally have the potential for full control of and visibility into the tenant’s code and data. Note that of those listed in the figure, only Intel SGX and AMD SEV support runtime memory encryption.

*Remote attestation.* This feature securely verifies the internal state of a platform and the in-enclave application. It gives certain guarantees about the enclave before the transfer of “secrets” (keys/certificates, data, or code) to a remote platform and enables verification that the code and data running in the enclave are correct—for example, not modified or tampered with. It can be achieved either statically at boot time or dynamically at runtime to establish a *dynamic root of trust*, which can serve as a foundation for providing various systems-level guarantees.

*Sealing.* Persisting an enclave’s secrets to a disk in a protected manner enables the retrieval and reinstantiation of in-enclave state, perhaps as a result of system reboot, power interrupt, or being in line with application requirements. Sealing requires binding the secrets to a specific enclave’s state or device.

*Secure boot.* A secure boot defines and ensures a chain of trust regarding the enclave images, operating system components, and configurations. Often this entails loading immutable and verified images to an enclave.

*Isolated peripherals.* Secure peripheral sharing (for example, NICs) and isolated I/O paths are enabled via fine-grained control over hardware resources or using cryptography-based mechanisms.

#### Overview of some prominent TEEs.

| TEE                        | Target ISA       | Security Features |   |   |   |   |   |   |
|----------------------------|------------------|-------------------|---|---|---|---|---|---|
| TPM/vTPM                   | Multiple Targets | ⓪                 | ● | ● | ● | – | – | – |
| Intel TXT                  | x86_64           | ●                 | ● | ● | ● | – | – | – |
| Intel SGX                  | x86_64           | ●                 | ● | ● | – | – | – | – |
| ARM TrustZone              | ARM              | ●                 | – | ⓪ | ● | ● | ⓪ | – |
| Sanctum/KeyStone/MultiZone | RISC-V           | ●                 | ● | ● | ● | – | ⓪ | ● |
| AMD SEV                    | AMD x86          | ●                 | ● | ● | – | – | – | – |

- provides property
- ⓪ supports partially
- does not support

In-enclave protection  
 Remote attestation  
 Sealing  
 Secure boot  
 Isolated peripherals  
 Side-channel resistance  
 Open source hardware

*Side-channel resistance.* This feature mitigates information leakage through various software- or hardware-based side-channel attacks.

**Technical considerations.** Leveraging TEEs raises several practical considerations, some of which we outline below.

*Usability.* Properly leveraging the capabilities of enclaves requires expertise, including an understanding of the particular technology's role, features, and limitations in a specific context. TEEs operate only to directly protect that code and data residing within the enclave, although they are often used as a foundation to provide broader systems protections and guarantees. Developing enclave-assisted applications involves considering which aspects require and are most amenable to protection and compartmentalizing the application to use the TEE's functionality appropriately. Tools are being developed to support running legacy applications inside enclaves without modification (for example, SCONE<sup>5</sup>) or running applications on heterogeneous enclaves (Microsoft's OpenEnclave<sup>20</sup>). One must also determine how best to manage and protect the code and data running outside the enclave.

*A growing TCB.* Traditionally, enclaves were designed to support very specific and small operations (for example, cryptography or key management). The ideal is for enclaves to contain an uncomplicated and verifiable in-enclave software stack, with minimal interactions with the outside environment, so as to offer strong security guarantees and assurances.<sup>28</sup> In practice, however, enclaves are used in much more complex application architectures such as web services, auditing, and privacy-aware data analytics.

The risks associated with service provision and software stack vulnerabilities, and the difficulties in engineering and refactoring applications into trusted and untrusted compartments to leverage enclaves, have led to enabling in-enclave support for a larger portion of an applications' dependencies, including entire language runtimes and operating systems.<sup>27</sup> For the cloud, a larger TCB allows more complex applications to be deployed by tenants, thereby making the cloud more attractive to them. Again, however, security guarantees and assurances are

generally significantly lowered by having larger and more complex TCBs.<sup>29</sup>

*Security vulnerabilities.* TEE hardware provides only particular security properties—the software or system design must properly leverage these to benefit. Insecure or buggy in-enclave code will often not (depending on the issue) mitigate security risks and, indeed, could actually render a worse outcome as the enclave works to protect the problematic computation (for example, acting as an enclave that protects malware<sup>23</sup>). As already mentioned, increasing the size and complexity of what resides in the enclave and various layers of interactions with the untrusted world causes a range of vulnerabilities that can jeopardize the enclave's security properties and guarantees.<sup>26,29</sup> Further, there may be vulnerabilities regarding the TEE itself (architecture or supporting services)—various attacks have been demonstrated<sup>18</sup>—and, as a commodity technology, an enclave issue can result in insecurity at scale, as the Spectre/Meltdown processor vulnerabilities demonstrate (see [www.meltdownattack.com](http://www.meltdownattack.com)).

Note that many enclave offerings are proprietary and closed (to varying degrees), meaning that it can be difficult to ascertain precisely what is offered and how they work, and to assess and validate their claims. Moves toward more open access to software and hardware stacks would assist such interrogations.

*Standardization.* The growing adoption of enclaves means they will increasingly form part of a wider technical ecosystem. Therefore, standardization is important for facilitating adoption, consistency in functionality and use, as well as allowing interoperability and management in increasingly heterogeneous environments. Various standardization initiatives are under way, including by GlobalPlatform, the Trusted Computing Group, and the Confidential Computing Consortium, with many different areas warranting consideration.

### Application Contexts

As a general-purpose security technology, there are many potential applications for TEEs. In the early 2000s came the various “Trusted Computing” initiatives, which aimed to enforce particular behavior and functionality on

a machine. They were controversial, seen as technologies serving vendor (rather than user) interests.<sup>4,21</sup> Concerns included power asymmetries and anti-competitiveness, given that trusted computing effectively enabled vendors to control what users could and could not do on their machines, thereby facilitating DRM (digital rights management), the reinforcement of vendor ecosystems (lock-in), and so on.<sup>4</sup> Note that this was in an era when software and processing predominantly occurred locally, on user hardware; however, as we explore, the concerns about who the technology benefits, as well as its wider implications, remain.

Today a prominent use of enclaves is providing security guarantees and assurances regarding data and its processing, and how they relate to broader computational infrastructures. The focus here is on the use of TEEs in the context of cloud computing, where various technical capabilities are offered as a service in a multiparty environment. Note, however, that enclaves are also increasingly embedded within a wide range of systems to underpin security functionality,<sup>9</sup> and will likely play a role in supporting edge and IoT (Internet of Things) ecosystems.<sup>24</sup>

**Cloud computing.** TEEs are increasingly marketed as a means of helping mitigate some of the risks associated with the cloud. Cloud computing entails the provision of computing functionality, resources, and infrastructure, delivered as a service.<sup>17</sup> The cloud encompasses several parties: the cloud service *provider*, which offers the particular service(s) to *tenants*, who are the customers and direct consumers of the services.

A vast range of cloud services are available; in general, tenants use the cloud to provide or manage (aspects of) their technical infrastructure, including, the storage and processing of data. That is, cloud providers offer the infrastructure and components for supporting, hosting, and running customer applications, services, data analytics/ML, and so forth. Cloud providers leverage economies of scale, whereby the services and resources they provide may be shared among tenants. Often the cloud uses a pay-per-use model that is attractive for tenants, given that they can access compute resources on demand, as

needed, typically at a substantially lower cost and with lower overhead than maintaining such services in-house.

Security is a key concern in the cloud. Enclaves can allow a tenant's code, data, and processing to be better isolated and protected: from other tenants on the shared infrastructure; and, importantly, and from the cloud provider itself—though a provider might host particular data or computation, enclaves can limit a provider's access to such. This is important, particularly where the data, code, computation, or analytics might be personal or otherwise sensitive. Further, TEEs can also be used to give guarantees that certain processing occurred, certain management policies were enforced, and the like.

Note that the service provider integrates the enclave technology into its service infrastructure. In this context, TEEs provide certain controls and guarantees over the providers' systems, as opposed to the situations where a TEE operates as a control mechanism over the user's hardware/system.

**Enclaves as a service.** TEEs can be offered by service providers at different levels of technical abstraction, depending on the features of the particular TEEs employed and the particular use cases. In practice, enclave offerings tend to take the following forms: Trusted/trustworthy virtual machines (for example, Intel's Trust Domain Extension (TDX) or AMD's SEV (Secure Encrypted Virtualization)); an in-enclave LibOS (library operating system, for example, Graphene-SGX<sup>27</sup>) or container (for example, SCONE<sup>5</sup>); enclave-assisted applications and analytics frameworks (for example, SGX-PySpark,<sup>12</sup> and VC3 [Verifiable Confidential Cloud Computing]<sup>22</sup>); or a limited SDK and programming framework (for example, Google's Asylo and Microsoft's Open Enclave).

Enclaves can underpin a number of services in a cloud context. First, they can be used to support and protect specific operations—for example, for cloud-based key or credential storage, management, and verification. Alternatively, the TEEs of the cloud provider can be used to support custom applications, as developed by the tenants themselves, or more generally where a whole application, container, or operating system might be enclave protected (see the earlier section on “Technical

Considerations”). Allowing larger applications as well as data processing or analytics operations to run within an enclave may be attractive to prospective tenants. These offerings can support a range of tenant requirements, while serving to make some of the technology's functionality more widely accessible, thus recognizing that leveraging enclaves can require some expertise.

The major cloud providers already offer enclave-backed infrastructure services. Indeed, providers see the emergence of TEEs as facilitating the growth of cloud computing. With industry materials describing the technology as enabling “confidential computing” and removing the provider from the “trust loop,”<sup>13,16,25</sup> enclave-backed offerings attempt to encourage otherwise risk-averse customers (tenants) to make more use of cloud services because of the security guarantees and assurances that the technology can provide.

In particular, the technology can enable computation to happen on provider infrastructure, without that provider having plaintext access to the code, data, and processing. This can be attractive where the nature of the data or its processing is highly confidential, personal, or subject to other constraints. The services are also marketed as providing integrity guarantees (that is, assurance the data has not been unduly modified, that the correct processing actually occurred, that machines were configured correctly, and so forth).

TEE offerings are also promoted to tenants as supporting the management of legal obligations. For example, Intel's description of IBM's cloud explicitly describes the technology as assisting with GDPR compliance.<sup>25</sup>

In summary, enclaved-backed service offerings provide incentives for customers to use cloud services. Prospective tenants can leverage the cloud providers' significant resources and technical capacity to take advantage of the security and possible compliance benefits of the technology (and related services), while avoiding the potentially significant overheads of in-house infrastructure management.

### Legal Context

The design, deployment, and use of systems, and the management and processing of data, are the subjects of

growing legal and regulatory attention. Various legal obligations and requirements—arising from data protection, security, and product safety legislation, as well as from contractual obligations and other liability concerns—are increasingly relevant for technical systems and will encourage technological responses. The adoption of technical measures (including for security) to meet legal obligations is often driven by the fact that failure to do so can result in penalties and restrictions on operations, among other repercussions.

The focus here is on data protection law, given its particular prominence and general relevance to systems and data concerns. Note, however, that other laws can be relevant and can also impose legal obligations and responsibilities regarding security and data management (see “Law as a Technology Driver”).

**General Data Protection Regulation.** This article's discussion is in the context of the EU's GDPR, given its profile and breadth. Though the GDPR is EU law, it has worldwide relevance; it can apply to entities outside the EU that process data on EU individuals (GDPR Art.3), and it's becoming a de facto worldwide standard, with other jurisdictions establishing laws along similar lines.

*GDPR's framework.* GDPR applies to the processing of *personal data*—that is, any data relating to an individual who can be directly or indirectly identified (GDPR Art.4(1)). This includes not only names or IDs, but also location data, device identifiers, online identifiers, IP addresses, and so on. Similarly, personal data can also exist where an individual may be identified from some combination of factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that individual. Important for this discussion is that encrypted personal data is often still considered personal data and is thus subject to GDPR (even if the entity holding the data does not hold the decryption keys or can't otherwise access the plaintext).<sup>7</sup> This is because anyone with access to the plaintext could use it directly or indirectly to identify an individual.

As defined by GDPR, *processing* includes any operation or set of operations performed on personal data,



including, for example, collection, storage, adaptation or alteration, retrieval, and use (GDPR Art.4(2)). Note that this is broader than the standard computer science use of the term, as it entails far more than just computation.

GDPR establishes that those involved in processing personal data act as data *controllers* or data *processors*. Controllers are those entities that determine the means and purposes of processing personal data (GDPR Art.4(7)). Processors, on the other hand, are entities that process personal data on behalf of and under the instruction of a controller (GDPR Art.4(8)). In simple terms, controllers generally decide why and how the processing happens, while processors carry out particular processing operations for the controllers.

Whether one is a controller or processor is important, as it has a direct bearing on legal obligations and responsibilities. In many instances a company providing an application is the data controller, while others providing various technical services are usually data processors (that is, regarding the cloud, for example, often the cloud provider is the controller, though this will depend on the specific circumstances<sup>17</sup>). The role taken by a particular entity, however, is determined on the basis of who is actually doing what, regardless of any contract or other agreement between the entities.

*Security-related obligations.* GDPR places positive obligations on controllers and processors in relation to security and data management. GDPR mandates the principles of data protection by design and by default (GDPR Art.25), whereby technologies, applications, and processes need to be designed from the earliest stages with data protection in mind. GDPR also requires that both data controllers and processors take “technical and organizational measures” to ensure a level of security appropriate to the risks incurred in processing personal data (GDPR Art.32; Recital 83). These risks may relate to, among other things, loss, alteration, or unauthorized data disclosure or access.

*Enforcement and penalties.* The penalties for GDPR noncompliance can be significant. Supervisory authorities (regulators) are empowered to take various actions against both controllers and processors for failures to comply

## GDPR places positive obligations on controllers and processors in relation to security and data management.

with GDPR’s requirements, including banning them from processing data (GDPR Art.58(2)), which could prove fatal to an organization. Both controllers and processors are liable for administrative fines of up to the greater of €10M or 2% of global turnover for certain breaches, including the failure to meet GDPR’s security requirements, or €20M or 4% of turnover for some classes of other GDPR breaches (GDPR Art.83(4)). In this way, GDPR seeks to be effective against even the largest companies.

**Law as a technology driver.** The GDPR is an example showing how law can introduce legal requirements and obligations that impact system design, and where compliance has technical implications. Though data protection law is particularly prominent and broadly applies to personal data processing concerns, other laws can also raise security- and data-related considerations. (For reasons of space these are not explored here.) Various parties in a technical ecosystem will face legal requirements and obligations, whether through contract, liability, statute, or regulation.

With the prospect of potentially significant penalties for noncompliance, whether stemming from data protection requirements or elsewhere, there are strong incentives to implement risk-mitigation measures beginning from the earliest stages of system design. Therefore, as a security technology offering a broad range of capabilities, TEEs have received considerable attention as a *possible* means for helping systems designers and operators meet legal obligations and requirements.

### Legal Relevance and Discussion

TEEs might assist in meeting legal obligations primarily by: practically reducing risks relating to security (and data governance) or providing evidence (technical or otherwise) of steps undertaken to mitigate those risks, or demonstrating that the appropriate actions were taken.

In this way, TEEs can help meet some of the requirements set out in GDPR, as well as those from other regulations. Enclaves might also assist in meeting the security obligations that can arise from contractual relationships. Even where security requirements are not explicit


in law, security-oriented requirements may arise from best practices, adherence to industry standards, and so on. TEEs can assist with meeting these requirements in much the same way as they can assist with meeting similar obligations established by law.

The following sections consider in more detail how TEEs relate to obligations under GDPR in a cloud context. Note that some of the discussion may be applicable more widely.


**Security-related requirements.** Recall that GDPR requires data controllers and processors to take the appropriate “technical and organizational measures” to ensure a level of security appropriate for the risks of data processing (see “Security-related Obligations”). In determining the measures to employ, parties should take into account the current technological state of the art; the scope, context, nature, and purpose of the data and its processing and the associated risks; and issues around loss, alteration, unauthorized disclosure and access, among others (GDPR Art.32; Recital 83). GDPR explicitly envisages that appropriate measures might include encrypting personal data and mechanisms to ensure the confidentiality and integrity of processing (GDPR Art.32; Recital 83).

Because of their significant resources, it is argued that cloud providers are often better placed to deal with security concerns than companies managing their infrastructure in-house. Moreover, GDPR requires that processors can be used only if they provide sufficient guarantees that they have implemented measures ensuring GDPR-compliant processing, including in relation to security obligations (GDPR Art.28). In this way, enclaves, beyond potentially further enhancing a provider’s security posture, could provide an additional mechanism to help cloud providers offer such guarantees.

Conversely, a vulnerability in the cloud service (for example, an incorrectly configured enclave-backed service) potentially affects everyone using that service. There is, therefore, a significant reliance on cloud providers to implement TEEs correctly (as with other aspects of their infrastructure)—where the discovery and consequences of a vulnerability can be more widespread, affecting swaths of tenants.



## Using TEEs, which include cryptographic means for isolating data and compute, does not remove one’s obligations under GDPR.



Ultimately, enclaves are but one measure that can help meet security obligations. By adopting technical measures such as TEEs—which can provide isolated execution, protected memory, remote attestation, and enabling other relevant functionality such as secure recordkeeping/logging (see “Enclaves: A Basis for Compliance and Accountability?”)—controllers and processors may be in a better position to meet their security obligations and mitigating or reducing their risk of penalty where breaches occur. Enclaves will provide only so much on their own, however; appropriate compliance regimes entail a holistic approach encompassing both technical and non-technical organizational measures.

*Encryption and personal data.* Encrypting personal data does not necessarily mean it is no longer personal data.<sup>7</sup> Again, this is because the data will still relate to an identifiable individual and can be decrypted, either with the appropriate key or because of some vulnerability in the encryption mechanism—both of which could be discovered or compromised at a later date. Moreover, both the act of encrypting and subsequent operations performed with that encrypted data (including storage, retrieval, transfer, decryption, or use) will also be considered processing and subject to GDPR.

In other words, although encryption can mitigate some risks regarding the disclosure of personal data, it often does not affect the status of that data as personal (GDPR Recital 28). Organizations will therefore still have the same obligations for appropriately managing encrypted personal data as they do for personal data in general.

In this way, using TEEs, which include cryptographic means for isolating data and compute, does not remove one’s obligations under GDPR. This raises implications for those processing data on behalf of others, such as cloud providers. Those providing such services still have responsibilities under GDPR for encrypted personal data, even if they do not have access to the legible (plaintext) data or decryption keys, and importantly, even if they are unaware that the data is personal in the first place.<sup>7,17</sup>

*Analytics and machine learning.* Data analytics and machine learning are areas of considerable commercial inter-

est. Note that despite the discussions concerning privacy-preserving analytics, the position under GDPR remains the same for analytics and ML as it does for any other form of processing. That is, analytics and machine learning are just particular forms of the wider category of operations performed on personal data that are collectively termed *processing*, and therefore it has the same obligations.

The cloud is described as having a key role to play in providing the infrastructure for enabling data storage, analytics, model building, among others. Though using the cloud will not relieve an organization's data protection obligations, there are responsibility-related incentives for prospective tenants leveraging enclave-backed processing environments for analytics and ML. As noted, controllers are obliged to use only those providers that can provide sufficient guarantees that their undertakings are in accordance with GDPR requirements—where, again, TEEs might assist providers in demonstrating their compliance. Further, using enclaves for cloud-based analytics/ML can provide tenants with additional confidentiality assurances, whereby the data and compute can be “sealed” from other parties and potentially the provider itself. This becomes particularly relevant where the data, computation, or results may be personal or otherwise sensitive. Moreover, integrity guarantees (over the code, data, and execution) can also be used to provide evidence demonstrating that the data was used only for particular purposes.

*Data sharing and transfer.* Many data processing scenarios involve the transfer of data, whether as part of its collection, aggregation, distribution, computation, or through interactions with various online services. Data may flow across technical boundaries (software, service, or hardware) or administrative domains (that is, infrastructures of different organizations).<sup>24</sup> In a cloud context, this can include data moving in-cloud (for example, across VMs, containers, and other hosted services), as well as data moving to/from cloud services through the tenants up/downloading data, and interactions with remote services.

The remote-attestation functionality of enclaves can be relevant here to help

ensure the confidentiality and integrity of code, data, and computation. This is because remote attestation allows an interrogation of the machine to which data is being considered for transfer by verifying, for example, that it is correctly configured, runs the correct code or applications, is operated by the expected organization, and so on. This can provide some assurance that data will be processed in an appropriate and expected manner. In this way, the technology can allow for more measured and controlled data transfers between parties.

From a data protection perspective, remote attestation can give controllers a greater degree of control over the processing operations for which they are ultimately responsible. Processors (typically providers) can process data only on behalf of and on the instruction of controllers (typically tenants) (GDPR Art.28 and 29), while also employing measures to ensure they process data in a manner that complies with GDPR (GDPR Art.28). Remote attestation potentially assists with this, paving a way for controllers to establish processing boundaries and to verify processor adherence. Attestation may also aid processors in demonstrating their own compliance and assisting their audit (GDPR Art.28), for example, where a provider involves others as part of its supply chain.

As such, TEEs represent one possible measure (of many) that could help controllers ensure that processors are operating appropriately. They also assist processors in demonstrating that they are meeting their obligations.

**Benefits and risk.** Cloud services are used by tenants for a wide variety of purposes. Given the growing awareness of legal obligations relating to data and its processing, various materials describe TEE technology as a means for tenants to meet their GDPR obligations (as mentioned earlier). Yet, while industry focuses on the benefits for tenants, less discussed is how the technology may actually serve the providers' interests.

*Cloud provider liabilities.* Under GDPR, the controller is the entity that remains ultimately responsible for compliance (GDPR Art.5(2)). In the cloud this is generally (but not always) the tenant. GDPR provides several ways, however, in which processors

(usually cloud providers) may be liable for certain violations, including in relation to security obligations, and therefore may face penalties for noncompliance. It follows that providers can benefit through means helping them manage these concerns.

Again, GDPR requires processors to take appropriate technical and organizational security measures, and to assist the controller in ensuring compliance with those obligations (GDPR Art.28(3)(c), Art.28(3)(f)). GDPR also requires that controllers and processors establish contractual provisions that oblige processors to make available to controllers the information for demonstrating compliance and facilitating audits (GDPR Art.28(3)(h)). Indeed, prominent cloud providers incorporate such terms into their standard form services agreements (see, for example, those for Amazon AWS<sup>2</sup> and Microsoft Azure<sup>14</sup>), although these raise practical considerations (for example, the limitations on the ability of tenants to meaningfully “inspect” a provider's operations<sup>17</sup>). This means that while both tenants and providers may be accountable to regulators and other oversight bodies for failures to meet these security requirements (which can result in potentially severe penalties), cloud providers should, according to GDPR, also be accountable in contract to the tenant and may face action, potentially including for damages, when they do not meet their obligations.

In other words, cloud providers have a number of obligations, and this exposes them to legal and financial liability. Cloud providers tend to be the major player in a technology supply chain, and as large organizations with substantial resources at their disposal with which to cover damages, they are often a prime target for litigation. Therefore, there are strong incentives for providers to limit and manage their risk exposure.

*Benefits for providers.* In adopting TEEs, cloud service providers may help mitigate their own liability exposure.

First, the security properties offered by enclaves can provide additional guarantees regarding the confidentiality of data, code, and processing. If there is a (plaintext) data leak, the provider may point to the use of enclaves to show that the data was protected on its platform, indicating that the responsibility for the

leak lies elsewhere. The integrity guarantees of enclaves can also help demonstrate that the cloud provider has acted appropriately (for example, in terms of system configuration, that the data was not tampered with, the right code was executed, and so on). It follows that in practice, enclaves may assist a provider by presenting evidence that it wasn't at fault. This can help in absolving liability when issues arise.

Although the provider, as a processor, still has obligations under GDPR, even where it does not know that data is personal and even if it cannot access the plaintext, the fact that the tenant requested an enclaved-backed service gives the provider an extra signal that the particular scenario may require special care. This can help inform the provider's associated risk-mitigation strategy.

*Benefits for tenants.* There are some benefits for tenants. First, the technology can help tenants manage their own legal obligations. As discussed, controllers can use only those processors that can provide sufficient guarantees they implement the appropriate technical and organizational measures to meet GDPR's requirements, including the security obligations (GDPR Art.28(1)). Tenants using the services of a cloud provider that employs TEEs can go some distance toward meeting their own obligations under GDPR and mitigate their own liability exposure. This is where the technology gives extra assurances that better justify the use of the cloud, while allowing tenants to benefit from the provider's functionality and expertise, where the provider, indeed, may well be better placed to deal with security and data-management concerns.

Enclaves can also provide a foundation for tenants to "audit" the provider. For example, if a cloud provider is claiming to use TEEs, the tenant should be able to verify this, as well as the associated operations and functionality that the enclave underpins. Where this is not occurring, the tenant can potentially take action against the provider.

More generally, however, enclaves, as a means for limiting a provider's involvement in processing operations (that is, by segregating data and compute), may actually encourage the uptake of the cloud in situations that otherwise would have been too risky

(for example, where data is particularly sensitive). Therefore, the technology can make cloud services more viable for prospective tenants by influencing the risk calculus such that compliance becomes less of a barrier for cloud adoption. This may be desirable for prospective customers, which may seek to migrate to the cloud to benefit from reduced operational costs (compared with running operations in-house) and to leverage the provider's other security measures, resources, and expertise.

*A gain for providers.* In short, TEEs can be useful for mitigating the provider's own risk. At the same time, the technology can also encourage more cloud adoption, which again benefits the cloud provider. This is interesting, given that providers are actively marketing the technology to prospective tenants, often at a premium, despite the significant benefits for providers in terms of their own risk management and in driving further business.

#### **The power of dynamics of adoption.**

A vast array of applications, services, and organizations rely on the use of cloud services. At the same time there is much consolidation among vendors in the cloud computing space, and in the tech industry more generally, in which there are a few dominant players.<sup>6</sup>

TEEs have the potential to help entrench the reliance on the cloud and strengthen the position of the major cloud service providers. This is because the guarantees provided by enclaves could drive further cloud adoption, including in situations where this would previously have been untenable. Moreover, if TEEs become widely accepted as best practice, it will be these providers that are best placed to meet the requisite standards. This is because of their significant resources, access to security expertise, experience with the technology, and ability to absorb the cost of implementing new technologies.

In this way, regulatory and market incentives may interact with the development and adoption of TEEs to help drive the consolidation of the technical infrastructure, which supports a range of applications, around a few dominant companies. This raises a number of concerns.<sup>6</sup>

First are those around security and resilience. An issue or vulnerability in a

cloud provider's service, such as an incorrectly configured TEE, can potentially impact their entire customer base. This is particularly problematic in a consolidated environment, given that the cloud aims at scale—a provider serves a vast range of customers, covering a vast range of applications (of varying importance for individuals, business, and society), meaning any issue can have systemic implications.

Further, by controlling technical infrastructure, providers have the power to act as gatekeepers, determining how, when, and why the cloud services that now underpin many applications are used. Since the major cloud providers offer a substantial collection of other products and services, some of those applications and services offered by tenants will be in competition with the offerings from the companies also providing the cloud. The power to determine who can use their services, and for what purposes, means they might—intentionally or otherwise—influence the behavior of their competitors in other markets, while at the same time generally influencing the broader application landscape.

The context for this discussion is that the dominant tech firms are in position to derive significant market and societal power.<sup>30</sup> As societal reliance on cloud services increases, so too does the power conferred on providers. As these providers were already dominant in their markets, this would, in effect, be a circular, self-reinforcing phenomenon: TEEs provide incentives for greater use of the cloud; TEEs become more widely accepted as best practice; cloud providers are best placed to implement TEEs, given their expertise and economies of scale; which provides further incentives for the use of cloud services; and so on. As a result, the current structural conditions of the cloud ecosystem, and the power of providers within that ecosystem and in society more generally, would likely be reinforced and entrenched.

Therefore, the potential for TEEs to increase the power of cloud providers and provide incentives to use the cloud should be factored into discussions concerning the dominance of technology firms—particularly in light of society's increasing reliance on technical infrastructure.<sup>30</sup>

*Enclaves: A basis for compliance and accountability?* A related area warranting consideration is how enclaves might provide a technical foundation for enabling new measures that aim explicitly at increasing levels of assurance, compliance, and accountability<sup>24</sup>—that is, to explore whether and how TEEs can provide a suitable processing environment and other guarantees upon which specific legal compliance and accountability mechanisms could be built. For example, TEEs could provide the basis for raising levels of assurance in systems audit data (evidence) through improved recordkeeping regimes (for example, by improving the security and integrity of logs and logging mechanisms<sup>10</sup>). Generally, there is a need for further work on enabling more reliable and robust mechanisms supporting the oversight, monitoring, management, scrutiny, and review of technical infrastructure and organizational practices. This is an area ripe for attention.

### Concluding Remarks

Organizations that deal in data are increasingly subject to legal and regulatory obligations. In exploring how data protection relates to enclaves in a cloud context, we have shown that—despite material suggesting the possibility—TEEs will not *solve* compliance issues. They do not provide a means by which those designing, deploying, or using technical systems can avoid their data protection responsibilities. Instead, TEEs represent yet another security tool (but one tool among many) that can aid risk management. They have the potential to help in building more secure systems, and in doing so can perhaps assist with meeting *some* legal obligations.

For tenants, the real gains appear less about the legal aspect and more about reducing the compliance barriers for cloud adoption—where cloud uptake can result in cost savings. While enclaves are marketed to prospective tenants, many of the potential benefits of TEEs actually fall to the providers. That is, providers can use the technology to manage their own risks better and to drive further business. Indeed, this accords with the prior criticisms of such technology as primarily being in the interests of vendors. As outlined here, lower-level security technologies such as TEEs can

reinforce the power of the few dominant tech companies and, therefore, warrant further consideration.

There also appears to be an opportunity for the technology to support stronger governance regimes by underpinning compliance, reporting, and other accountability mechanisms. This is an area that is so far underexplored.

Overall, the above tech-legal analysis shows that there is more to enclaves than just their functionality. As such, the technology should be on the radar of not only technologists, but also businesses, regulators, policymakers, and civil society.

### Acknowledgments

The Compliant and Accountable Systems Group acknowledges the financial support of the Engineering and Physical Sciences Research Council, University of Cambridge, through the Trust & Technology Initiative, and Microsoft through the Microsoft Cloud Computing Research Centre. Do Le Quoc is supported by funding from an innovation program under the LE-GaTO Project (legato-project.eu), grant agreement no. 780681. **C**

### References

1. Alder, F., Asokan, N., Kurnikov, A., Paverd, A., Steiner, M. S-FaaS: Trustworthy and accountable function-as-a-service using Intel SGX. In *Proceedings of the ACM SIGSAC Conf. Cloud Computing Security*, 2018, 185–199. <https://dl.acm.org/doi/10.1145/3338466.3358916>.
2. Amazon. AWS GDPR Data Processing Addendum; [https://d1.awsstatic.com/legal/aws-gdpr/AWS\\_GDPR\\_DPA.pdf](https://d1.awsstatic.com/legal/aws-gdpr/AWS_GDPR_DPA.pdf).
3. Anati, I., Gueron, S., Johnson, S., Scarlata, V. Innovative technology for CPU-based attestation and sealing. In *Proceedings of the 2nd Intern. Workshop on Hardware and Architectural Support for Security and Privacy*. ACM.
4. Anderson, R. Cryptography and competition policy: issues with “trusted computing.” In *Proceedings of the 22nd Annual Symp. Principles of Distributed Computing*, 2003, 3–10.
5. Arnaudov, S. et al. SCONe: Secure Linux Containers with Intel SGX. In *Proceedings of the 12th Usenix Symp. Operating Systems Design and Implementation*, 2016, 689–703; <https://dl.acm.org/doi/10.5555/3026877.3026930>.
6. Cobbe, J., Norval, C., Singh, J. What lies beneath: Transparency in online service supply chains. *J. Cyber Policy* 5, 1 (2020), 65–93.
7. European Commission. What is personal data? [https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data\\_en](https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en).
8. Gollamudi, A., Chong, S. Automatic enforcement of expressive security policies using enclaves. In *Proceedings of the ACM SIGPLAN Intern. Conf. Object-oriented Programming, Systems, Languages and Applications*, 2016, 494–513; <https://dl.acm.org/doi/10.1145/2983990.2984002>.
9. Hunt, G., Letey, G., Nightingale, E. The seven properties of highly secure devices. Microsoft Technical Report MSR-TR-2017-16; <https://www.microsoft.com/en-us/research/publication/seven-properties-1st-edition/>.
10. Karande, V., Bauman, E., Lin, Z., Khan, L. SGX-Log: Securing system logs with SGX. In *Proceedings of the 2017 ACM Asia Conf. Computer and Communications Security*, 19–30.

11. Le Quoc, D., Gregor, F., Arnaudov, S., Kunkeland, R., Bhatotia, P., Fetzer, C. secureTF: A secure TensorFlow framework. In *Proceedings of the 21st Intern. ACM/IFIP Middleware Conference*, 2020.
12. Le Quoc, D., Gregor, F., Singh, J., Fetzer, C. SGX-PySpark: Secure distributed data analytics. In *Proceedings of WWW '19: The World Wide Web Conf.*; <https://dl.acm.org/doi/10.1145/3308558.3314129>.
13. Linux Foundation. Confidential Computing Consortium, 2020; <https://confidentialcomputing.io/>.
14. Microsoft. Online Services Data Protection Addendum; <https://www.microsoft.com/en-us/licensing/product-licensing/products>.
15. Microsoft. Virtualization-based security (VBS) memory enclaves: Data protection through isolation, 2018; <http://bit.ly/3ps8rF2>.
16. Microsoft. Azure confidential computing, 2020; <https://azure.microsoft.com/en-us/solutions/confidential-compute>.
17. Millard, C.J., ed. *Cloud Computing Law*, second edition. Oxford University Press, 2021.
18. Nilsson, A., Bideh, P.N., Brorsson, J. 2020. A survey of published attacks on Intel SGX; arXiv:2006.13598.
19. Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., Costa, M. Oblivious multi-party machine learning on trusted processors. In *Proceedings of the 26th Usenix Conf. Security Symposium*, 2016; <https://dl.acm.org/doi/10.5555/3241094.3241143>.
20. Open Enclave SDK. 2019; <https://github.com/openenclave/openenclave>.
21. Schoen, S.D. Trusted computing: promise and risk. Electronic Frontier Foundation, 2003; [https://www.eff.org/files/20031001\\_tc.pdf](https://www.eff.org/files/20031001_tc.pdf).
22. Schuster, F., Costa, M., Fournet, C., Gkantsidis, C., Peinado, M., Mainar-Ruiz, G., Russinovich, M. VC3: Trustworthy data analytics in the cloud using SGX. In *Proceedings of the 2015 IEEE Symp. Security and Privacy*; <https://ieeexplore.ieee.org/document/7163017>.
23. Schwarz, M., Weiser, S., Gruss, D. Practical enclave malware with Intel SGX. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer International Publishing, 2019, 177–196.
24. Singh, J., Millard, C., Reed, J., Cobbe, J., Crowcroft, J. Accountability in the IoT: Systems, law, and ways forward. *IEEE Computer* 51, 7 (2018), 54–65; <https://ieeexplore.ieee.org/document/8423131>.
25. Skillern, R. Intel architecture enables new IBM cloud service with enhanced container security. Intel IT Peer Network, 2018; <https://itpeernetwork.intel.com/intel-ibm-cloud-container-security>.
26. Tarkhani, Z., Madhavapeddy, A. Sirius: enabling system-wide isolation for trusted execution environments. 2020; arXiv:2009.01869.
27. Tsai, C.-C., Porter, D. E., Vij, M. Graphene-SGX: A practical library OS for unmodified applications on SGX. In *Proceedings of the 2017 Usenix Annual Technical Conf.*, 645–658; <https://dl.acm.org/doi/10.5555/3154690.3154752>.
28. U.S. Department of Defense. 1985. DoD Trusted Computer System Evaluation Criteria.
29. Van Bulck, J., Oswald, D., Marin, E., Aldoseri, A., Garcia, F.D., Piessens, F. A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes. In *Proceedings of the 2019 ACM SIGSAC Conf. Computer and Communications Security*, 1741–1758; <https://dl.acm.org/doi/10.1145/3319535.3363206>.
30. van Dijk, J., Poell, T., de Waal, M. *The Platform Society: Public Values in a Connective World*. Oxford University Press, 2018.

**Jatinder Singh** heads the Compliant and Accountable Systems Research Group in the Department of Computer Science and Technology, the University of Cambridge, U.K.

**Jennifer Cobbe** is a research associate in the Compliant and Accountable Systems Group, Department of Computer Science and Technology, the University of Cambridge, U.K.

**Do Le Quoc** is a postdoc at the Systems Engineering Group of TU Dresden, Germany, and a co-founder of Scontain UG.

**Zahra Tarkhani** is a Ph.D. candidate in the Systems Research Group at the Cambridge University Computer Laboratory, Cambridge, U.K.

Copyright held by authors/owners.

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

## A closer look at the technology that makes portable electronics possible.

BY JESSIE FRAZELLE

# Battery Day

**more online**  
A version of this article with embedded informational links is available at <https://queue.acm.org/detail.cfm?id=3439415>

TESLA HELD ITS first Battery Day on September 22, 2020. What a fantastic world we live in that we can witness the first Apple-like keynote for batteries. Batteries are a part of everyday life; without them, the world would be a much different place. Your cellphone, flashlight, tablet, laptops, drones, cars, and other devices would not be portable and operational without batteries.

At the heart of it, batteries store chemical energy and convert it into electrical energy. The chemical reaction in a battery involves the flow of electrons from one electrode to another. When a battery is discharging, electrons flow from the *anode*, or negative electrode, to the *cathode*, or positive electrode. This flow of electrons provides an electric current that can be used to power devices. Electrons have a negative charge;

therefore, as the flow of negative electrons moves from one electrode to another, an electrolyte is used to balance the charge by being the route for charge-balancing positive ions to flow.

Let's break this down a bit and uncover the chemical reactions at play within batteries. An electrical current requires a flow of electrons. Where do those electrons come from?

Electrons in the anode are produced by a chemical reaction between the anode and the electrolyte. Simultaneously, another chemical reaction occurs in the cathode, enabling it to accept electrons. These chemical reactions create the flow of electrons, resulting in an electric current.

A chemical reaction that involves the exchange of electrons is known as a *reduction-oxidation reaction*, or redox reaction.

*Reduction* refers to a gain of electrons. Thus, half of this reaction—the reduction—occurs at the cathode because it gains electrons. *Oxidation* refers to a loss of electrons. Therefore, the other half of this reaction—oxidation—occurs at the anode because it loses electrons to the cathode. Each of these reactions has a particular electric potential. An electrochemical cell can be made up of any two conducting materials that have reactions with different standard potentials, since the more robust material, which makes up the cathode, will gain electrons from the weaker material, which makes up the anode.

Batteries can be made up of one or more electrochemical cells, each cell consisting of one anode, one cathode, and an electrolyte, as described earlier. The electrodes and electrolyte are generally made up of different types of metals or other chemical compounds. Different materials for the electrodes and electrolyte produce different chemical reactions that affect how the battery works, how much energy it can store, and its voltage.

### Volts

The word *volt* refers to the measure of electric potential. The term derives

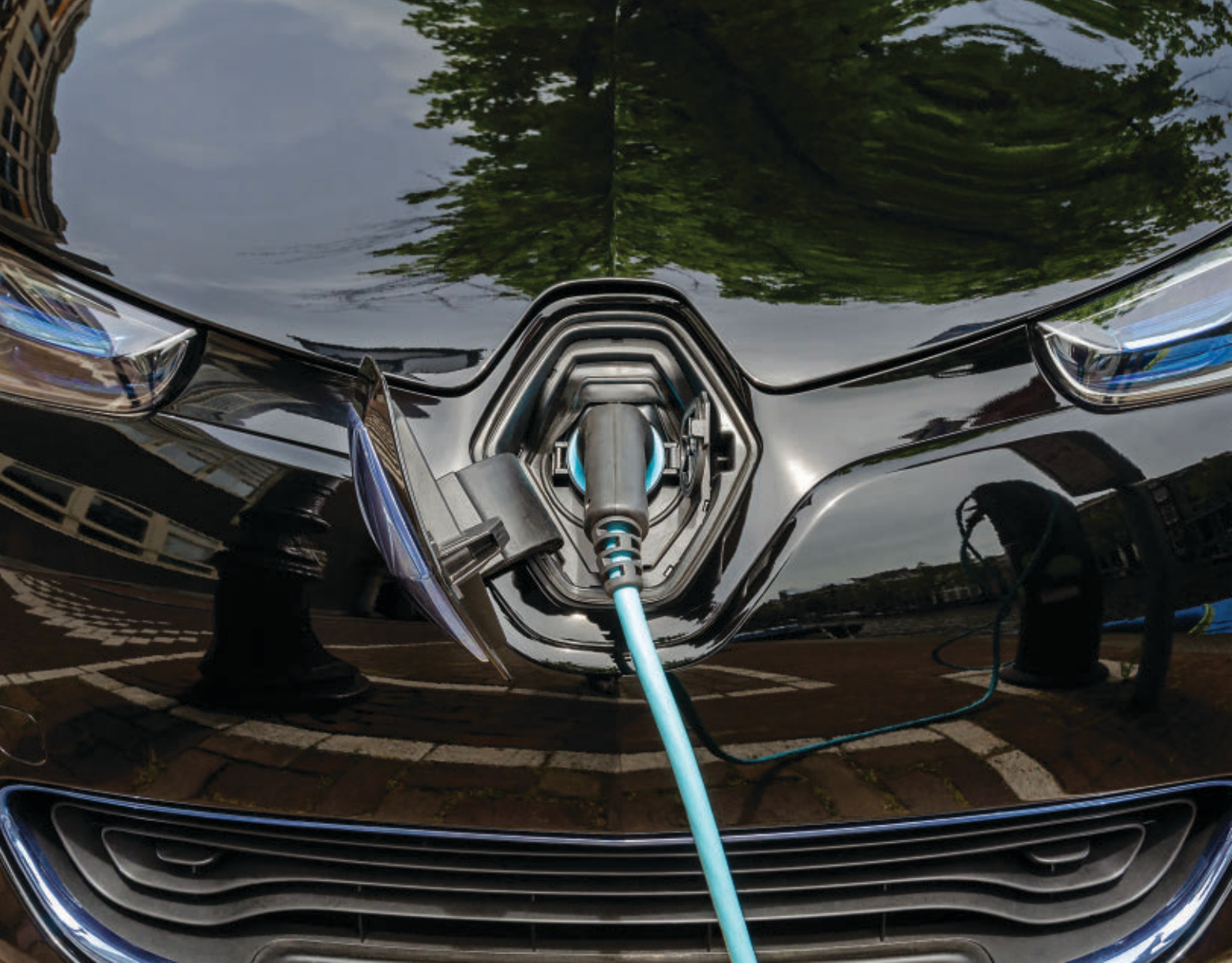


IMAGE BY ANDRII MALIKOV

from the name of the Italian scientist who is credited with inventing the first battery, Alessandro Volta. In 1780, Luigi Galvani, another Italian scientist, observed that the legs of frogs hanging on iron or brass hooks would twitch when touched with a probe of some other type of metal. Galvani believed that this was caused by electricity from within the frogs' tissues. He called it "animal electricity."

Volta believed the electric current came from the two different metal types: the hooks on which the frogs were hanging and the probe's different metal. He thought the current was merely being transmitted *through*, not *from*, the frogs' tissues. Volta experimented with stacks of silver and zinc layers interspersed with layers of cloth or paper soaked in saltwater and found an electric current flowed through a wire applied to both ends of the pile. Volta also found the amount of voltage could be

increased by using different metals in the pile, leading to what we know today as the scientific unit of a volt.

There are two ways to increase a battery's voltage: stack several cells together or increase a cell's electrochemical potential by choosing different materials.

Combining cells in a series has an additive effect on the battery's voltage. Essentially, the force at which the electrons move through the battery can be seen as the total force as they move from the first cell's anode through all the cells the battery contains to the last cell's cathode.

In contrast, combining cells in parallel increases the battery's possible current, which is defined as the total number of electrons flowing through the cells, but not its voltage.

### Measuring Electricity

When you buy a light bulb, the box indicates the wattage for the bulb. The

watt is a measurement of power. It describes the rate at which electricity is being used at a specific moment. Therefore, a 60-watt light bulb uses 60 watts of electricity at any moment while turned on.

A *watt-hour* (Wh), on the other hand, is a measurement of energy. It describes the total amount of electricity used over time. You can derive from its name that watt-hours are a combination of watts, the rate electricity is used; and hours, the length of time used. Going back to the example, a 60-watt light bulb that draws 60 watts of electricity at any moment while turned on, uses 60 watt-hours of electricity in one hour.

Watt-hours will get you only so far, however. To measure the electricity used by a large appliance or a household, the more common term is *kilowatt-hour* (kWh). A kilowatt is equal to 1,000 watts; therefore, one kilowatt-hour is equal to 1,000 watt-hours.

To measure the output of a power plant or the amount of electricity used by an entire city, *megawatt* is the more practical term. A megawatt is 1,000 kilowatts or 1 million watts. Going even larger, a *gigawatt* is 1,000 megawatts, 1 million kilowatts, or 1 billion watts. Gigawatt is the namesake of Tesla's Gigafactory. In 2018, battery production at the Gigafactory in Nevada reached 20 gigawatt-hours (GWh) per year.

### Alkaline Batteries

Most people are familiar with alkaline batteries. These are the batteries you typically use to power toys, electronics, flashlights, among others. The bulk of alkaline batteries produced are single-use, although there are some rechargeable alkaline batteries.


Alkaline batteries use zinc as the anode and manganese dioxide as the cathode. The name comes from the alkaline solution used as the electrolyte. The electrolyte is typically potassium hydroxide, which can contain a large number of dissolved ions. The more ions the electrolyte solution can absorb, the longer the redox reaction that drives the battery can keep going.

The zinc anode is usually in powdered form. Powder has a greater surface area for a reaction, which means the cell can quickly release its power. The zinc anode gives up its electrons to the manganese-dioxide cathode, to which carbon, in the form of graphite, is added to improve its conductivity and help it keep its shape.


Alkaline batteries are popular because they have a low self-discharge rate, giving them a long shelf life, and they don't contain toxic heavy metals such as lead or cadmium. They account for the bulk of batteries that are made today, although their place at the top will likely soon be challenged by the lithium-ion batteries in our phones, laptops, cars, and an increasing number of other gadgets.

### Lithium-ion Batteries

Lithium-ion batteries are popular because of their energy density. Because the energy is dense, your phone can last all day and still be the small, portable, handheld device we are all familiar with. As you likely know from the behavior of your phone, lithium-ion batteries are rechargeable. The name



**A lithium-ion battery starts its life in a state of full discharge: All its lithium ions are intercalated within a cathode, and its chemistry cannot yet produce any electricity.**



of the battery comes from the fact that lithium ions are involved in the chemical reactions that make up the battery.

In a lithium-ion cell, both electrodes— anode and cathode—are made of materials that can absorb lithium ions. The absorbing action is known as *intercalation*, which allows the charged ions of an element to be stored inside a material without significantly disturbing it. The lithium ions are paired with an electron within the structure of the anode. When the battery discharges, the intercalated lithium ions are released from the anode and travel through the electrolyte solution to be intercalated in the cathode.

A lithium-ion battery starts its life in a state of full discharge: All its lithium ions are intercalated within the cathode, and its chemistry cannot yet produce any electricity. Before the battery can be used, it needs to be charged. As the battery is charged, an oxidation reaction occurs at the cathode, meaning that it loses some negatively charged electrons. An equal number of positively charged intercalated lithium ions are dissolved into the electrolyte solution to maintain the charge balance in the cathode. These travel over to the anode, where they are intercalated, or absorbed, into what is typically graphite. This intercalation reaction also deposits electrons into the graphite anode, to pair with the lithium ions.

There are many other types of batteries, but for the context of this article you primarily need to understand lithium-ion batteries. (Some people might, however, be interested in a urine-powered battery.)

### New Technologies

**Solid-state batteries.** Instead of the liquid or polymer-gel electrolyte found in batteries today, a solid-state battery uses a solid electrolyte and solid electrodes. Recall that positive ions flow through the electrolyte to balance the electrons' negative charge. Today, batteries are quite efficient at transferring positive ions since a liquid electrolyte is in contact with the electrodes' entire surface area. Using a solid makes this a bit harder. Imagine the difference between dipping a chip in soup and dipping it into chopped tomatoes. The soup would cover more



of the chip's surface area than would the chopped tomatoes.

Why use a solid electrolyte if it is less efficient? Today's lithium-ion batteries typically rely on flammable liquids as the electrolyte. By using a solid electrolyte, batteries can be less prone to bursting into flame. You may remember Samsung's Galaxy Note 7, which had the unfortunate side effect of catching fire. Solid electrolytes provide a much safer alternative.

Solid electrolytes typically tend to be either solid polymers at high temperatures or ceramics at room temperature. The downside of solid polymers is they need to operate at temperatures above 220°F (105°C). That is certainly not practical for a handheld device such as a phone or tablet but could work for storing energy to power a home.

A few companies are working on using ceramics at room temperature to create a solid-state battery. Toyota has been talking about its battery for years and aims to have it completed in 2025. Startups such as Solid Power and A123 Systems (with the help of Ionic Materials) aim to do the same.

A lot of the novel research being done on solid-state batteries is the work of Jürgen Janek, who recently published a benchmark of the performance of all-solid-state lithium batteries. Another

high-profile battery scientist, Gerbrand Ceder, published a paper on interface stability in solid-state batteries. New research on solid-state batteries is being published quite frequently. Although there are many skeptics of solid-state batteries, since they have yet to be commercially delivered and scaled, I would not dismiss it from having a seat at the table in the future.

**Nuclear batteries.** So far we have discussed only batteries powered by chemical reactions, such as those in flashlights, phones, and other gadgets. Chemical batteries, also known as galvanic cells, discharge in a given amount of time and need to be either thrown away or recharged. Is there a type of battery that could last long term?

Nuclear batteries, or atomic batteries, use the energy of beta decay and are known as *betavoltaics*. They are receiving research attention because they could last much longer than those powered by chemical reactions. Radioactive isotopes used in nuclear batteries have half-lives ranging from tens to hundreds of years, so their power output remains nearly constant for a very long time.

If nuclear batteries can last from tens to hundreds of years, why are they not being used everywhere today? Doesn't everyone want a phone that

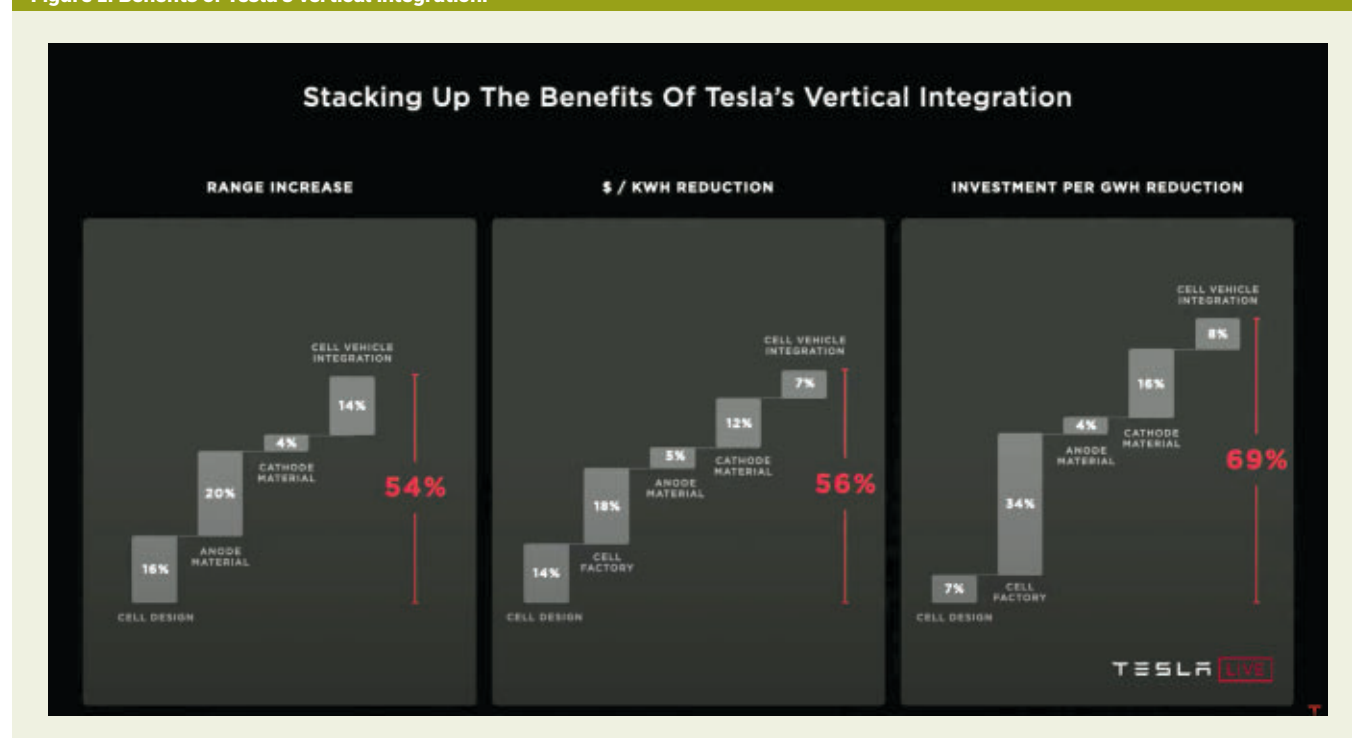
could last at least a decade without needing to be charged?

Nuclear batteries have a few side effects. They cannot be turned off; electrons are continually being produced, even when they are not needed. Research is being done into stimulating beta decay, which would create more current on demand, allowing the output to drop to almost nothing when it is turned off.

Another downside is the power density of betavoltaic cells is much lower than that of chemical batteries. It is interesting to note that betavoltaics were used in the 1970s to power cardiac pacemakers, before being replaced by cheaper lithium-ion batteries, even though the lithium-ion battery has a shorter lifetime.

In 2016, Russian researchers from the National University of Science and Technology MISiS presented a prototype betavoltaic battery based on nickel-63. A downside of using nickel-63 is that it is not readily available, making their research hard to commercialize. City Labs sells a betavoltaic battery with a 14.4-year half-life, which you can buy today starting at \$1,000, but you would need 1.2 million of these just to have one watt of power. NDB is a startup working on a nano diamond battery that could last for thousands of years.

Figure 1. Benefits of Tesla's vertical integration.



UPower Technologies is another start-up working on a megawatt-scale atomic generator.

**Silicon anode.** Today the material typically used for the anode is graphite because it is economical, reliable, and relatively energy dense, especially compared with current cathode materials. The limiting factor of lithium-ion batteries is the amount of lithium that can be stored in the electrodes. Using silicon as the material for the anode, rather than graphite, allows around nine times more lithium ions to be held in the anode.

The ability to store more lithium ions using silicon sounds amazing; why isn't everyone doing this? The problem is a silicon anode swells to three to four times its original volume when it absorbs lithium ions. Making the casing bigger doesn't circumvent the problem because the expansion causes the silicon to fracture, leading to failure of the battery. A passivation layer, also known as the SEI (solid electrolyte interphase), forms on electrode surfaces from the decomposition of electrolytes. This passivation layer typically inhibits further electrolyte decomposition, giving the battery a longer life. The absorption of lithium ions gums up this layer making it less effective.

*"With silicon, the cookie crumbles and gets gooey."*  
—Elon Musk

As a solution to this problem, many companies use silicon as a fraction of the anode material, but such materials are expensive and highly engineered. Examples include silicon structured in SiO (silicon dioxide) glass (\$6.60 per kWh), silicon structured in graphite (\$10.20 per kWh), and silicon nanowires (>\$100 per kWh). Sila Nanotechnologies is using silicon as its anode material. Amprius claims to use silicon for 100% of the anode material with silicon nanowires, a highly engineered, expensive material. Advano, Enevate, and Enovix are startups working on a silicon solution for the anode material.

### Tesla's Battery Day

At Tesla's Battery Day event, the company announced many changes to its battery that encompass more than just the materials used. Tesla has on staff the renowned battery scientist Jeff Dahn. His most recent papers, "A Wide Range of Testing Results on an Excellent Lithium-Ion Cell Chemistry to be Used as Benchmarks for New Battery Technologies" and "Is Cobalt Needed in Ni-rich Positive Electrode Materials for Lithium-Ion Batteries?" help give some insight into what Tesla has been working on.

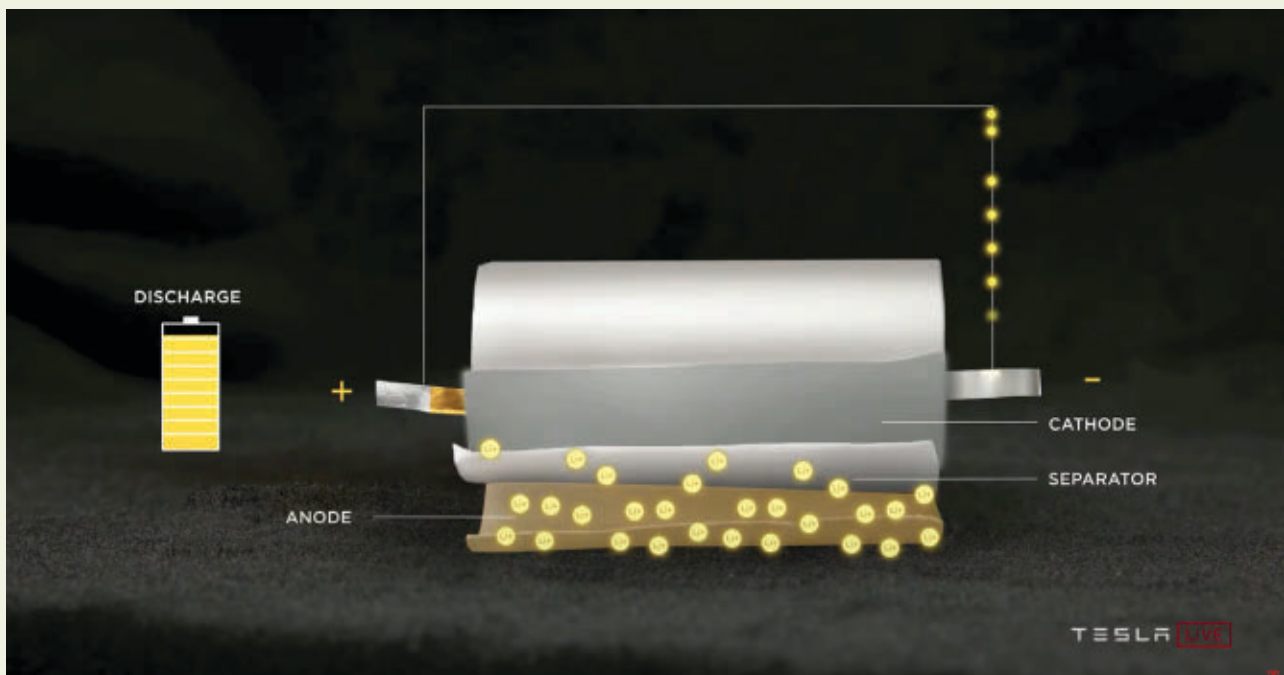
The Battery Day outcomes could increase Tesla vehicles' range while being more economical; the plan is to halve the cost per kilowatt-hour. Most startups (except for Sila Nanotechnologies, which seems to be most closely aligned with Tesla's methodology) in this space tend to take a single design decision into account for their products—for example, anode material—and focus on that. Tesla, on the other hand, took a broader approach. It took into account not only the materials for the cathode and anode, but also the cell design, factory, and integration with the vehicle, illustrated in Figure 1. (Tesla claimed in its presentation there were more aspects not mentioned that it could improve in the future.)

Let's break down each of these improvements.

**Cell design.** For Tesla's batteries, when discharging, the positive ions flow over the tabs, while the lithium ions flow from the anode to the cathode, as shown in Figure 2. The tabs allow a cell's energy to be transferred to an external source.

The Tesla team sought to increase the cell size to 46 millimeters, which optimizes vehicle range and cost reduction. Increasing the cell's size, however, has a negative side effect on supercharging because of thermal issues.

Figure 2. Battery discharge.



To circumvent these issues, the Tesla team removed the tabs, calling the new design “tabless.”

The tabless design leads to simpler manufacturing, fewer parts, and a five-fold reduction of the electrical path. Going from a 250-millimeter to a 50-millimeter electrical path length leads to substantial thermal benefits. The length of the electrical path is significant because the distance the electron has to travel is much less. Even though the cell is much bigger, the power-to-weight ratio is better than a smaller cell with tabs.

Why does this new tabless design matter? Instead of calling it tabless, Tesla could have called it *many tabs*, because each of the folded pins is a tab, as shown in Figure 3. What is the function of a tab?

Consider this analogy: When I was growing up, my family would always leave sporting events before they ended to avoid the crowd trying to exit the stadium. If we had stayed to the end of the event, the exiting process would have taken more time and been uncomfortable since everyone would be trying to leave through a few exits at the same time. They get closer and closer to one another, and the environment becomes hot and rowdy. If you think of the exiting people as electrons, a stadi-

um with a single exit is similar to a battery’s behavior with a single tab; electrons are all trying to leave through the single tab and bumping up against one another until they heat up.

Tesla’s new design has multiple tabs, equivalent to a stadium with lots of exits. Now people, or electrons, can exit quickly while staying cool and calm.

Tesla didn’t provide many details in its presentation on the new tabless design and its implementation, but it can be attributed to “secret sauce.”

Manufacturing a cell consists of an electrode process where the active materials are coated into films onto foils; the coated foils are then wound in the winding process. The roll is assembled into the can, sealed, and filled with electrolyte and then sent to formation, where the cell is charged for the first time. Recall that a lithium-ion battery starts its life in a discharged state. For a battery cell with tabs, manufacturing is much more complicated. When the cell with tabs is going through the assembly line, it has to keep stopping where all the tabs are so you can’t do continuous motion production. It is also a lot more error prone.

*“It is really a huge pain ... to have tabs from a production standpoint.”*

—Elon Musk

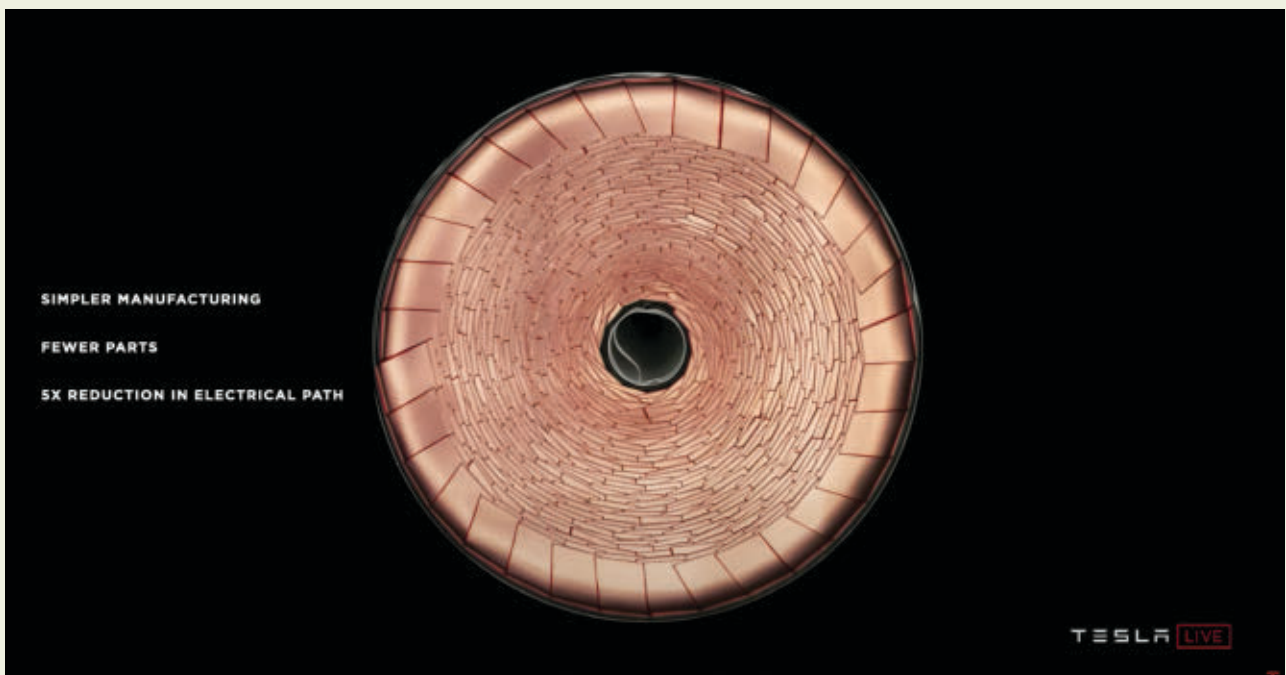
The new batteries are 46 millimeters by 80 millimeters, leading to the name 4680, referring to the diameter and length. Previously, an extra zero was added onto the end of the name but was removed since it had no purpose.

The 4680 batteries have five times more energy with six times the power and enable a 16% range increase. At the battery-pack level, the form-factor improvements alone result in a 14% reduction in cost per kWh.

**Cell factory.** Earlier we learned a bit about how removing tabs from the battery cells simplifies the manufacturing process. In an assembly line, you don’t want the line to stop and start, but continuously move. Stopping the process leads to inefficiency. The Tesla team aims to speed up its process to make one factory be multiple scales of efficiency better than a typical battery factory.

We also learned the electrode process is where the active materials are coated into films onto foils. The wet step of the electrode process begins with mixing, which occurs when the powders are mixed with either water or a solvent, typically a solvent for the cathode. The mix then goes into a large coat-and-dry oven, tens of meters long, where the slurry is coated onto the foil and dried. The solvent

Figure 3. Tabless cell.



**more online**

A version of this article with embedded informational links is available at <https://queue.acm.org/detail.cfm?id=3439415>

then has to be recovered. Finally, the coated foil is compressed to the final density. This process is complex and inefficient, especially since humans need to transport the mix from the mixing step to the ovens. It is also inefficient because of the need to put the solvent in and then recover it.

One significant change Tesla is making is skipping the solvent step of the electrode coating's wet process in favor of a dry process, which transforms the powder directly into film. This technology stemmed from Tesla's acquisition of Maxwell at the beginning of 2019. On Battery Day, Musk mentioned that since the acquisition, Tesla is now on the fourth revision of the equipment that turns powder into film. He noted, "There is still a lot of work to do. There is a clear path to success but a ton of work between here and there." When this process is scaled up, it results in a tenfold reduction in footprint and an equal reduction in energy, as well as a massive decrease in capital expenditure (CapEx).

The manufacturing step known as formation is where the cell is charged for the first time and its quality is verified. Formation is typically 25% of the

CapEx investment. The Tesla team improved density and cost effectiveness by using their knowledge from cars and power-wall charging and discharging. This led to an 86% reduction in CapEx investment per GWh in formation and a 75% reduction in footprint. For a factory that previously had an output of 150GWh, this translates to that same factory putting out 1TWh using the more efficient processes. At the battery-pack level, this led to an 18% reduction in cost per kWh.

**Anode material.** Tesla announced it was moving to silicon as its anode material. Silicon is the most abundant element in the earth's crust after oxygen. Rather than creating a highly engineered material that would be expensive, Tesla will use raw silicon and design for it to expand. The silicon's surface will be stabilized through an elastic, ion-conducting polymer coating and a highly elastic binder and electrolyte.

Tesla's silicon costs \$1.20 per kWh, whereas the other solutions covered here cost anywhere from \$6 per kWh to upward of \$100. Using silicon will lead to a 5% reduction in cost per kWh at the battery-pack level and a 20% longer range for Tesla vehicles.

**Cathode material.** A helpful analogy for understanding the cathode is to

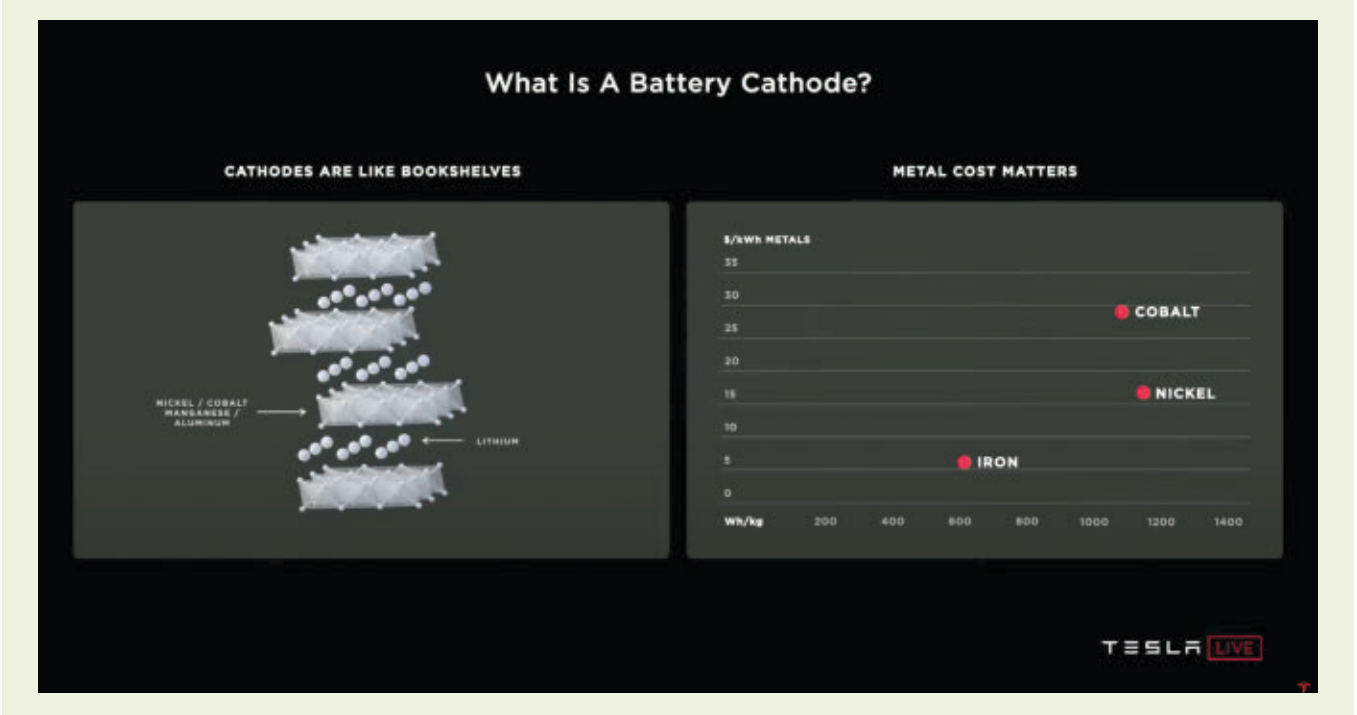
think of it as a bookshelf, as illustrated in Figure 4. In this case, the lithium ions are books. The most efficient bookshelf holds the most books while still being stable enough to retain its structure as the books are taken out and returned.

The Tesla team aims to increase the amount of nickel in its cathode material since it is less expensive and has the highest energy density (as shown previously). Cobalt is typically used as a cathode material because it is very stable. The Tesla team, however, aims to leverage novel coatings and dopants to stabilize nickel better and remove cobalt entirely from its materials. Removing cobalt will lead to a 15% reduction in the cathode's cost per kWh.

The cost and availability of materials were important considerations. Availability was not an issue for the anode material since silicon is abundant. The same goes for lithium, which is also highly accessible. For nickel, on the other hand, the Tesla team is diversifying the amount it is using for each type of vehicle.

The team also simplified the cathode-manufacturing process by removing all the legacy parts. According to the Battery Day presentation, the cathode manufacturing process, which is

Figure 4. Battery cathode.



35% of the cathode cost per kWh, had not had a fresh look in a long time and was wildly inefficient.


*“If you take a look at the ‘It’s a small world journey’ of I am a nickel atom, and what happens to me, it’s crazy, you’re going around the world three times, there is a moral equivalent of digging the ditch, filling in the ditch, and digging the ditch again. It’s total madness.”* —Elon Musk

A typical cathode process starts with the metal from the mine being turned into an intermediate material called metal sulfate, which, in turn, is processed again. The Tesla team removed the intermediate step of turning the metal into metal sulfate along with a bunch of other unnecessary steps. It also localized the cathode materials to the U.S., which decreased the number of miles required for the materials to travel. This led to a 66% reduction in CapEx investment, a 76% reduction in process cost, and zero wastewater. The cathode material improvements led to a 12% reduction in cost per kWh at the battery-pack level.


**Cell vehicle integration.** In the early days of aircraft, fuel was carried as cargo. Later, fuel tanks were made in the shape of the wings. This was a breakthrough because the wings, which are critical to the airplane’s function, now could be used for another purpose. The fuel tank was no longer cargo but fundamental to the structure of the aircraft. Tesla intends to do the same for cars.

Removing the intermediate structure in the battery pack allows the cells to be packed more densely. Instead of having supports and stabilizers in the battery cells, making up the intermediate structural elements, the battery pack itself is structural. Tesla had been filling the battery packs with a flame retardant. The new battery packs now are filled with a flame retardant and structural adhesive, giving them stiffness and stability without intermediate structural elements. This makes the structure even stiffer than a regular car.

The cells can now be moved more toward the center of the vehicle because the volumetric efficiency is better, avoiding the risk of a side impact potentially contacting the cells. This



**Producing more affordable electric vehicles broadens Tesla’s market to include new buyers, thereby reducing the number of gas-powered vehicles on the road.**




also allows the car to maneuver better because the polar moment of inertia is improved—much the way an ice skater can turn better with arms close to the body rather than extended out.

The improvements to the battery-pack integration led to a 10% mass reduction in the car’s body, a 14% range increase, and 370 fewer parts. The smaller, integrated battery and body also help increase the efficiency of manufacturing. This has resulted in a 55% reduction in CapEx investment and a 35% reduction in floor space. At the battery-pack level, the integration improvements mean a 7% reduction in cost per kWh.

The sum of all these improvements—including cell design, factory, materials, and vehicle integration—achieves the goal of halving the cost per kWh. Producing more affordable electric vehicles broadens Tesla’s market to include new buyers, thereby reducing the number of gas-powered vehicles on the road.

### Summary

It is satisfying to see a technology we all rely on day to day get its time in the spotlight. Although not mentioned during Battery Day, if Tesla were to achieve 400 watt-hours per kilogram, a zero-emissions jet just might be on the horizon. Now that batteries are vertically integrated into Tesla’s product, you can only imagine that the software will track more data on battery efficiency, leading to more improvements in the future.

It is encouraging to see Tesla take a fresh look at making more efficient and cost-effective batteries. The level of thought and detail put into rethinking old processes to make them more efficient is inspiring. The Tesla team didn’t just look at one angle but all the angles: cell design, manufacturing, vehicle integration, and materials. There is a clear *why* for every decision made that boils down to economics, not just technical gains. Hopefully, we will see another core technology in the spotlight soon. 

**Jessie Frazelle** is the cofounder and chief product officer of the Oxide Computer Company. Before that, she worked on various parts of Linux, including containers, as well as the Go programming language.

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

DOI:10.1145/3418296

## How to avoid insider cyber-attacks by creating a corporate culture that infuses trust.

BY ERIC GROSSE, FRED B. SCHNEIDER, AND LYNETTE L. MILLETT

# Implementing Insider Defenses

CLASSICAL APPROACHES TO cyber-security— isolation, monitoring, and the like—are a good starting point for defending against attacks, regardless of perpetrator. But implementations of those approaches in hardware and/or software can invariably be circumvented by *insiders*, individuals who abuse privileges and access their trusted status affords. An organizational culture in which people and procedures are part of the system's defenses is thus necessary. Such a culture would instantiate classical approaches to cyber-security but implemented by people who follow administrative procedures. So, a careful look at a system's defenses finds that many of the same classical approaches reappear at each level. But the implementation at the lowest layers— structures we might term *insider defenses*— involves people.

People do not slavishly follow administrative procedures the way a computing system executes its programs. In addition, people are more prone than computing systems to making errors, and people can be distracted or fooled. Finally, because they can

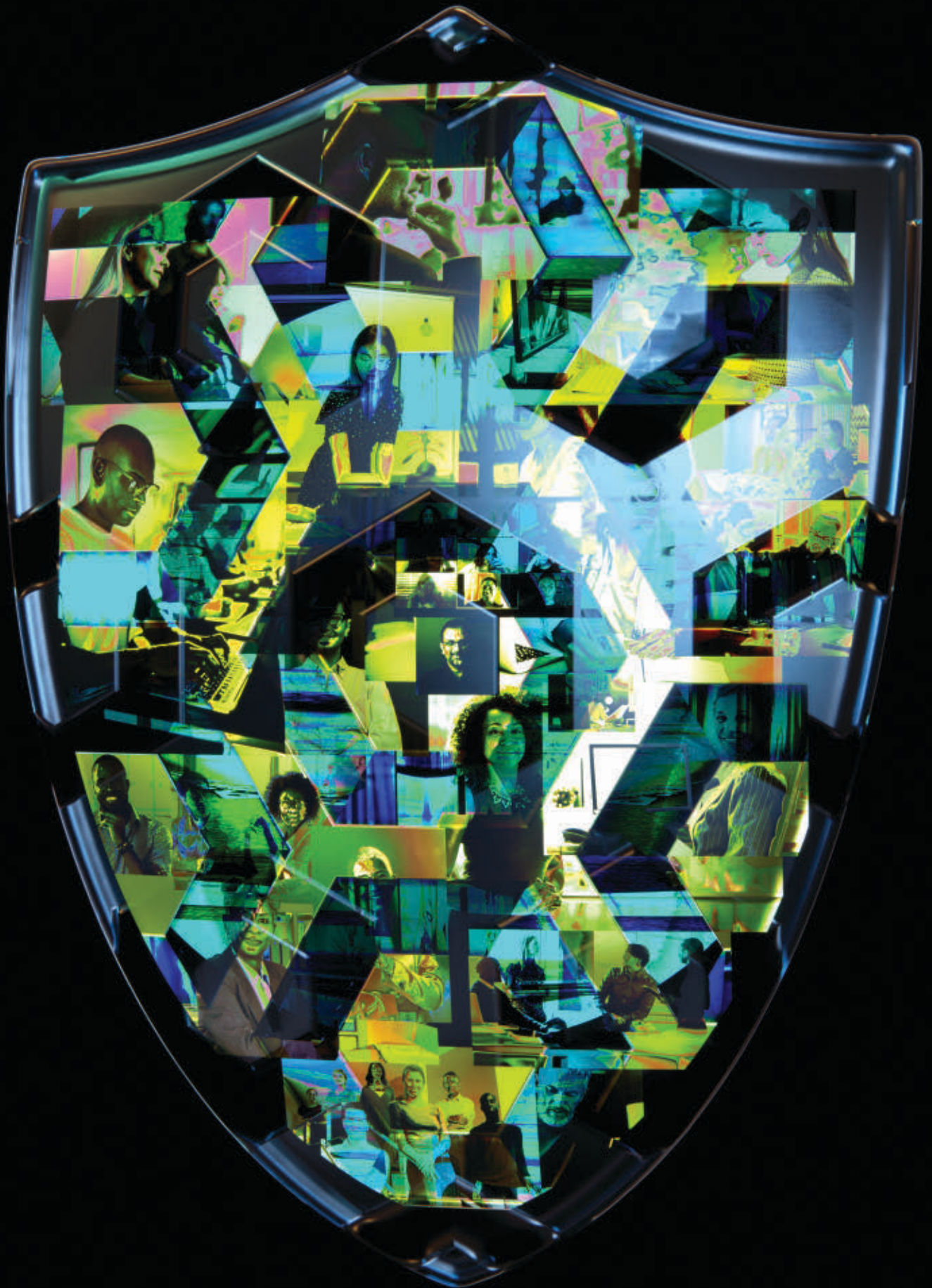
be influenced by events both inside and outside of the workplace, people have very different kinds of vulnerabilities than computing systems. But people alter their behaviors in response to incentives and disincentives and, when empowered by organizational culture, they will (unlike computing systems) respond in reasonable ways to unusual or unanticipated circumstances. Thus, the use of people in a defense both offers benefits and brings different challenges than using hardware or software.

Those benefits and challenges are the focus of this article, which is informed by some recent discussions<sup>a</sup> about best practices being employed at global IT companies and at the U.S. Department of Defense (DoD) for defense against insider attacks. The private sector and DoD are quite different in their willingness and ability to invest in defenses, in the consequences of successful attacks, and in the inclinations of their employees to tolerate strict workplace restrictions. Given those differences, two things we heard seemed striking and worth documenting for broader dissemination: How similar are the practices being used, and how these organizational structures and procedures to defend against insider threats can be seen as instanti-

a These discussions were facilitated by the National Academies Forum on Cyber Resilience.

### » key insights

- Remarkably similar practices are being used at large-scale employers that range from DoD to the private sector, despite the significant differences in employee incentives, assets to be protected, and institutional cultures.
- People, processes, and culture are key to defending against insider attacks. Many of the processes will be implemented by people doing just what computer security mechanisms do: monitoring, replication of independent function, and logging.
- Carefully audited and constrained automation of a datacenter's system administration can replace most super-user access by people.



ating some classical approaches to cyber-security.

### Assessing Risks from Insider Attacks

Risks from insider attacks will be part of any credible security assessment for an organization. In doing such an assessment, assets along with the protections they merit must be enumerated. That list is likely to include integrity and confidentiality of information about financial and customer data, confidentiality of intellectual property, integrity of system functionality, and availability of services. A risk assessment for insider attacks also requires determining which individuals and roles within the organization are being trusted and for what, as well as how those trust relationships are maintained and updated as roles change and as changes are made to the organizational structure itself. Part of this approach, articulated by Phil Veneables,<sup>9</sup> then ‘a financial services CISO and Board Director at Goldman Sachs Bank, is to understand each role in the organization and the potential impact subverting that role could have; the aim would be to ensure no individual’s role has the potential for damage that exceeds the organization’s risk tolerance. Note that operational challenges of effective and comprehensive insider risk mitigation might delay deployment until other areas of an organization’s security program are mature but understanding and communicating the insider risk is nevertheless worthwhile.

Compromised insiders not only pose a threat to an organization’s assets but are a threat to organizational stability (for example, through personnel or organizational changes made in reaction to a compromise), mission

success (for example, when critical products fail to perform as expected), and customer satisfaction. Insider attacks also are an obvious vehicle for perpetrating supply chain attacks on an organization’s immediate or downstream customers. Moreover, certain functions and activities within an organization might be sensitive enough to warrant protection from inadvertent mistakes or accidents by even trustworthy employees. Many defenses against insider attacks can serve here as well.

### Technical Controls

Classical technical controls and mechanisms play a key role in defending against insider attacks.

- ▶ Authorization prevents actions that compromise a security policy.
- ▶ Audit creates deterrence through accountability.
- ▶ Logging facilitates recovery after a security compromise.

Authorization is facilitated when the Principle of Least Privilege<sup>8</sup> is followed, since access by individuals is then limited according to need, which might be characterized (and, therefore, validated) by using past activity, time, location, and role. So, role-based access control<sup>4</sup> seems preferable, as should fine-grained privileges over coarse-grained ones.

Highly privileged administrator and operator accounts should be eschewed, which has led at least one global IT company to replace traditional system administrator root activities with automated systems that enable most data-center operations to run without the need for participation by individuals having root privileges. If fewer people have root privileges, then fewer people can abuse those privileges.

The choice between preventing action a priori versus deterrence through

accountability a posteriori often is dictated by the feasibility of monitoring, detection, and/or recovery. Can a compromise be detected in a timely way? Is recovery from a compromise possible? Of course, prevention should be preferred for actions that could lead to immediate and catastrophic outcomes.

Tamper-resistance of logs and backups is critical for implementations of accountability. For logs, tamper-resistance helps ensure attackers cannot easily change the logs to cover their tracks in order to avoid accountability. And for backups, tamper-resistance prevents attackers from installing system modifications that would perpetuate their presence after detection and restart. All accesses to sensitive data (see the sidebar “How Big Data and AI Complicate Things”) or functionality should be logged and attributed to an individual, if deterrence through accountability is intended. For programmatic access, log entries would indicate what program is running, what were its arguments, who is running the program, as well as who wrote and/or reviewed the program. Mechanisms that are guaranteed to intercept all requests and a strong form of authentication are thus a necessity for an implementation of logging.

Mechanisms to support deterrence through accountability can be further leveraged if the mechanisms also perform checks and/or pause to interrogate a developer or operator whenever sensitive or risky actions are initiated. For example, a pop-up message could inquire “Are you sure? Give a justification for your undertaking this action” whenever anyone is changing passwords, moving or removing large quantities of data, accessing highly sensitive information, and so on. Moreover, by requiring that a stylized form of justification (for example, bug identification references, support ticket, user involvement, requirements, and so on) be provided, the actor could be forced to reflect in a way that could head-off making an error. Obviously, automated tools can and should check logs after the fact to spotlight suspicious activities, such as actions involving too few individuals or where the explanation lacks detailed supporting references.

Physical security is an important element, not only to implement isolation

## How Big Data and AI Complicate Insider Protections

Recent progress in data science and in machine learning means large datasets are more prevalent. Scientists and engineers using machine learning and other AI tools are taught to examine raw data and check for unspoken assumptions needed to validate models. However, an organization concerned with protecting against insider attacks cannot have staff exploring sensitive data at will. Automated tools to validate assumptions and models would be a way to obviate the need for insider access to that data. Such tools have not yet been developed, though.



but also for authorization and for deterrence through accountability. One prevalent scheme is to require people to badge-in (use an identification badge or other unique, auditable token to gain access to specific locations) and badge-out individually. First, it creates defense-in-depth if authorization within the computing system depends on the physical location from which a request is issued. Second, deterrence through accountability is strengthened if the physical security means attacks must be instigated from physical locations where the perpetrator might be observed.

### Individuals as Monitors

Computing systems are not the only way to implement monitoring of an individual's behaviors. People can serve as monitors, too. We thus can see monitors as an underpinning for organizational structures where performing sensitive activities requires involvement by multiple individuals, observing each other. Common examples of such organizational structures that are in use today include:

- ▶ **Two-person integrity.** One person observes what a second person does. This structure, however, can be off-putting to staff, and it doubles the cost of the work being done.

- ▶ **Pilot and co-pilot or pair programming.** Two individuals are both active participants in the activity, reversing their roles periodically. Pair programming can be effective for some developers, but it can also impose costs and may pose workplace challenges for members of underrepresented or marginalized groups.

- ▶ **Maker/checker.** Widely employed within the financial industry, the *maker* creates a transaction and the *checker* approves it. High value transactions may require multiple checkers, although increasing the number of checkers can promote a climate where approval becomes a rubber stamp. In some implementations, a number of actions might be collected and the checker approves an aggregate rather than approving individual activities.

- ▶ **Poll-watching.** Used by U.S. election polling sites, this variation of maker/checker disaggregates critical sequential actions and has each performed by a different individual. In addition, inde-

pendent (uninvolved) individuals are engaged to ensure, either systematically or through spot-checking, that actions are undertaken properly.

- ▶ **Independent collaborators.** Tasks are accomplished by individuals who are sufficiently separated and independent that they are unable to collude, but sufficiently close to share a deep understanding of action and context. By selecting collaborators in a random fashion, we frustrate outsiders hoping to cultivate insiders for later compromise.

These and other organizational structures that embody *distributed trust* (so named because our trust in the aggregate exceeds our trust in its constituents) ultimately depend on assumptions about their participants. Typically, we assume a significant fraction will be trustworthy and that participants exhibit independence from each other—the cost to compromise  $N$  of them is  $N$  times the cost of compromising a single one, and the probability of collusion among multiple participants is small. There also will be an assumption that only mandated procedures are followed.<sup>3</sup> Role-based access control and other privilege assignments can help ensure that no participant in an organizational structure usurps authority and engages in actions on behalf of another. Monitoring by an independent party (typically, management) also can help check any assumptions required for an organizational structure.

Professionals in the public and private sector alike expect to be treated with respect, and constant monitoring can negatively affect their morale. The physical presence of another person necessarily creates a loss of privacy in the workplace, although expectations of workplace privacy vary by industry, sector, and type of work. Loss of privacy due to monitoring sometimes can be mitigated by also providing isolated space and time for private, solo work. In addition, an organizational culture that is explicit and public about assigning high priority to security and privacy of sensitive data (whether it's customer data or mission-critical intelligence analysis) helps staff accept that workplace privacy might not always be possible and that organizational procedures do not reflect any individual's trustworthiness.

Individuals who serve as monitors

of current actions or as analyzers of logs listing past actions could become inured to false positives. Informal discussions with managers from government and the private sector alike suggest that the risk of such burnout can be avoided if this kind of activity is limited to approximately one-third of an individual's time. And secondary benefits do accrue from having individuals check their own actions. Daily system reports of unusual actions can serve as a reminder and a training tool about actions considered suspect. However, besides training individuals about what actions are sensitive, this practice does risk training bad actors about how to avoid triggering alerts. Insiders are anyway likely to be experts in whatever automation the system they work with and likely also in how to defeat it.

Monitoring and other organizational structures for minimizing insider risk come with costs. Senior leadership must be prepared to make allowance for lower productivity, for the additional resources that will be needed, and for lowered work force morale. Even with the help of automated systems (for example, that only expose safe interfaces and thus employ prevention to block attacks), burdens imposed by security enforcement may contribute to the decision of valued staff to leave. Fortunately, security fatigue is usually given as a secondary, rather than primary, reason for these departures. And some employees—depending on their roles and responsibilities—even welcome monitoring and other tools that help reduce human mistakes or that provide means for defending against allegations of malfeasance.

Ideally, additional costs incurred to reduce the risk of insider attacks would not be a competitive disadvantage in the corporate sector. But in most sectors, today, they are. And significant, revealed insider attacks have been rare enough that the market has not incentivized expenditures for suitable defenses. Legal standards and/or regulatory approaches would be one way to a level playing field where the market is not responsive.

### Organizations as Monitors

Insiders include anyone who has access (even if unintended) to sensitive information and/or operations: em-

ployees, contractors, and friends. These different classes of individuals respond to different incentives which, in turn, affords different opportunities for defenses and requires different approaches to assessing threats.

In all cases, predicting when greater scrutiny of an individual may be needed can contain costs connected with reducing insider attacks. In addition, a model of an organization's business processes can be used to identify parts of the process that insider attacks are likely to target.<sup>1</sup>

Previous work in combining psychosocial data with cybersecurity efforts provides a path forward for identifying individuals who might warrant additional monitoring.<sup>2,5,7</sup> And in government and intelligence organizations, there is typically a small, intensely supervised group that integrates human resources and technical indicators to monitor the workforce. Private-sector best practices from the Securities Industry and Financial Markets Association similarly recommend the deployment of an institutionalized insider-threat team. However, some surveillance practices are not allowed in all jurisdictions where a multinational corporation might operate.<sup>b</sup>

Whether an individual becomes a threat is often correlated with signals from system-implemented prevention and monitoring as well as from information about non-technical activities. Staff turnover (incoming or outgoing) is one event where greater attention is typically justified. Experience has shown that theft of information is more likely to occur when an individual is preparing to leave the organization. So, monitoring indicators of staff dissatisfaction can help to anticipate such exfiltrations of confidential information. Another time to be vigilant is just after hiring—it can take time for newcomers to absorb the culture of an organization. Finally, it is wise to plan for exceptional events that require changes to a trustworthy person's normal behavior. These events



**Organizations about to implement new security policies and procedures can benefit by first identifying staff whose work demands high levels of security and reliability, since they are likely to embrace the transition.**



can range from dealing with climate and weather emergencies to operating in a country or region that suddenly finds itself in a violent conflict. Security and risk minimization processes must not be so compliance-oriented that they cannot handle complexity and extraordinary circumstances.

**Incentivizing Trustworthy Behavior**

Organizations with mature security cultures learn to treat their staff well (including staff not in security-specific organizations) while remaining slightly paranoid about damage that staff might inflict. A government intelligence agency will necessarily have different approaches and incentives in place than a private company. Insider defenses at government agencies can benefit from security clearances, background checks, and criminal penalties for disclosure of protected information. And the same general principles hold for public sector, national security, and private companies. In all, staff and senior leadership must understand that polite questions or requests for clarification are not rude and reinforce behavior when difficult cases are handled well. Existing training for compliance with the Foreign Corrupt Practices Act may already teach staff how to deal with some of these problems.<sup>c</sup>

Organizations about to implement new security policies and procedures can benefit by first identifying staff whose work demands high levels of security and reliability, since they are likely to embrace the transition. It is also wise to find staff whose work might be negatively affected or inconvenienced by new security measures, since they might need additional persuasion.

Establishing a baseline of trust and appropriate openness helps ensure that all staff are inclined to share any concerns, and that can often ease the way. In government contexts, whistleblower protections can help. Here, reported concerns should not be ignored, but overreacting could discourage borderline reports. Be care-

<sup>b</sup> An *Insider Threat Best Practices Guide* (<https://www.sifma.org/resources/general/best-practices-for-insider-threats/>) produced by the Securities Industry and Financial Markets Association cautions that “using such traits to profile insiders carries some degree of legal risk, particularly in EU member states where automated decision-making based on such profiles is restricted.”

<sup>c</sup> The Foreign Corrupt Practices Act is a law aimed at preventing companies and their senior leadership from paying bribes to foreign officials in order to assist a business deal and also imposes accounting transparency guidelines. The law applies to publicly traded companies.

ful that new security policies and procedures do not put staff into personally untenable positions, for instance, by ignoring local laws. For companies with staff located around the world, best practices for security increasingly have come to depend on location, citizenship status, secret laws on law enforcement access, border inspections of devices, the fragmentation of the internet, and sometimes even coercion of family.

To trust an individual is to assert you believe you can predict how that individual will behave in various contexts. Of course, people surprise even themselves when left alone and confronted with extraordinary circumstances. Mature security organizations recognize this fact and know that collaborative efforts with shared goals are likely to produce better results than imposing controls on creative individuals (who might simply be motivated to show how those controls can be defeated).

Staff inevitably must respond to competing demands—productivity and efficiency on the one hand versus diligence, care, and security on the other. When checks or safeguards are put in place, especially those seen as an impediment to efficiency or productivity, leadership should expect creative and amusing workarounds. One example is the use of a single password for all accounts, which is easy to generate but increases the damage from succumbing to a phishing attack.

## Discussion

There already has been a good deal of research on insider threats. That literature was recently surveyed, quite thoroughly, by Homoliak et al.,<sup>6</sup> who populate a taxonomy with the goal of systematizing knowledge and research in the area. Some of that prior work relates to our focus, by exploring behavioral frameworks and models, how organizations might put these to use, and psychological and social theory related to insider threat. But most prior work on the insider threat concerns technical aspects: defining what constitutes an insider<sup>d</sup> or an insider


d Beebe and Chang,<sup>2</sup> for example, suggest expanding the definition of an insider to include technologies within a system that have access and whose outputs are trusted by other machines and humans.

attack,<sup>9</sup> formulating security policies to defend against those attacks, designing technical means for enforcing those security policies, and creating datasets for testing mechanisms designed to detect insider attacks.

The focus of this article is non-technical means because technical means are invariably subject to compromise by insider abuses. By establishing the right culture and imposing administrative procedures, thereby enlisting trustworthy insiders to the cause, a defense in depth is achieved and a more comprehensive solution results. Rather than propose new administrative procedures, this article focuses on existing administrative procedures in use for defending against insider attacks. Specifically, we reported on practices at one large global IT company and one large DoD security organization who had not previously talked with each other about those practices. Considering the different incentives of employees there and the different kinds of assets being protected, we found it striking to see such overlap in methods that had been independently devised and deployed.

Finally, although the main thesis of this article concerns policy and administrative approaches, our discussions revealed that organizational culture and personal integrity are what matter most for building an organization that minimizes insider risk. Leadership support and buy-in is required, since mitigations can be costly. And a culture of trust and collaboration is necessary. No collection of safeguards will be sufficient to overcome a culture that is not security conscious or that lacks rigorous engineering practices. Developing and sustaining such a culture is incumbent on senior leadership.

**Acknowledgments.** Thanks to members of the National Academies of Sciences, Engineering, and Medicine Forum on Cyber Resilience for participating in discussions that contributed to this article. Special thanks to panelists at a July 2019 meeting: H. Adkins, J. Dhanraj, M. Gregory, A. Stubblefield, N. Ziring. Thanks also to Max Poletto, Susan Landau, Bob Blakley, and Brad Martin. Finally, we appreciate the helpful feedback from reviewers of our initial submission.

The Forum on Cyber Resilience is sponsored by National Science Foundation under award number CNS-14194917, the National Institute of Standards and Technology under award number 60NANB16D311, and the Office of the Director of National Intelligence under award number 10004154. Schneider is supported in part by AFOSR grant F9550-19-1-0264, and NSF grant 1642120. 

## References

1. Bishop, M., Conboy, H.M., Phan, H., Simidchieva, B.I., Avrunin, G.S., Clarke, L.A., Osterweil, L.J., and Peisert, S. Insider threat identification by process analysis. In *Proceedings of the Security and Privacy Workshops*. IEEE, Los Alamitos, CA, 251–264.
2. Bishop, M., Engle, S., Frincke, D.A., Gates, C., Greitzer, F.L., Peisert, S., and Whalen, S. A risk management approach to the 'Insider Threat'. *Insider Threats in Cyber Security: Advances in Information Security*. C. Probst, J. Hunker, D. Gollmann, M. Bishop, eds. Springer, Boston, MA, 115–137.
3. Clark, D.D. and Wilson, D.R. A comparison of commercial and military computer policies. In *Proceedings of the 1978 IEEE Symp. on Security and Privacy*. IEEE Computer Society Press, 184–194.
4. Ferraiolo, D.F. and Kuhn, D.R. Role-based access controls. In *Proceedings of 15<sup>th</sup> Nat. Computer Security Conf. National Institute of Standards and Technology, National Computer Security Center*, (Oct. 1992), 554–593.
5. Greitzer, F.L. and Frincke, D.A. Combining traditional cyber security audit data with psychosocial data: Towards predictive modeling for insider threat mitigation. *Insider Threats in Cyber Security: Advances in Information Security*. C.W. Probst, J. Hunker, D. Gollmann, and M. Bishop, eds. Jan 2010, Springer Science+Business Media LLC, New York, NY, 85–113.
6. Homoliak, I., Toffalini, F., Guarino, J., Elovici, Y. and Ochoa, M. Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures. *ACM Comput. Surv.* 52, 2, (Mar. 2019), Article 30; <https://dl.acm.org/doi/pdf/10.1145/3303771>
7. Kandias, M., Mylonas, A., Virvilis, N., Theoharidou, M. and Gritzalis, D. An insider threat prediction mode. In *Proceedings of the 7<sup>th</sup> Intern/ Conf. Trust, Privacy and Security in Digital Business*. S. Katsikas, J. Lopez, and M. Soriano, eds. Springer-Verlag, Berlin, Germany, 2010, 26–37.
8. Saltzer, J.H. Protection and the control of information sharing in multics. *Comm ACM* 17, 7 (July 1974), 388–402.
9. Venables, P. Insider Threat Risk—Blast Radius Perspective. (Dec. 1, 2019); <https://www.philvenables.com/post/insider-threat-risk-blast-radius-perspective>

**Eric Grosse** ([grosse@gmail.com](mailto:grosse@gmail.com)) is a private consultant in Los Altos, CA, USA.

**Fred B. Schneider** ([fbs@cs.cornell.edu](mailto:fbs@cs.cornell.edu)) is the Samuel B. Eckert Professor of Computer Science at Cornell University, Ithaca, NY, USA.

**Lynette I. Millett** ([lmillett@nas.edu](mailto:lmillett@nas.edu)) is Senior Program Manager of the Computer Science and Telecommunications Board at the National Academies of Sciences, Engineering, and Medicine, Washington, D.C., USA.

© 2021 ACM 0001-07/21/5



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

DOI:10.1145/3418291

**As a new era in computing emerges, so too must our fundamental thinking patterns.**

BY YUHANG LIU, XIAN-HE SUN, YANG WANG, AND YUNGANG BAO

# HCUDA: From Computational Thinking to a Generalized Thinking Paradigm

IN 2006, JEANNETTE M. Wing<sup>45</sup> proposed the concept of “computational thinking,” which has produced significant worldwide impacts on the education, research, and development of computer science. After more than a decade, we reexamine computational thinking, and find that a more general-thinking paradigm is urgently needed to address new challenges.

A couple of recent commentaries<sup>12,41</sup> regarding computational thinking attracted our interests and inspired us to reflect further. More than that, we want to summarize and generalize the rationale of our solutions,

for instance, the Labeled von Neumann Architecture (LvNA)<sup>1,28</sup> and the Layered Performance Matching (LPM) methodology.<sup>26,27</sup>

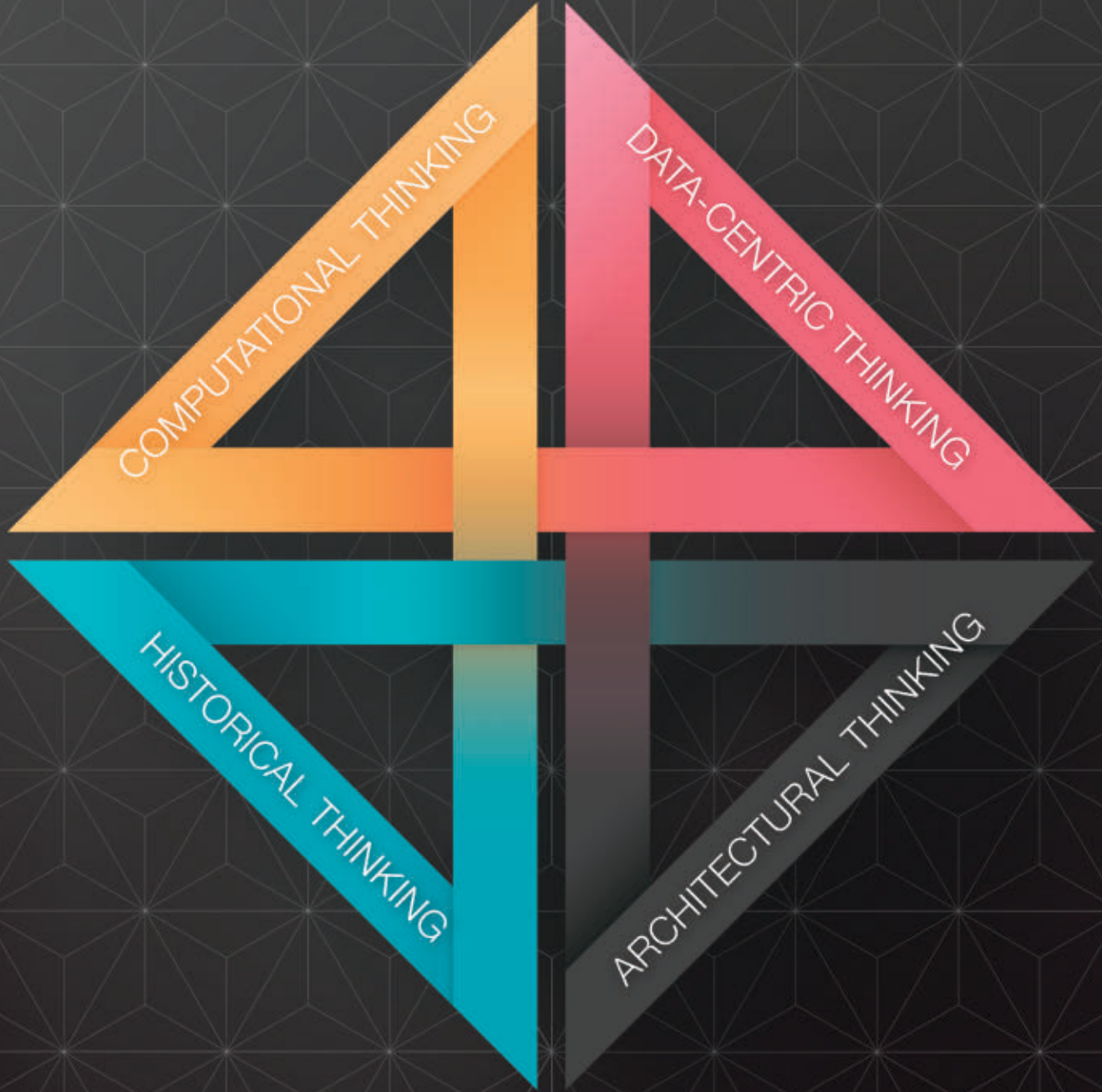
Nurtured by Moore’s Law, the number of transistors available on a single chip increases exponentially. Meanwhile, due to architectural innovations, transistors are organized more effectively and utilized more vigorously. As a result of the combined efforts, computers have witnessed a significant performance advancement during their 70-year history. However, the new age, characterized by the slowdown of Moore’s Law and Dennard scaling,<sup>44</sup> and by the rise of big data applications, brings serious challenges that computer scientists must face.

The scaling of on-chip transistors impacts microprocessor performance significantly. However, further improvements to transistor density and power become more difficult due to the limits of semiconductor physics.<sup>44</sup> As a result, architectural innovations become increasingly crucial for performance breakthroughs, and the epoch we are entering is “a new golden age for computer architecture.”<sup>14</sup>

The rise of big data has caused an unprecedented shift, where the memory system, rather than the computational core, plays a more vital role. Accordingly, the memory access limitation de-

## » key insights

- **Architecturally organizing the computing system and exploiting the available transistors more effectively can reduce the number of transistors required, and the labeling and matching are crucial in architectural design.**
- **Computing history is a growing warehouse that has not been well utilized, and the evolution of computing technology is not always linear but has lots of “reuses” or “rising from the ashes,” and hence historical thinking is needed.**
- **Memory access can become a killer performance bottleneck for high performance computing, and simply adding more on chip memories is not a feasible solution to the memory-wall problem.**



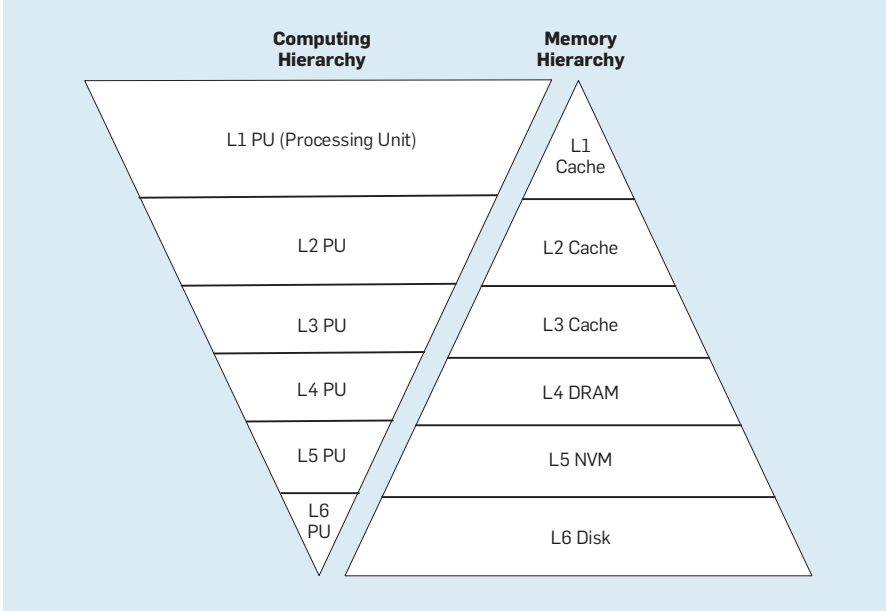
COMPUTATIONAL THINKING

DATA-CENTRIC THINKING

HISTORICAL THINKING

ARCHITECTURAL THINKING

**Figure 1. The symmetry between memory and computation from a hierarchy perspective.**



scribed by Sun-Ni’s Law<sup>38</sup> is becoming a performance killer for many applications. Thus, data-centric innovations of computer system design are urgently needed to address the issues of data storage and access. Both the emergence of big data and the slowdown of Moore’s Law have changed the landscape of computer systems and require us to examine past solutions to pave a new path for future innovations and for addressing new challenges.

The first step to learn from extant innovations is to find commonalities and patterns within them. In the seminal literature “The Art of Scientific Research,”<sup>3</sup> Beveridge summarized the research methods adopted by classical natural scientists and found commonalities, routines and regularities in the innovation process, which was previously thought to be accidental. By following Beveridge’s analysis methodology, from a thinking pattern perspective, we try to summarize and therefore provide some useful hints for the art of computer science research.

One interesting observation that motivates us in this study is the symmetry between computation (data processing) and memory (data access). The name “computer” is somewhat misleading, leaving an impression that the only focus of “computer” is computation. In contrast to the traditional computation-centric design, the newly emerged memory-centric computer system design views memory as the in-

trinsic bottleneck of the von Neumann architecture.

Before turning to non von Neumann architectures (for example, quantum computing<sup>20</sup> and DNA computing<sup>6</sup>), we should determine whether it is possible to design new von Neumann architectures that are still transistor-based but can overcome the memory wall. After examining the original representation of von Neumann,<sup>42,43</sup> we found that von Neumann preserves neutrality between computation and memory. That is, data processing and data access are equally important according to the von Neumann architecture. Computation and data are two sides of the same coin. They are the premises of each other. Therefore, equilibrium (a.k.a. balanced) design is the must without any room for maneuver.

The traditional computation-centric architectural design is due to historical reasons. In the beginning, CPUs were very slow and expensive compared to memory. As a result, computer systems are computation-centric, focusing on the speed of computing components. Later, due to the mismatch in advancement of processor and memory technologies, computer systems became increasingly memory-centric where data was treated as a first-class citizen. From a historical perspective, the later memory-centric focus counteracts the former computation-centric focus.

More specifically, as shown in Fig-

ure 1, the memory hierarchy is the counterpart of the computational hierarchy, memory-level-parallelism (MLP) is the counterpart of instruction-level-parallelism (ILP), and processor in memory (PIM) is the counterpart of memory in processor (MIP).<sup>a</sup> Therefore, in this study, we propose a framework that allows computational thinking and data-centric thinking to naturally coexist.

The thought process of a computer system designer is dynamic, but it has patterns. Wing introduced the important concept of “computational thinking” in 2006.<sup>45</sup> In this study, we build on her ideas to extend the concept of computational thinking and present a paradigm of thinking patterns for computer researchers and practitioners in the new era of computing.

More generally, we identify four foundational thinking patterns that will foster computer system innovations: *historical thinking, computational thinking, data-centric thinking, and architectural thinking* (HDCA). The combination of these four patterns forms a regular tetrahedron thinking paradigm, where history is the source of references and lessons, computation is the functionality provided by the computing systems, data is the processing objects of computing systems, and architecture is the hardware mechanism to physically execute operations. The four thinking patterns are four dimensions that are logically connected, providing a unified framework for computer design innovations. This thinking paradigm will provide a reference model for researchers and practitioners to guide and boost the next wave of innovations in computer systems.

### Historical Thinking

Historical thinking is the first point of the regular tetrahedron. Present innovations are based on past innovations. Revolutionary innovations are based on the accumulation of knowledge. Historical thinking explores solutions and methodologies from past results, lessons and experiences.

<sup>a</sup> State-of-the-art processors usually have multiple levels of cache on a single chip. For instance, Fujitsu’s 48-core A64FX processor used in Fugaku supercomputer has 32MB L2 cache.<sup>40</sup>

As Grier noted,<sup>13</sup> many researchers seldom read literature published over five years ago. They only pay attention to the immediate concerns but miss the wealth of important ideas and methodologies accumulated in the field of computing for many years. That limits their vision and often leads to researchers “re-inventing the wheel.” One instance is “The Computer and the Brain,” an unfinished book written by John von Neumann more than 60 years ago, which includes many valuable ideas that today’s researchers could benefit from. Unfortunately, many have not read it.

Historical thinking urges us to reflect on the successful innovations from the past by reading more historical literature in the field, and to learn more from others’ experiences to facilitate our research. This kind of reading and “casual” study will extend our horizons and stimulate our imaginations, further motivating us to think outside the box and start on solid ground.

Historical thinking is a necessity rather than a luxury, and it plays several vital roles in scientific innovation. Armed with a rich historical perspective, computer scientists could frame their work within the broader advancement of human civilization and give them insight into the innovation process. The contributions of computer science research toward the well-being of mankind should be fully recognized and emphasized historically.

Through historical thinking, we can discover new opportunities on the ground of past experiences. The creativity of individual scientists undoubtedly plays an important role in their success. However, the impact and necessity of joint efforts that usually last for a long time and cross geographic lines should not be ignored. History shows that a scientific discovery is generally the result of many scientists’ continuous efforts. Scientists rarely work from scratch; instead, they usually extend and follow up others’ work.

Taking one example of our work, by taking inspiration from Amdahl’s Law in 1967 and Gustafson’s Law in 1988, Sun and Ni developed the memory-bounded speedup model (often referred to as Sun-Ni’s Law) in 1990, which not only extends and unifies Amdahl’s Law and Gustafson’s Law, but also sheds light on the trade-off be-

tween computation and memory for scalable computing.<sup>38</sup> These three laws have been incorporated into many textbooks and taken as classic principles in the field of supercomputing. As another example, by extending the well-known memory performance model, AMAT (Average Memory Access Time), we proposed C-AMAT (Concurrent AMAT) in 2014<sup>39</sup> to characterize the memory access delay in modern computing systems where data access concurrency is pervasive. Furthermore, by combining Sun-Ni’s law and C-AMAT, a new performance model, C<sup>2</sup>-bound (the square bound of memory capacity and concurrency), was proposed to show the combined effects of memory capacity and concurrency in 2015.<sup>25</sup>

Figure 2 depicts a historical progression of memory modeling with respect to changing architectures, from AMAT to C-AMAT, from Amdahl’s Law to Sun-Ni’s Law, and from Moore’s Law to Hill-Marty’s model.<sup>15</sup> These new models are technology driven. For instance, the introduction of CAMAT is due to the prevalence of concurrent data access in modern computing systems. The Hill-Marty’s model is an extension of Amdahl’s Law to multi-core design (please note, though not listed in Figure 2, paired with Hill and Marty, Sun and Chen also extended the scalable computing concept to multicore design<sup>37</sup>). With the advancement of technologies, each of the models has its own historical significance.

The evolution of technology is not always linear but has lots of “reuses” or “rising from the ashes.” For example, the contemporary Google TPU is the result of a team of history-aware architects.<sup>18</sup> Three important architectural

features could date back to the early 1980s: systolic arrays<sup>23,24</sup> decoupled-access/execute,<sup>35</sup> and Complex Instruction Set Computer (CISC) instructions.<sup>32</sup> These features are historical “re-runs.” Some of them did not become the mainstream when they were first introduced, but they are effective and valuable in the TPU’s design.

Based on historical thinking, we can focus on computation and data separately, to develop computational thinking and data-centric thinking, respectively.

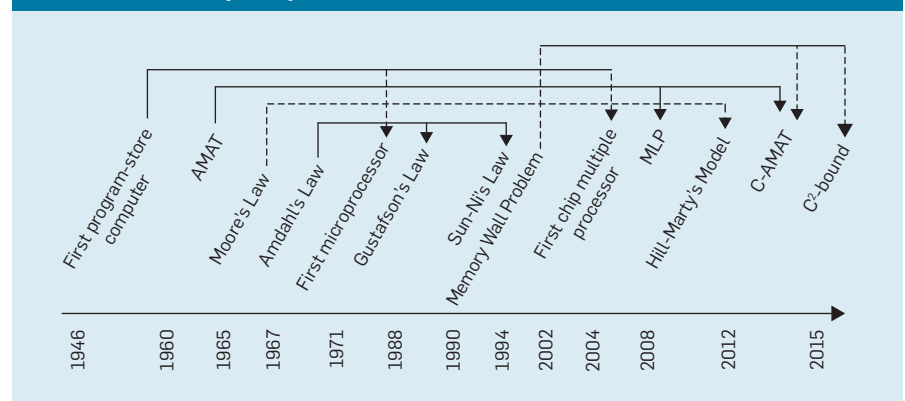
### Computational Thinking

Computational thinking (aka computation-centric thinking) is the second point of the regular tetrahedron. Computational thinking has two different meanings from the perspectives of mechanism and behaviorism, respectively. Mechanism focuses on “how” and “performance”, while behaviorism is concerned with “what” and “functionality.” Therefore, computational thinking focuses on making the machine more functional (for example, via AI) and more quickly (for example, via parallel computing).

From a mechanism perspective, computational thinking implies computation-centric design, making the speed of computing components as fast as possible. Pursuing high peak performance of a computer is such an example. On the other hand, from a behaviorism perspective, computational thinking implies solving problems via computation.

In the sense of mechanism, an essential feature of computational thinking is the creation of microarchitectures and the design of computational structures to assist data processing with re-

**Figure 2. An instance of historical thinking that shows the machines, laws, and models related to the memory wall problem.**



gards to the performance of a problem solver. Time and patience are scarce resources for mankind, so we want to make computers faster and faster, and make the turn-around time of problem-solving as small as possible.

In the sense of behaviorism, the goal of computational thinking is to make the computer more versatile by increasing its utility for an ever-expanding set of applications. Computers are intelligent and powerful machines that can execute different programs for different purposes. For instance, the 2013 Nobel Prize in chemistry was awarded to three scientists due to their contributions to the computational model.<sup>33</sup> As the Nobel committee pointed out, “computer models mirroring real life have become crucial for most advances made in chemistry today, and computers unveil chemical processes, such as a catalyst’s purification of exhaust fumes or the photosynthesis in green leaves.”<sup>30</sup> It is simply more difficult and time-consuming to obtain the same results that one would with a simulation via traditional experimental methods, that is, in a wet laboratory.

Computational thinking enables us to understand computation across multiple phases of human history. Computational thinking existed far before the birth of modern computers. For example, humans had a tool to conduct decimal multiplication as early as 305 BC.<sup>32</sup> The 2,300-year-old matrix hidden in Chinese bamboo strips was found to be the world’s oldest decimal multiplication table. Before modern computers were invented, a variety of computational tools had already been developed, including ropes, rods and abacuses. Computation is already a crucial part of our daily life, with a profound impact on production and living quality. With computation as a tool, mankind can pursue activities that would be impossible without it. Moreover, with high performance computing, mankind can solve problems that would be prohibitive with traditional computational methods.

Computational thinking also requires us to understand computation in a multidisciplinary manner. The usefulness and broad application of computers benefit from the involvement of domain experts from other fields (for



**Both the emergence of big data and the slowdown of Moore’s Law have changed the landscape of computer systems and require us to examine past solutions to pave a new path for future innovations and for addressing new challenges.**



example, biology, chemistry and physics), rather than just computer scientists.

Jeannette M. Wing described computational thinking as using abstraction and decomposition when solving real problems and designing large complex systems.<sup>45</sup> Thinking like a computer scientist means more than being able to program a computer. Instead, it requires thinking at multiple levels of abstraction and examining what computers and automated processes can do. In this way, scientists in diverse fields can extend the functionality of computers to solve more problems.

Computational thinking can greatly improve the level of machine automation and relieve humans of performing tedious computational processes. To illustrate, recall that computers based on the von Neumann architecture take an instruction sequence (program) as a generalized stream of data items, then executes the stream sequentially, guided by the program counter. Computers are powerful and accurate but can only run executable programs. The ability to transform scientific problems into computable mathematical models is essential to computational thinking and must be trained and learned.

### **Data-Centric Thinking**

Data-centric thinking (aka data thinking) is the third point of the regular tetrahedron. Like computational thinking, data-centric thinking also has behavioral and mechanistic meanings in the view of external and internal characteristics of data, respectively. From a behaviorism perspective, data-centric thinking implies solving problems via data. From a mechanism perspective, data-centric thinking implies memory-centric design, making the impact of data access delay as small as possible.

As massive data becomes available, the interests and feasibility of knowledge discovery are also increased. For instance, Google found that if the number of Internet searchers for “flu” suddenly peaked in a region, that region might be experiencing a flu epidemic. Internet search activity can be considered as one form of knowledge discovery, while discovery of the link between search activity and the flu epidemic can be treated as another form.

Data-centric thinking solves prob-



lems by means of collecting and utilizing data. While computational thinking focuses on formulating a problem to make it computationally solvable, data-centric thinking is for gathering and exploiting data to provide insights.<sup>2,29,24</sup> For instance, suppose we want to provide a set of travel plans for users. Using computational thinking, we might identify the shortest path along a graph (representing cities and routes) with Dijkstra’s algorithm or the Bellman-Ford algorithm. Here, the graph is the abstract mathematical model of the travel planning. Creating this graph mathematical model is a process of computational thinking, as well as the shortest-path algorithms. On the other hand, with data-centric thinking, we no longer need to focus on computation. Instead, we focus on collecting historical route data, analyzing, and exploring the data, and finally making a recommendation to users based on the data. If the recommendation is generated based on deep learning or other AI methods, we often do not know exactly the reasoning behind the recommendations. In other words, data thinking solutions are weak in “causality and interpretability” than mathematical model-based solutions, as stated in Liu et al.,<sup>29</sup> while they may be more effective or even the only way to find a solution.

As shown in Figure 3, data-centric thinking is needed to address the challenges of knowledge discovery (aka data mining), which is a daunting task similar to ore smelting, because it is difficult to decide what kinds of data should be collected and how to exploit knowledge from a large amount of data.

The 4V (that is, Volume, Variety, Velocity, and Veracity)<sup>4,17</sup> characteristics of big data applications have a significant impact on memory localities and have changed the landscape of computing. Big data applications usually have a large quantity of data with poor locality. They are nightmares for the modern memory hierarchy, leading to data starvation that causes processor pipeline stall, known as the memory-wall problem. Local disks and interconnection networks for remote access also can be considered a layer of a more generalized memory hierarchy

The Fugaku supercomputer, currently the world’s fastest computer, only had a peak efficiency of 80.86% when it was first released.<sup>5</sup> Notably,

for a supercomputer, that is a remarkable achievement, and was the result of a long and labor-intensive performance tuning effort for a particular application (that is, LINPACK). In practice, a high system utilization is very difficult to reach, and the achieved sustained efficiency is often in the single-digit range.

Data access can become a killer performance bottleneck for high performance computing. Simply adding more on-chip memories is not a feasible solution to the memory-wall problem. Deep multi-level cache hierarchy involves more than 80% of the microprocessor on-chip transistors, while studies show that on average these cache blocks are never used during more than 80% of their lifetime.<sup>19</sup> These unutilized cache blocks not only waste power consumption and die spaces, but also increase data searching time. We need to have data-centric thinking toward both the hardware and system designs, to increase system efficiency, and to reduce memory stall time at the same time.

Data-centric thinking for system design involves the whole life cycle of data processing, from data collection, data storage, to data movement and operation. It places the memory system at the highest priority, at least at the same level as the CPU. As early as 1990, we revealed that memory is a major constraint of scalable computing and introduced the memory-bounded speedup model.<sup>38</sup> More recently, we established the C-AMAT model to promote and utilize concurrent data access.<sup>39</sup> These models are embodiments of data-centric thinking for computer system designs. Big data infrastructures and supercomputers have different design considerations and are with indi-

vidual ecosystems. Since data access is as important as data processing (that is, computing), big data infrastructures and supercomputers will coexist for a long time, and need to support and complement one another, resulting in a converged unified eco-system.

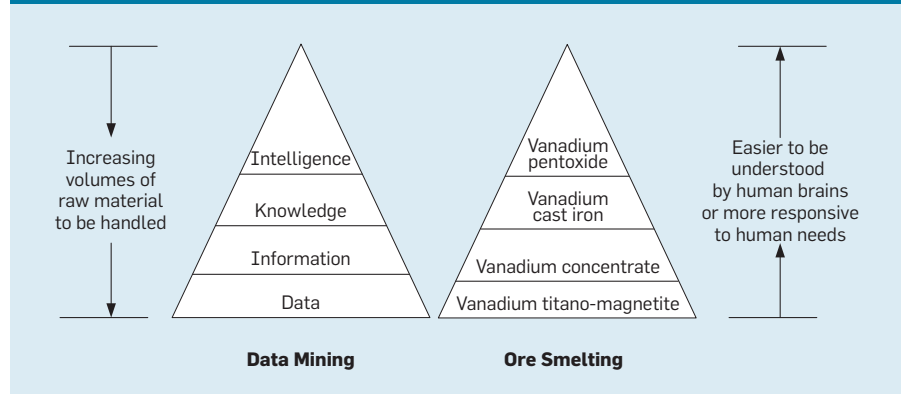
### Architectural Thinking

Architectural thinking is the fourth point of the regular tetrahedron. Architectural thinking refers to utilizing existing transistors to build optimized computer systems through hardware and firmware designs. In general, the term “computer system” could mean a hardware system, a software system or a combination of the two. Architectural thinking focuses on hardware systems. The emphasis on utilizing existing transistors implies that we no longer merely call for more transistors but focus squarely on their optimal organization and configuration.

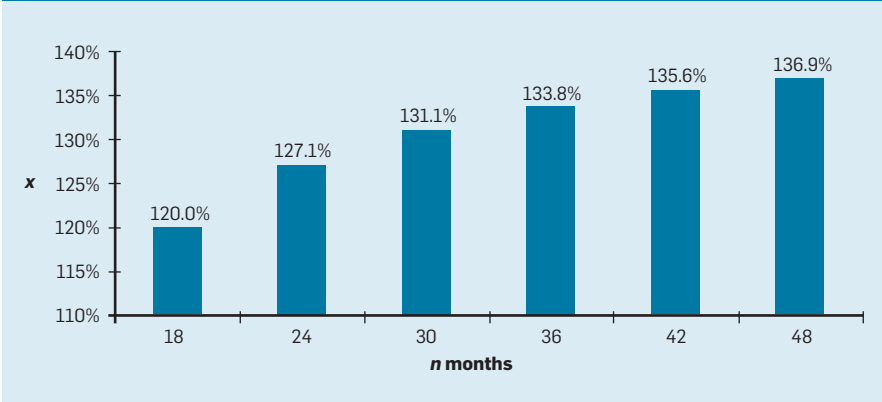
Architectural thinking can be conducted at different levels, the highest of which is the system level, which includes processor, memory, network, and input and output (I/O). For instance, the choice of adopting MCDRAM<sup>36</sup> and NVRAM<sup>11</sup> has revolutionized the organization of memory systems. This organization and configuration can be extended to the chip level and component level as well. For example, the number and connection of cores, the capacity of on-chip caches, the cache management (insertion, promotion and replacement), and the design of solid-state drives (SSDs) all involve architectural thinking.

Computer architecture is facing many new challenges. First, conventionally, the “best” system is measured in CPU performance; now, as discussed, to optimize CPU performance

Figure 3. Analogy between data mining and ore smelting.



**Figure 4. Architectural innovation  $x$  needs to be increased with the slowdown of Moore's Law (Assume the number of transistors in a unit area doubles every  $n$  month, and the performance improvement that must be contributed by architectural innovation is  $x$ ).**



we first need to consider memory performance in addition to memory power consumption. Second, even for conventional CPU-centric architectural design, we cannot merely rely on Moore's Law to improve CPU performance.

The International Technology Roadmap for Semiconductors (ITRS) has called off the pursuit of Moore's Law and stopped the half-century practice.<sup>44</sup> Note that the fastest supercomputer, Fugaku, reached 415.5 petaflops of performance in June 2020.<sup>40</sup> Assume the number of compute nodes in a supercomputer is fixed.<sup>b</sup> If we want to achieve exascale (1,024 petaflops) compute speed before June 2021, we have only one year and require a 2.46-fold (1024/415.5) speedup. In other words, on average about 146% performance improvement per year is required.

Realizing 146% per year performance improvement is a daunting challenge. According to Moore's Law, the density of transistors is doubled every 18 months, which amounts to a 58% density increase per year. According to Pollack's Rule,<sup>5</sup> the performance is proportional to the square root of the number of transistors, and therefore 58% density increase can be restated as 26% performance improvement. As a result, even with the contribution of Moore's Law, we still must have an ad-

ditional 120% performance improvement to fulfill the 146% per year requirement. However, the speed of density increase has slowed down, and the density of transistors has been doubled every three years since 2010, which is only 50% of what Moore's Law predicted.

Assuming the number of transistors in a unit area doubles every  $n$  months, and the annual rate of growth is  $a$ , we then have

$$(1 + a)^{\frac{12}{n}} = 2$$

Therefore, according to Pollack's rule, the increase in the number of transistors can be converted into performance improvement by  $b$ -fold,

$$b = (1 + a)^{\frac{1}{2}} - 1$$

By combining Eq. (1) and Eq. (2), we have

$$b = 2^{\frac{6}{n}} - 1$$

By Eq. (3), as  $n$  increases, the performance improvement due to non-architectural innovation,  $b$ , is decreasing. Assume the performance improvement that needs to be contributed by architectural innovation is  $x$ , where the sum of  $x$  and  $b$  is expected to be 146% (that is, the total performance improvement is 146%), then by Eq. (3) we can obtain

$$x = 2.46 - 2^{\frac{6}{n}}$$

By Eq. (4),  $x$  increases with  $n$ , and the relation between  $x$  and  $n$  is shown in Figure 4. As Moore's Law begins to

slowdown, the requirement for architectural improvement will grow from 120% to 136.9% as  $n$  increases from 18 to 36 (see Figure 4). This is a new challenge for architectural design.

Architecture is the hardware skeleton for running algorithms, impacting the efficiency of computer systems and the quality of user experience. For example, an 8-core processor developed by our colleagues has attained a 35% improvement in efficiency compared with its predecessor.<sup>16</sup> The significant performance improvement is mainly attributed to the innovations of its architecture: the input/output structure and the associated bandwidth and latency are improved by upgrading the interconnect and the memory interface.

From the early mainframe computers to the millions of processor cores in present supercomputers, there is an analogy to the "few but giant" and "small but many." By this analogy, as shown in Figure 5, several dimensions need to be considered for the trade-off between the components in a computer system, including uni-core vs. multi-core, reduced vs. complex, shared vs. private, distributed vs. centralized, latency vs. bandwidth, locality vs. concurrency, homogeneous vs. heterogeneous, synchronous vs. asynchronous, general purpose vs. special purpose, and so on. Many works published on top conferences and journals on computer architecture can be mapped onto different positions in the huge design space shown in Figure 5. For instance, domain-specific hardware accelerators<sup>8</sup> realized by ASICs, FPGAs, or GPUs are for special-purpose computing, conventional CPUs are for general purpose computing, and architecture design choices are laying between the two extremes.

Architectural thinking has several dimensions, each having many choices. We assume an architecture A including  $n$  different components ranging from computation, to memory access and communication. For any component  $p_i$  in a computer system, there exist multiple ( $N_i$ ) design choices. According to the combinatorics, all the combinations of choices constitute a huge design space, which is of size  $\prod_{i=1}^n N_i$ . Even if the design space only has 10 dimensions and each dimension only has 10 different options, there would be 10 billion ( $10^{10}$ ) possible configurations.

<sup>b</sup> This assumption is intended to save resources and maintain system size, but in practice people tend to increase the overall performance of the supercomputer by increasing system size. For instance, as the fastest supercomputer in the top500 list, Fugaku's performance is 2.8 times that of the runner-up (Summit), but the former has 3.02 times more processor cores than the latter.<sup>40</sup>

What makes the issue even more challenging is the effectiveness of architectural design is application dependent. The huge space of applications/workloads must be considered in an architectural design. In addition, there are a series of constraints in terms of energy consumption, temperature, area, as well as cost and performance. An architectural innovation must satisfy these constraints under current technologies, while optimized to serve a variety of applications. However, it is difficult for a single architecture to fit all applications to deliver the highest performance. A feasible way to achieve this goal is to discover an elastic and flexible structure, which can dynamically map application workloads onto the architectural design space. Given the size of the design space, an exhaustive search of architectural and application pairings is extremely difficult, if not impossible.

### Discussion on Solutions

Thinking paradigm can guide us to address research issues more effectively. In the following, we give some examples on how our thinking paradigm helped us in research:

We start with historical thinking. First, after investigating and examining a large number of solutions and literature, we find two keywords that can summarize the commonalities of the solutions of memory system optimization. They are “labeling”<sup>1,21,28</sup> and “matching.”<sup>26,27</sup>

Then we apply computational thinking and data-centric thinking to reason about the necessity of “labeling” and “matching.”

Memory accesses are the ties between computing components and data components. Without memory accesses, there would be no computer systems but only separated components. While many computer nodes form a network, the inside of each computer node is also a network in which hardware components communicate via network packets (for example, over the NoC or PCIe).

Finally, inspired by the above observations, with architectural thinking, we proposed the labeled von Neumann architecture.<sup>1,28,46</sup> The architecture has and only needs to have three features Existence, Simplicity, and Utilization referred to as ESU with regards to labeling:

► **Existence.** Each memory or I/O ac-

cess is attached a high-level semantic label (for example, a virtual machine or thread ID) to explicitly convey high-level information to the underlying hardware.

► **Simplicity.** Labels should be as simple as possible to travel across the data path of a machine in which different hardware components can identify labels and fulfill control policies based on labels.

► **Utilization.** Control logic is introduced on the data paths for leveraging labels to achieve specific goals such as quality-of-service and security to increase the utility of the hardware.

For ESU, the first two features that are about the label representation provide the foundation for the third feature to achieve the matching between computation and data. The ESU principle is to efficiently enable a computing system to have three abilities: Distinguishing, Isolation, and Prioritizing, referred to as DIP.<sup>46</sup> A computing system with DIP can handle the resource contention among concurrent tasks and can eliminate the uncertainty around data access latency to guarantee user experience.

Also, with computational thinking and data-centric thinking, we notice that the symmetry between computation and data requires the matching be-

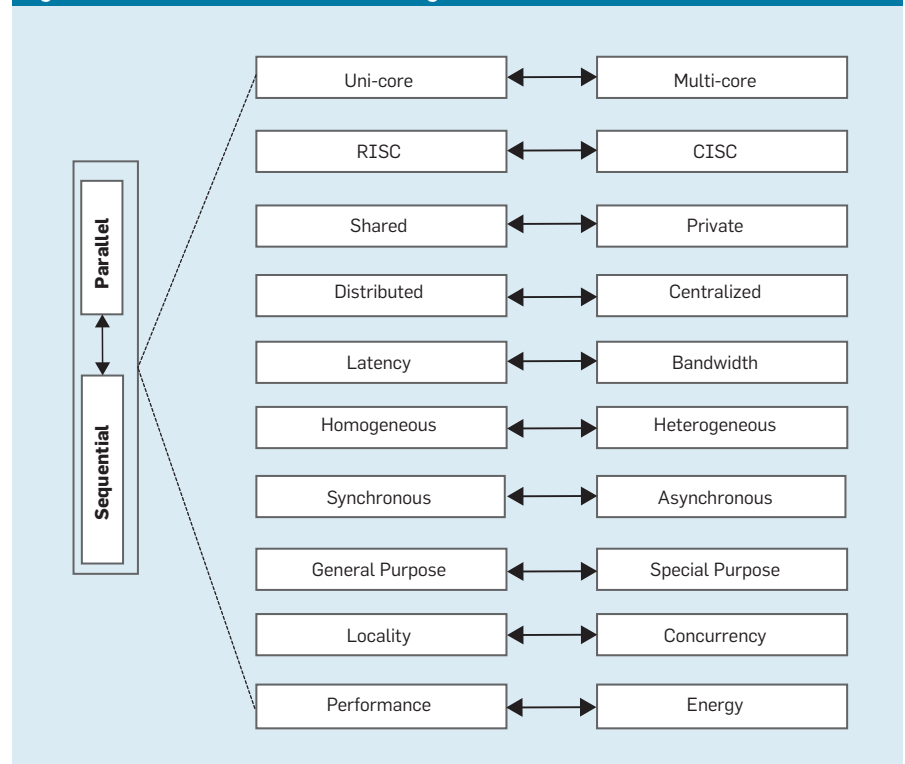
tween application data access patterns and memory system architectures. To this end, we proposed an architectural optimization methodology—Layered Performance Matching (LPM)<sup>26,27</sup> to achieve the desired matching.

With architectural thinking, we designed light-weight hardware structures to quantify and detect the data request-reply matching degree and utilized diverse hardware economically to explore data access concurrency and locality until the mismatch is eliminated. As with Jason Cong<sup>7</sup> in examining the customized computation, we believe this matching is a representative direction of architectural innovations.

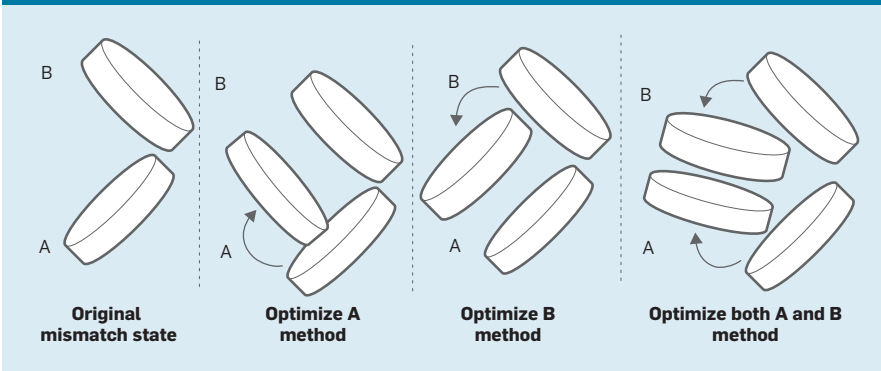
Architectural thinking urges us to consider whether to address issues in a codesigned manner. Notice that besides the hardware approach, the LPM optimization also can be achieved in a software approach. Generally, a matching of *A* and *B* can be achieved by reorganizing *A* to match *B*, or reorganizing *B* to match *A*, or simultaneously reorganizing *A* and *B* to agree with each other. Figure 6 depicts these three methods.

In practice, for layered performance matching, if the architecture configuration is *A*, and the application data access pattern is *B*, there will exist three different optimization approaches. The meth-

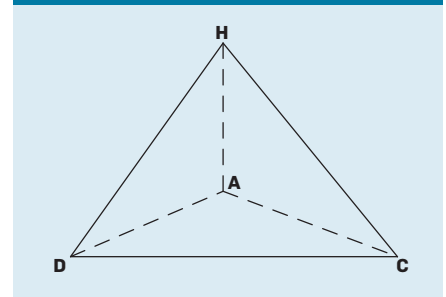
Figure 5. Dimensions of Architectural Design Trade-off.



**Figure 6. Three different matching methods (A is the “the underlying memory system architecture,” while B denotes “application data access pattern”).**



**Figure 7. A Framework of the Four Thinking Patterns (“H” represents “Historical thinking,” “A” represents “Architectural thinking,” “D” represents “Data-centric thinking,” and “C” represents “Computational thinking”).**



od optimizing *A* is a hardware approach, the method optimizing *B* is a software approach, and the method optimizing both *A* and *B* is a mixed approach. For the hardware approach, the architecture configuration can be adapted online to the data access patterns of applications. On the other hand, for the software approach, the architecture configuration is fixed but with heterogeneity, and through scheduling we also can achieve a better match of the underlying memory systems for a better performance. The scheduling can be implemented in two ways. We can schedule across heterogeneous processors to allocate application data to the underlying hardware according to its data access pattern. Alternatively, we can reorder data accesses at the memory controllers to reshape application data access patterns to adapt to the underlying hardware.

We have illustrated how the generalized thinking paradigm helped us in our research, which includes the labeling and matching designs for computing systems. The formal identification and recognition of the thinking paradigm certainly will boost innovation and productivity.

**Discussion on Relations**

As shown in Figure 7, the four thinking patterns make up a triangular pyramid, which has six undirected edges (12 directed edges) that correspond to six relationships, that is, H-A (History-Architecture), H-D (History-Data), HC (History-Computation), A-C (Architecture-Computation), A-D (Architecture-Data), and C-D (Computation-Data).

In HCDA, the six undirected edges are six dimensions of the paradigm we can follow. In each dimension, there are two directions, thus the 12 directed

edges are 12 directions. For example, the dimension “H-A” can be separated into “H→A” and “A→H”. “H→A” means using historical knowledge and experience to boost architectural innovation. “A→H” denotes examining and viewing current architectural innovation from a historical perspective. It is not uncommon that a long-existing, warehoused technology resurged from the past history to become modern success. A good example is the deep learning technology. It was first proposed in 1986,<sup>10</sup> and only found its stunning success more than 20 years later.<sup>22</sup>

For edge H-A, H-D and H-C, historical thinking provides inspiration and foundation for the other three thinking patterns. We can learn from history to explore solutions following the architectural, data-centric and computational thinking patterns. For instance, Danowitz et al developed a CPU database, which allows designers to mine micro-processor trends over the past 40 years.<sup>9</sup> In general, we can build databases and conduct corresponding data mining from the perspective of the relationship H-A, H-D and H-C, respectively.

For the connections of A-C and A-D, computational thinking and data-centric thinking can be implemented physically through architectural thinking. The dimensions in Figure 5 can be applied in either compute components or memory components. For example, the dimension of sequential vs. parallel that applies to compute components includes techniques from pipelining, superscalar organization, multithreading, and multi-core. Similarly, all levels of the cache hierarchy also support multiple concurrent data accesses by multiple banks, multiple ports and multiple channels.

For the link of C-D, computational thinking and data-centric thinking should coexist and be conducted simultaneously. When we consider how to solve a problem by computation, we also need to consider how to address the problem via data collection, mining and discovery, in addition to the consideration of data access delay and cost. Computational thinking and data-centric thinking have different focuses. This is one of the reasons why high-performance computing (HPC) and big data currently have different ecosystems and form two partially isolated communities.

For A-C, A-D, and C-D, we need to notice the interactions between compute components and the memory system, which have two distinguishing characteristics. First, regardless of whether the memory system has or does not have computing capabilities, the combined computing-memory system can be seen as a filtering process. A stream of access requests, as seen by lower-level cache, has been filtered by the higher-level caches.

Meanwhile, data access can influence computation in a way with negative feedback. When the memory system can quickly deliver data, the computing component is able to issue more data requests. However, when the number of outstanding memory requests increases beyond the memory system’s handling capacity, queue delays and bus contention will become inevitable, and therefore the memory system can no longer feed data on time. As a result, computing components will slow down (partially or completely stall) due to data starvation, and consequently will reduce data requests.

Due to the combined impact of filtering and feedbacking, CPU performance


and memory performance are entangled and mutually influenced by one another. Therefore, they need to be matched and considered in parallel. High-speed computation requires rapid data movement, and data movement is a critical factor for next-generation architectural design, either from computational thinking or from data-centric thinking perspective. As examples of the HCDA thinking paradigm, the labeled von Neumann architecture and the layered performance matching are general and have their significance in system design.

## Conclusion

Modern computers have been developed for more than seventy years. Moore's Law, which has guided the chip design for more than five decades, is approaching its end. The 25-year-old memory wall problem is becoming increasingly problematic. The landscape of computers is changing from computation-centric to data-centric. Literally, the term "computer" cannot convey the full meanings we intend it to. Therefore, to adapt to the changing landscape, we need to explore new ways of thinking and new directions for innovation.

While new computing models such as quantum computing and DNA computing under continued development, this article holds that the conventional von Neumann architecture also can be further developed for a great future. We propose a tetrahedron "historical, computational, data, and architectural" thinking paradigm, referred to as HCDA, to extend the computation thinking in the big data era. Enriching computer scientists with historical thinking, computational thinking, data-centric thinking, and architectural thinking, will boost innovation and provide corresponding actions that enhance the impact of technology on our rapidly evolving society. The four patterns of thinking and their corresponding actions<sup>41</sup> constitute an effective paradigm that can facilitate the advance of computer technologies in the new age of computing.

**Acknowledgments.** This work is supported in part by the National Natural Science Foundation of China (No. 61772497, 61672513), National Key R&D Program of China (No. 2016YFB1000201), Strategic Priority Research Program of Chinese Academy of Sciences (XDC05030100), YIPA-CAS,

BAAI, PCL, and the U.S. National Science Foundation under NSF CCF-1536079 and CNS-1730488. We thank the anonymous reviewers for their feedback and Kyle Hale for their assistance. The first author thanks Mingfa Zhu for his guidance. 

## References

- Bao, Y.-G. and Wang, S. Labeled von Neumann architecture for software-defined cloud. *J. Computer Science and Technology* 32, 2 (2017), 219–223.
- Berman, F. et al. Realizing the potential of data science. *Commun. ACM* 61, 4 (Apr.) 67–72.
- Beveridge, W. *The Art of Scientific Investigation*. Vintage Books, 1954.
- Beyer, M. Gartner says solving big data challenge involves more than just managing volumes of data; <http://www.gartner.com/it/page.jsp?id=1731916>.
- Borkar, S. Thousand core chips: a technology perspective. In *Proceedings of the 54<sup>th</sup> IEEE Annual Design Automation Conf.* (San Diego, CA), 746–749.
- Carell, T. Molecular computing: DNA as a logic operator. *Nature* 469, 7328, 45–46.
- Cong, J., Wei, P., Yu, C. H., and Zhou, P. Bandwidth optimization through on-chip memory restructuring for HLS. In *Proceedings of the 54<sup>th</sup> IEEE Annual Design Automation Conf.* (Austin, TX), 1–6.
- Dally, W.J., Turakhia, Y., and Han, S. Domain-specific hardware accelerators. *Commun. ACM* 63, 7 (July 2020), 48–57.
- Danowitz, A., Kelley, K., Mao, J., Stevenson, J.P., and Horowitz, M. CPU DB: Recording microprocessor history. *Queue* 10, 4, (2012), 10–27.
- Dechter, R. Learning while searching in constraint-satisfaction problems. In *Proceedings of the 5<sup>th</sup> National Conf. Artificial Intelligence* (Philadelphia, PA, USA, Aug. 11–15, 1986).
- Fernando, P., Kannan, S., Gavrilovska, A., and Schwan, K. Phoenix: Memory speed HPC I/O with NVM. In *Proceedings IEEE 23<sup>rd</sup> Intern. Conf. High Performance Computing* (Hyderabad, India, 2017).
- Grier, D.A. How to think from the perspective of computing? *Commun. Chinese Computing Federation* 15, 4, (2019), 36–38.
- Grier, D.A. Old and new: The computer and the brain. *Commun. Chinese Computing Federation* 13, 3 (2017), 67–68.
- Hennessy, J.L. and Patterson, D.A. A new golden age for computer architecture. *Commun. ACM* 62, 2 (2018), 48–60.
- Hill, M.D. and Marty, M.D. Amdahl's law in the multicore era. *Computer* 41, 7 (2008), 33–38.
- Hu, W., Zhang, Y., Yang, L., and Fan, B. Godson-3b1500: A 32nm 1.35ghz 40w 172.8gflops 8-core processor. In *Proceedings of Intern. Solid-State Circuits Conf. IEEE*, 54–55.
- IBM big data hub; <http://www.ibmbigdatahub.com/infographic/fourvs-big-data>.
- Jouppi, N.P. et al. In datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44<sup>th</sup> Annual Intern. Symp. Computer Architecture*, 2017, 1–12.
- Khan, S.M., Tian, Y., and Jimenez, D.A. Sampling dead block prediction for last-level caches. In *Proceedings of the 43<sup>rd</sup> Annual IEEE/ACM Intern. Symp. on Microarchitecture*. IEEE, 2010, 175–186.
- Knill, E. Physics: Quantum computing. *Nature* 463, 7280 (2010), 441–443.
- Kougkas, A., Devarajan, H., Lofstead, J., and Sun, X.-H. Labios: A distributed label-based I/O system. In *Proceedings in the 28<sup>th</sup> ACM Intern. Symp. High-Performance Parallel and Distributed Computing* (Phoenix, AZ, USA).
- Krizhevsky, A., Sutskever, I., Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems* 25, 2, 1–9.
- Kung, H. Why systolic architectures? *IEEE Computer* 15, 1 (1982), 37–46.
- Kung, H. and Leiserson, C. Algorithms for VLSI Processor Arrays. Introduction to VLSI Systems, 1980.
- Liu, Y. and Sun, X.-H. C2-bound: A Capacity and Concurrency Driven Analytical Model for Many-core Design. In *Proceedings of Intern. Conf. for High Performance Computing, Networking, Storage and Analysis*. IEEE/ACM (Austin, TX, 2015), 1–11.

- Liu, Y. and Sun, X.-H. LPM: A systematic methodology for concurrent data access pattern optimization from a matching perspective. *IEEE Trans. on Parallel and Distributed Systems* 30, 11 (2019), 1–16.
- Liu, Y. and Sun, X.-H. LPM: Concurrency-driven layered performance matching. In *Proceedings of the 44<sup>th</sup> Intern. Conf. Parallel Processing*. ACM/IEEE (Beijing, China, 2015), 879–888.
- Ma, J. et al. Supporting Differentiated Services in Computers via Programmable Architecture for Resourcing-on-Demand. In *Proceedings of the 20<sup>th</sup> Intern. Conf. Architectural Support for Programming Languages and Operating Systems* (2015), 131–143.
- Mayer-Schnberger, V. and Cukier, K. *Big Data: A Revolution that Will Transform How We Live, Work, and Think*. Houghton Mifflin Harcourt Publishing Co., Boston, New York, 2013.
- Norwegian Nobel Committee. Nobel prize in chemistry; <http://www.nobelprize.org/nobel>
- Patterson, D.A. and Ditzel, D.R. The case for the Reduced Instruction Set Compute. *ACM SIGARCH Computer Architecture News* 8, 6, (1980), 25–33.
- Qiu, J. "Ancient Times Table Hidden in Chinese Bamboo Strip. *Nature News*, Jan. 2014.
- Reed, D.A. and Dongarra, J. Exascale computing and big data. *Commun. ACM* 58, 7 (2015), 56–68.
- Reshef, D.N. et al. Detecting novel associations in large data sets. *Science* 334, 6062, (2008), 1518–1524.
- Smith, J. Decoupled access/execute computer architectures. In *Proceedings of the 9<sup>th</sup> Annual Symp. Computer Architecture* (Austin, TX, 1982), 26–29.
- Smith, S., Park, J., and Karypis, G. Sparse tensor factorization on many-core processors with high-bandwidth memory. In *Proceedings of the 2017 IEEE Intern. of Parallel and Distributed Processing Symp.* IEEE, 2017, 1058–1067.
- Sun, X.-H. and Chen, Y. Reevaluating Amdahl's Law in the multicore era. *J. Parallel and Distributed Computing* 70, 2, (2010), 183–188.
- Sun, X.-H. and Ni, L. Another view on parallel speedup. In *Proceedings of Intern. Conf. for High Performance Computing, Networking, Storage and Analysis*. IEEE/ACM, New York, (1990), 324–333.
- Sun, X.-H. and Wang, D. Concurrent average memory access time. *Computer* 47, 5 (2014), 74–80.
- Supercomputer top 500 list; <https://www.top500.org/lists/2020/06/>
- Tissenbaum, M.J., Sheldon, J., and Abelson, H. From computational thinking to computational action. *Commun. ACM* 62, 3 (Mar. 2019), 34–36.
- Von Neumann, J. *The Computer and the Brain*. Yale University Press, 1958.
- Von Neumann, J. First draft of a report on the EDVAC. *IEEE Annals of the History of Computing* 15, 4, (1993), 27–75.
- Waldrop, M. The chips are down for Moore's Law. *Nature* 530 (Feb. 2016), 144–147.
- Wing, J.M. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.
- Xu, Z. and Li, C. Low-entropy cloud computing systems. *Scientia Sinica Informationis* 47, 9, (2017), 1149–1163.

**Yuhang Liu** (liuyuhang@ict.ac.cn) is an associate professor at the State Key Laboratory of Computer Architecture, University of Chinese Academy of Sciences, and Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

**X.-H. Sun** is a distinguished professor with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA.

**Yang Wang** is a professor at the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China.

**Yungang Bao** is a professor at the State Key Laboratory of Computer Architecture, University of Chinese Academy of Sciences, and Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

© 2021 ACM 0001-0782/21/5



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

DOI:10.1145/3406011

**The evolution that serverless computing represents, the economic forces that shape it, why it could fail, and how it might fulfill its potential.**

BY JOHANN SCHLEIER-SMITH, VIKRAM SREEKANTI, ANURAG KHANDELWAL, JOAO CARREIRA, NEERAJA J. YADWADKAR, RALUCA ADA POPA, JOSEPH E. GONZALEZ, ION STOICA, AND DAVID A. PATTERSON

## What Serverless Computing Is and Should Become: The Next Phase of Cloud Computing

IN 2010, SOME OF US co-authored a *Communications* article that helped explain the relatively new phenomenon of cloud computing.<sup>4</sup> We said that cloud computing provided the illusion of infinitely scalable remote servers without charging a premium for scale, as renting 1,000 servers for one hour costs the same as renting one server for 1,000 hours, and that economies of scale for the cloud provider allowed it to be surprisingly inexpensive. We listed challenges to cloud computing, and then predicted that most would be overcome so the industry would increasingly shift from computing inside local data centers to “the cloud,” which has indeed happened. Today two-thirds of enterprise information technology spending for infrastructure and software is based in the cloud.<sup>8</sup>

We are revisiting cloud computing a decade later to explain its emerging second phase, which we believe will further accelerate the shift to the cloud. The first phase mainly simplified system administration by making it easier to configure and manage computing infrastructure, primarily through the use of virtual servers and networks carved out from massive multi-tenant data centers. This second phase hides the servers by providing programming abstractions for application builders that simplify cloud development, making cloud software easier to write. Stated briefly, the target of the first phase was system administrators and the second is programmers. This change requires cloud providers to take over many of the operational responsibilities needed to run applications well.

To emphasize the change of focus from servers to applications, this new phase has become known as *serverless computing*, although remote servers are still the invisible bedrock that powers it. In this article, we call the traditional first phase *serverful computing*.

Figure 1 shows an analogy. To attend a remote conference, you either rent a car or hail a taxicab to get from the airport to your hotel. Car rental is like serverful computing, where you must wait in line, sign a contract, reserve the car for your whole stay no

### » key insights

- The cloud originally revolutionized system administration. This second phase of cloud computing simplifies cloud programming.
- Serverless computing encompasses much more than cloud functions, or Function-as-a-Service (FaaS)—other cloud programming abstractions such as object storage also hide the complexity of servers, and more are on the way.
- Serverless today works well in limited applications, so cloud providers will create new application-specific and general-purpose serverless products to enable more use cases.
- This next phase of cloud computing will change the way programmers work as dramatically as the first phase changed how operators work.



matter how much you use it, drive the car yourself, navigate to the hotel, pay for parking, and fill it with fuel before returning it. The taxi is like serverless computing, where you simply need to give the hotel name and pay for the ride; the taxi service provides a trained driver who navigates, charges for the ride, and fills the gas tank. Taxis simplify transportation, as you don't need to know how to operate a car to get to the hotel. Moreover, taxis get higher utilization than rental cars, which lowers costs for the taxi company. Depending on the length of the conference, the cost of car rental, the cost of parking, the cost of gas, and so on, taxis are not only easier, they might even be cheaper.

In serverless computing, programmers create applications using high-level abstractions offered by the cloud provider. For example, they can define

*cloud functions*<sup>a</sup> using functional-style “stateless” programming in the language of their choice, often JavaScript or Python, then specify how the functions should run, whether in response to Web requests or to triggering events. They may also use serverless object storage, message queues, key-value store databases, mobile client data sync, and so on, a group of services offerings known collectively as *Backend-as-a-Service* (BaaS). Managed cloud function services are also called *Function-as-a-Service* (FaaS) and collectively Serverless Cloud Computing

<sup>a</sup> Different cloud platforms have different names for their offerings: Azure Functions for Microsoft Azure, Cloud Functions for Alibaba Cloud, AWS Lambda for Amazon Web Services (AWS), Google Cloud Functions and Google Cloud Run for Google Cloud Platform (GCP), IBM Cloud Functions for IBM Cloud, and Oracle Functions for Oracle Cloud.

today = FaaS + BaaS (see Figure 2).

The main innovation of serverless is hiding servers, which have an inherently complex programming and operating model. Server users must create redundancy for reliability, adjust capacity in response to changes in load, upgrade systems for security, and so on.<sup>17</sup> This often requires difficult reasoning about failure modes and performance in distributed systems. Tools can help, for example, by adjusting capacity heuristically, a form of *autoscaling*, but these too require detailed configuration and ongoing monitoring. By contrast, serverless hands these and other responsibilities to the cloud provider.

Three essential qualities of serverless computing are:

1. Providing an abstraction that hides the servers and the complexity of programming and operating them.

2. Offering a pay-as-you-go cost model instead of a reservation-based model, so there is no charge for idle resources (see Figure 3).

3. Automatic, rapid, and unlimited scaling resources up and down to match demand closely, from zero to practically infinite.

The cloud-based synthesis of *all* of these properties is substantially more

transformative than previous environments that came close to providing them.<sup>8,17</sup> Returning to our analogy, a taxi service (serverless computing) must provide a cab with a licensed driver (hide operation), charge only when giving a ride (pay as you go), and schedule enough cabs to minimize customer wait time (autoscaling). If taxis don't reliably provide all three, then custom-

ers may instead rent and operate cars (serverful computing).

A recent *Communications* article gave an excellent introduction to the current state of serverless computing, how it differs from Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS), its market share, example use cases, and its limitations.<sup>8</sup> In this article, we share our views on the evolution that serverless computing represents, the economic forces that shape it, why it could fail, and how it might evolve to fulfill its potential.

We project that the majority of data center computing will be dominated by serverless computing but we also believe that serverless computing will depart substantially from the serverless offerings of today. In particular, we believe that new general-purpose serverless abstractions will emerge, adding sophisticated state management and automatic optimization to enable many more use cases. Serverless now depends upon homogeneous CPUs, but in the future serverless will simplify use of hardware accelerators such as Graphical Processing Units (GPUs) or Tensor Processing Units (TPUs)<sup>19</sup> that support specific workloads—they offer the most likely path to higher performance as Moore's Law slows.<sup>14</sup> While there are concerns today about serverless security, we believe that a careful design could in fact make it *easier* for application developers to secure their software against external attackers.

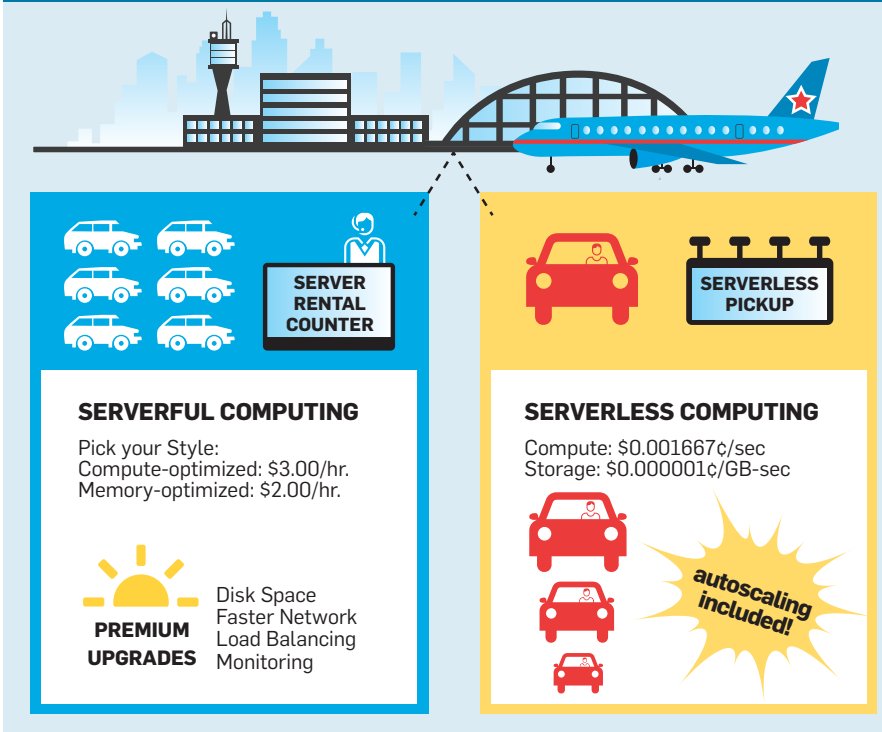
As in 2010, we once again predict that these challenges will be overcome and this second phase will become the dominant form of cloud computing, accelerating its popularity by putting the power of the cloud in the hands of *all* application developers.

### Understanding What Serverless Is Today

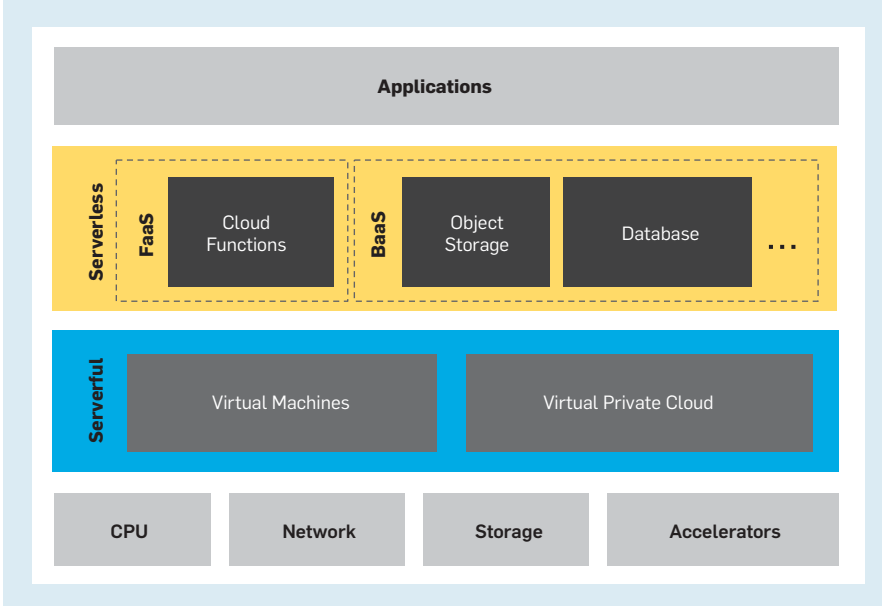
Cloud functions<sup>8</sup> capture much of the mindshare in serverless computing, but they are one of many services in the serverless cloud. The excitement around FaaS is well justified because it offers a glimpse of what general-purpose serverless computing might look like, yet BaaS services comprise a much larger, and older, set of serverless services.

For example, AWS initially offered

**Figure 1. Cloud computing approaches compared to rides from an airport: Serverful as renting a car and serverless as taking a taxi ride.**



**Figure 2. Serverless vs. Serverful cloud computing: serverless provides an abstraction between applications and the underlying servers.**





their S3 object storage as a remote backup and archival service, years before announcing EC2 virtual machine rental. You can think of S3 as a precursor to serverless computing that offered “diskless storage,” that is, providing storage but hiding the disks. Over time, cloud providers offered additional BaaS services to help serverful computing. Message queues (for example, AWS SQS, Google Cloud Pub/Sub) were another early service. Later came key-value databases (for example, Google Cloud Datastore, AWS DynamoDB, Azure CosmosDB) and SQL-based big data query engines (for example, AWS Athena, Google BigQuery).

When AWS Lambda launched in 2015 it was the first cloud functions product and offered something unique and compelling: the ability to execute nearly any code that runs on a server. It included support for several programming languages and for arbitrary libraries, all on a pay-as-you-go basis, operating securely and at any scale. However, it imposed certain limitations on the programming model that even today restrict it to certain applications. These include a maximum execution time, the lack of persistent state, and restricted networking.<sup>13</sup>

Today, several serverless environments can run arbitrary code, each catering to a particular use case. For example, Google Cloud Dataflow and AWS Glue allow programmers to execute arbitrary code as a stage in a data processing pipeline, while Google App Engine can be thought of as a serverless environment for building Web applications.

These many serverless offerings have in common the three essential qualities of serverless computing: an abstraction that hides the servers, a pay-as-you-go cost model, and excellent autoscaling. Taken together they offer a set of alternatives that may be combined to meet an ever-growing range of applications.

### Serverless Cloud Economics

Today’s cloud has been shaped as much by business considerations as by technical progress, and its future will be as well. Cloud customers choose serverless computing because it allows them to stay focused on solving problems that are unique to their domain or

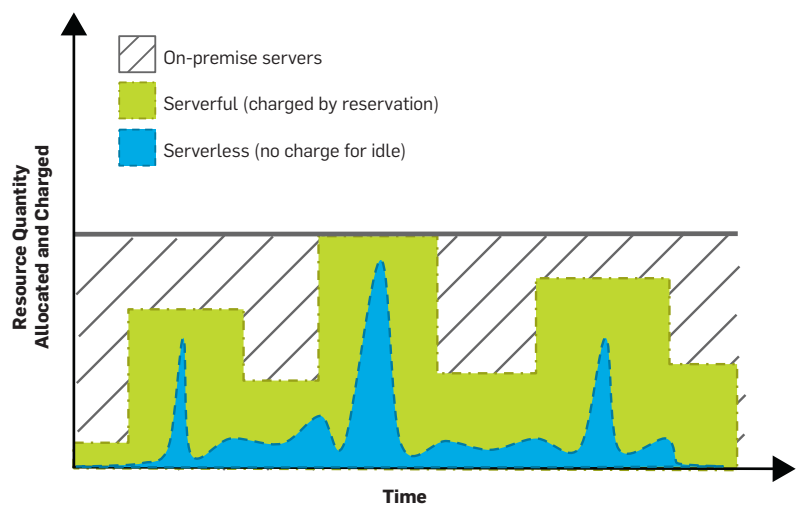
## The Cost of Serverless

If you compare the per-minute cost of running an AWS Lambda cloud function with the cost of an AWS t3.nano VM with the equivalent 0.5 GB memory, it might look like serverless computing is 7.5x as expensive. Such a comparison is misleading, however.

The beauty of serverless computing is that it provides much more than servers, yet results in cloud bills that are often much lower. Included in the price is redundancy for availability, monitoring, logging, and automated scaling, all of which need to be provided separately in a serverful context. Cost comparisons must also factor in expected utilization, since serverless users pay only while their code executes. The users of the t3.nano VM must pay for the resources reserved, whether their code is running or not. Cloud providers claim that in practice customers see cost savings of 4x-10x when moving applications to serverless.<sup>30</sup>

While serverless often saves money, for some organizations the pay-as-you-go model is at odds with the way they manage their budgets. These may be fixed in advance, often annually. Planning to use a fixed amount of server capacity may seem easier, but managing to budget is challenging in practice, especially when many teams deploy cloud VMs, or when business needs are difficult to anticipate. We believe that as organizations use serverless more, they will be able to predict their costs based on history, similar to the way they do for other pay-as-you-go services, like electricity.

**Figure 3. Serverless vs Serverful cloud computing: serverless users pay only for resources consumed, not for idle reserved capacity.**



business, rather than on problems in server administration or distributed systems.<sup>6</sup> The strength of this customer value proposition is a primary cause for optimism about the future adoption of serverless computing.

While serverless computing may appear more expensive since the unit prices of resources are higher (see sidebar “The Cost of Serverless”), customers only pay for resources that they are using, while the cloud provider bears the cost of idle resources. In practice, customers realize substantial cost savings when porting applications to serverless.<sup>30</sup> While this cost reduction could threaten cloud provider revenues, the Jevons Paradox<sup>2</sup>

suggests that low prices can spark consumption growth that more than offsets the reduction in unit costs, leading to revenue growth. Cloud providers also gain a profit opportunity by helping customers meet variable and unpredictable resource needs, something they can do more efficiently from a shared resource pool than customers can do using their own dedicated resources.<sup>16</sup> This opportunity also exists in serverful computing but grows as resources are shared on a more fine-grained basis. Serverless computing also offers cloud providers opportunities to improve their margins because BaaS products often represent product categories traditional-

**Table 1. Alternative abstraction approaches.**

Cloud functions might appear to offer a general-purpose abstraction since they run arbitrary code, however due to their limitations they work only in some applications. More sophisticated derivatives might achieve the goal of general-purpose serverless computation.

| Serverless Abstraction Approach |                         | Big Data Example                   |
|---------------------------------|-------------------------|------------------------------------|
| Application-specific            | Tool or component       | AWS Athena                         |
|                                 | Application framework   | Cloud Dataflow                     |
| General-purpose                 | Hints to implementation | Affinity hints                     |
|                                 | Automatic optimization  | Communication-minimizing placement |

ly served by high-margin software products such as databases.

The serverless pay-as-you-go model has an important positive implication for the cloud providers' incentive to innovate. Before serverless, autoscaling cloud services would automatically provision VMs, that is, reserve resources, but the customer would then pay for this capacity even if it remained idle. With serverless, the cloud provider pays for idle resources, which creates "skin in the game" on autoscaling, and provides incentives to ensure efficient resource allocation. Similarly, as the cloud provider assumes direct control over more of the application stack, including the operating system and language runtime, the serverless model encourages investments in efficiency at every level.

More productive programmers, lower costs for customers, greater profits for providers, and improved innovation all create favorable conditions for serverless adoption. However, some cloud customers have raised concerns about vendor lock-in, fearing reduced bargaining power when negotiating prices with cloud providers.<sup>16</sup> The serverful VM abstraction is standardized—mostly on account of the Linux operating system and the x86 instruction set—but each provider's serverless cloud functions and BaaS APIs differ in both readily apparent and subtle ways. The resulting switching costs benefit the largest and most established cloud providers, and give them an incentive to promote complex proprietary APIs that are resistant to de facto standardization. Simple and standardized abstractions, perhaps introduced by smaller cloud providers, open source communities, or academics, would remove the most prominent remaining economic hurdle to serverless adoption.

### The Next Phase of Cloud Computing

Perhaps the best way to understand the shift that serverless computing represents is to focus on the first of the essential qualities (as noted previously): providing an abstraction that hides servers and thus simplifies the programming and operating model. From the outset, cloud computing provided a simplified operating model, but simplified programming comes from hiding servers. The future evolution of serverless computing, and in our view of cloud computing, will be guided by efforts to provide abstractions that simplify cloud programming.

It is striking how little cloud computing has changed how programmers work to date, especially when compared to the impact it has had on operators. Much of the software that runs in the cloud is the exact same software that runs in a traditional data center. Compare the programming skills most in demand today against those needed 10 years ago and you will notice that the core skill set has changed very little, even as specific technologies come and go. By contrast, the operator's job has changed tremendously. Installing and maintaining servers, storage, and networks are largely things of the past, replaced by a focus on managing virtualized infrastructure through cloud provider APIs, and by the DevOps movement, which emphasizes the technical and organizational aspects of change management.

What makes programming the cloud hard? While it is possible to use the cloud with just one server, this offers neither fault tolerance nor scalability nor pay-as-you-go, so most cloud programming quickly becomes distributed systems programming. When writing distributed systems, programmers must reason about the data cen-

ter's spatial extent, its various partial failure modes, and all of its security threats. In the language of Fred P. Brooks, these concerns represent "accidental complexity," which arises from the implementation environment and stands in contrast to "essential complexity," which is inherent in the functionality that the application provides.<sup>7</sup> At the time of Brooks's writing, high-level languages were displacing assembly language, freeing programmers from reasoning about complex machine details such as register allocation or data layout in memory. Just as high-level languages hide many details of how a CPU operates, serverless computing hides many details of what it takes to build a reliable, scalable, and secure distributed system.

We next consider alternative approaches to serverless abstraction, including ones that exist today and ones that we imagine. These vie to answer the question, "if not servers, then what?" We group these alternative abstraction approaches into *application-specific* and *general-purpose* categories (see Table 1). Application-specific abstractions solve a particular use case, and several of them exist in products today. General-purpose abstractions must work well in a broad variety of uses and remain a research challenge.

Let us examine an illustrative example from big data processing. Consider a simple query that might arise in an e-commerce setting: computing an average over 10 billion records using weights derived from one million categories. This workload has the potential for a lot of parallelism, so it benefits from the serverless illusion of infinite resources.

We present two application-specific serverless offerings that cater to this example and illustrate how the category affords multiple approaches. One could use the AWS Athena big data query engine, a tool programmed using SQL (Structured Query Language), to execute queries against data in object storage. SQL is particularly well suited to analytics and can express this computation with a single statement. Alternatively, one could use a framework such as that which Google Cloud Dataflow provides. Doing so requires writing a simple MapReduce-style<sup>11</sup> program, for example, using Java or Python, with two func-

tions: one that computes a weighted average for some chunk of data, and another that combines weighted averages for separate chunks into one for their union. The framework takes care of piping data in and out of these functions, as well as autoscaling, reliability, and other distributed systems concerns. In contrast to the SQL-based tool, this abstraction can run arbitrary code, which can make it suitable to a wider range of analytics problems.

General-purpose serverless abstractions that offer a performant solution to our big data example do not yet exist. Cloud functions might appear to provide a solution since they allow users to write arbitrary code, and for some workloads they do,<sup>28</sup> but due to limitations they sometimes perform much worse than alternatives.<sup>13,17</sup> Figure 4 illustrates how network traffic could be much higher if we implement our example using cloud functions, rather than using an application-specific framework such as Cloud Dataflow. With cloud functions, the provider distributes work across various VM instances without regard to the application's communication patterns, which simplifies autoscaling but increases network traffic.

We suggest two paths to enhancing cloud functions so that they work well in a broader range of applications, potentially turning them into general-purpose serverless abstractions. First, we imagine that hints provided by the programmer might indicate how to achieve better performance. Hints might describe application communication patterns (for example, broadcast or all-reduce), or suggest task placement affinity.<sup>25</sup> Such an approach has precedent in compilers (for example, branch prediction, alignment, and prefetching hints).

Second, and more compellingly, we envision inefficiencies being removed by automatic optimization. In our example the cloud provider might promise to infer locality optimizations from observed communication patterns. In some cases, such inferences might also be made statically, based on an analysis of the program. In the single-machine context this has ample precedent in what modern compilers and language runtimes do, and one might think of this form of serverless com-

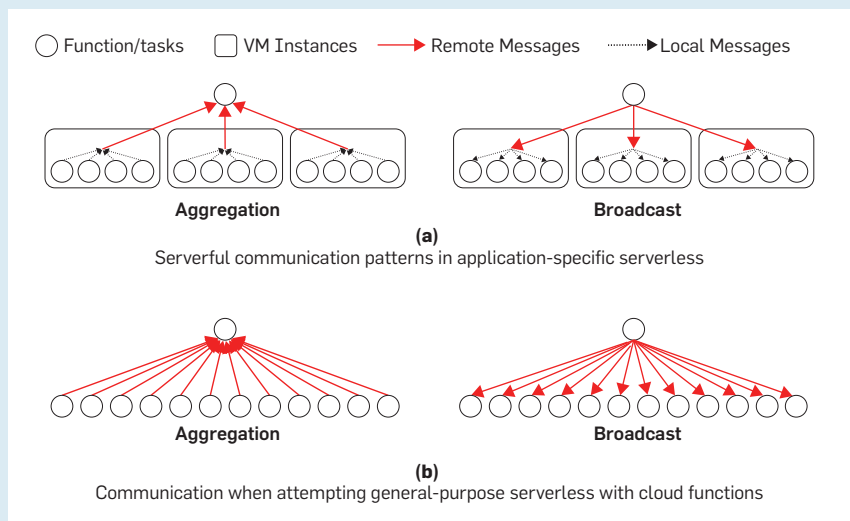
puting as extending language support to distributed systems.

Figure 5 illustrates the difference between application-specific and general-purpose serverless abstractions. In the general-purpose case the cloud provider exposes a few basic building blocks, for example, an enhanced version of cloud functions and serverless storage of some sort. A variety of application-specific use cases can be built on top of these foundations. With application-specific serverless, cloud providers instead offer a proliferation of BaaS to meet the needs of an ever-greater number of applications.

Today, serverless computing remains entirely of the application-specific variety. Even cloud functions, which can execute arbitrary code, are popular mainly for stateless API serving and event-driven data processing.<sup>27</sup> We expect application-specific serverless computing to grow, but we are most excited about the potential emergence of general-purpose serverless abstractions, which could host software ecosystems catering to every need. In our view, only the general-purpose approach can ultimately displace servers to become the default form of cloud programming. However, general-pur-

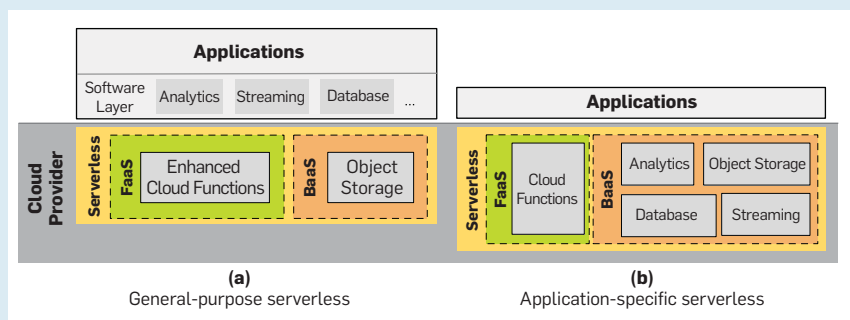
**Figure 4. Increased communication for aggregation and broadcast patterns.**

Application-specific serverless frameworks (for example, Cloud Dataflow) can be implemented with serverful communication patterns. In this case (a) the fewer arrows indicate less network communication than in (b) the general-purpose serverless option. By packing  $K$  tasks per VM instance, an application-specific serverless solution, like a serverful solution, is able to achieve a communication complexity of  $O(N/K)$  for a job with  $N$  tasks, as opposed to  $O(N)$  for the cloud function based alternative which can not influence task placement. Typical values for  $K$  range from 10 to 100, leading to an overall difference of one to two orders of magnitude.



**Figure 5. Potential future directions for serverless.**

(a) General-purpose serverless abstractions support a wide range of needs, with application-specific functionality provided by software above the cloud provider interface, (b) application-specific serverless abstractions with many BaaS point-solutions.



pose serverless technology does not exist today, and developing it presents research challenges.

### Research Challenges

Serverless computing is evolving rapidly and offers various research challenges, many of them common to both application-specific and general-purpose serverless.

**State management.** Distributed cloud applications often need to exchange short-lived or ephemeral state between their component tasks. Examples include application-wide caches, indexes, and other lookup tables, or intermediate results of big data analytics. Cloud functions today allow applications to store ephemeral state locally at each function, which is useful for caching and as working memory for the program. Serverless shared state may be saved in object storage or key-value stores, but these do not simultaneously provide low latency, low cost, high throughput, and fine-grained access, as is possible with servers.<sup>17</sup> Approaches to addressing these challenges include temporary data storage for analytics<sup>21</sup> as well as stateful cloud functions that integrate caching and provide consistency guarantees.<sup>29</sup>

**Networking.** Cloud functions transfer the responsibility of scheduling work from the user to the cloud provider, which has several interesting consequences. Since users cede control over when functions run, passing state between cloud functions requires a trip through shared storage; direct network communication makes little sense and cloud providers block it. Accessing shared storage adds significant latency, sometimes hundreds of milliseconds. Users also cede control over where func-

tions run, thus precluding optimizations common with servers, including sharing common inputs between tasks and combining outputs before sending them over the network (see Figure 4 and previous discussion). Attempts to overcome these challenges will highlight the tension between giving programmers more control and allowing the cloud provider to make optimizations automatically.

**Predictable performance.** Both FaaS and BaaS can exhibit variable performance which precludes their use in applications that must meet strict guarantees. Part of the reason for this is fundamental: serverless providers rely on statistical multiplexing to create the illusion of infinite resources, while denying users control over resource oversubscription. There is always some chance that unfortunate timing will create queuing delays. There is also a latency cost to re-assigning resources from one customer to another, which in the cloud function context is known as a “cold start.” Cold start latency has several components,<sup>17</sup> and significant among them is the time it takes to initialize the software environment of the function. There has already been progress in this area. Cloud function environments such as Google gVisor and AWS Firecracker<sup>1</sup> can now start in about 100 ms, whereas traditional VMs take tens of seconds to boot. It is also possible to accelerate application-level initialization such as loading libraries.<sup>26</sup> There is probably still much room for improvement in these areas, though there is also evidence that performance optimization and isolation for security are fundamentally at odds.<sup>24</sup> Customers of AWS

Lambda can also avoid cold start latencies by purchasing “provisioned concurrency,” which controversially reintroduces a form of resource reservation to the serverless model. We hope to also see pricing based on statistical guarantees, or Service Level Objectives (SLOs), which are absent in serverless today.

**Security.** Serverless computing leads to fine-grained resource sharing and so increases the exposure to *side-channel attacks*, whereby attackers exploit subtle behaviors of real hardware that differ from either specifications or programmer assumptions (see sidebar “Serverless and Security”). Threats range from Rowhammer attacks on DRAM<sup>20</sup> to those exploiting microarchitectural vulnerabilities.<sup>22</sup> In addition to adopting mitigations developed for serverful computing, serverless might employ randomized scheduling to make it more difficult for an attacker to target a specific victim. Serverless computing also can incur greater information leakage through network communication because of the fine-grained decomposition of an application and physical distribution of its pieces. An attacker observing the size and timing of network traffic, even if it is encrypted, might make inferences about private data. Addressing these risks may be possible through oblivious computing.<sup>12</sup>

**Programming languages.** Simplified distributed systems programming is a core benefit of serverless computing,<sup>18</sup> and while much previous work in this area is relevant, the serverless setting calls for a new perspective and adds urgency. Traditional challenges include fault tolerance, consistency, concurrency, and the performance and efficiency that comes from locality. New challenges include first-class support for autoscaling, pay-as-you-go, and fine-grained multiplexing.

Fault tolerance concerns are elevated by attempts to extend serverless computing beyond stateless cloud functions. Azure Durable Functions uses C# language features to provide transparent checkpointing, which makes it easier to write stateful and resumable serverless tasks. Microsoft Orleans,<sup>5</sup> which implements an *actor model*,<sup>15</sup> similarly hides fault tolerance concerns from programmers. Actors


## Serverless and Security

Today, serverless computing merely shifts some security responsibilities from the cloud customer to the cloud provider, just as it shifts other system administration responsibilities. With cloud functions, security updates to operating systems, language runtimes, and standard software packages are applied without customer involvement, usually quickly and reliably. For BaaS services, the cloud provider assumes responsibility for securing everything behind an API. This path may prove to be an important advantage because it allows developers to reason about security at a higher abstraction level. They do not need to implement lower-level security mechanisms, which could lead to fewer security mistakes. While this benefit must be weighed against the exposure to attacks through shared hardware, we believe that improved abstractions may eventually make application security easier to achieve with serverless computing.


also provide a notion of locality, and could be a counterpart to cloud functions for stateful serverless computing. Ray<sup>25</sup> embodies elements of both. Approaches to consistency include language-integrated transactions, pioneered by Argus.<sup>23</sup> However, transactions are fraught with performance and scalability challenges, which an autoscaling serverless environment may exacerbate. An alternative approach lies in languages like Bloom,<sup>3</sup> which allows automated analysis to determine which parts of a program can run independently, without coordination, and thus scalably. Pay-as-you-go should encourage language developers to rethink resource management, for example, automated garbage collection might be adapted to metered memory pricing. Language approaches to cloud programming,<sup>9</sup> which address the complexity of distributed systems programming head-on, may represent the most direct and ambitious approach to simplifying cloud programming.

**Machine learning.** We believe that automatic optimization with machine learning will play an important role in all of the areas discussed above. It may help decide where to run code, where to keep state, when to start up a new execution environment, and how to keep utilization high and costs low while meeting performance objectives. It may also aid in identifying malicious activity that threatens security, or in automatically cutting up large programs into pieces that can execute in separate cloud functions. Machine learning can help optimize serverful computing too,<sup>10</sup> but serverless abstractions give cloud providers more control over the relevant knobs, as well as the visibility across many customers required to train robust and effective models.

**Hardware.** Current trends in hardware may be complementary to serverless computing. The x86 microprocessors that dominate the cloud are barely improving in performance; in 2017, program latency improved only 3%,<sup>14</sup> a trend that if continued implies that performance won't double for 20 years. Similarly, the ending of Moore's Law is slowing the growth of per-chip DRAM capacity. The industry response has been the introduction of Domain Spe-



**It is striking how little cloud computing has changed how programmers work to date, especially when compared to the impact it has had on operators.**



cific Architectures (DSAs), which are tailored to a specific type of problem, offering significant performance and efficiency gains, but performing poorly for other applications.<sup>14</sup> GPUs have long been used to accelerate graphics, and we are starting to see DSAs for ML such as TPUs. GPUs and TPUs can outperform CPUs for narrow tasks by factors of 30x.<sup>19</sup> These examples are the first of many, as general-purpose processors enhanced with DSAs will likely become the norm.

We believe serverless computing may provide a useful programming model for integrating diverse architectures, say with separate cloud functions running on separate accelerators. It also helps create room for innovation by raising the level of abstraction, for example, by allowing a cloud provider to substitute a DSA for a CPU when recognizing a workload that could benefit.

### **Why Serverless Computing Might Still Fail**

While we believe serverless computing can grow to become the cloud programming default, we can also imagine several scenarios in which serverful computing retains its dominance. First, serverful computing is a moving target, one that improves relentlessly, if slowly. Cloud VMs that once were billed by the hour now have a minimum billing increment of one minute, and charge by the second thereafter. Container and VM orchestration tools (for example, Kubernetes, Terraform) help streamline complex deployments, and increasingly automate administrative tasks such as taking backups. Programmers can rely on mature software ecosystems and strong legacy compatibility when building applications, while companies already have teams skilled in serverful cloud deployments. Server hardware also keeps getting bigger and more powerful, bringing CPU, memory, and accelerator power together in a closely coupled environment, a benefit for some applications.

Second, today's successful serverless products fall into the application-specific category and are narrowly targeted, whereas general-purpose serverless abstractions have a better chance of displacing serverful comput-

ing (which is also general-purpose). However general-purpose serverless computing faces hurdles: the technology that we envision does not exist yet, and it may be a less lucrative business for cloud providers.

Finally, even if our vision plays out, the brand of “serverless computing” might not survive. The temptation to label older products as the next new thing is strong and can create confusion in the marketplace. We have been happy to see products such as Google App Engine pick up the serverless moniker, and along with it features such as scaling to zero. However, if the term becomes diluted by half-hearted efforts, then perhaps general-purpose serverless computing will emerge under another name.

**Conclusion and Predictions**

Cloud computing is both flourishing and evolving. It has overcome the challenges that faced it in 2010, as we projected.<sup>4</sup> Offering lower costs and simplified system administration, the business is growing up to 50% annually and proving highly profitable for cloud providers. Cloud computing is now entering a second phase in which its continued growth will be driven by a new value proposition: simplified cloud programming.

Analogous to how hailing a taxi simplifies transportation over renting a car (see Figure 1), serverless computing relieves programmers from thinking about servers and everything complicated that goes along with them. Following the same naming convention, you could classify a taxi service as *carless transportation* in that the passenger need not know how to operate a car to get a ride. Serverless raises the level of abstraction of the cloud, adopts pay-as-you-go pricing, and rapidly auto-scales down to zero and up to practically infinite resources.

Serverless computing is still evolving, and many open questions remain, both in defining its abstractions and in implementing them. We (boldly) conclude this paper with five predictions for serverless computing in the next decade:

1. Today’s FaaS and BaaS categories will give way to a broader range of abstractions, which we categorize as either *general-purpose serverless computing* or

*application-specific serverless computing*. While serverful cloud computing won’t disappear, its relative use in the cloud will decline as serverless computing overcomes its current limitations.

2. We expect new general-purpose serverless abstractions to support just about any use case. They will support state management, as well as optimizations—either user-suggested or automatically inferred—to achieve efficiencies comparable or maybe better than those of serverful computing.

3. We see no fundamental reason for the cost of serverless computing to exceed that of serverful computing. We predict that as serverless evolves and increases in popularity almost any application, be it tiny or massive-scale, costs no more—and perhaps a lot less—with serverless computing

4. Machine learning will play a critical role in serverless implementations, allowing cloud providers to optimize execution of large-scale distributed systems while providing a simple programming interface.

5. Computer hardware for serverless computing will be much more heterogeneous than the conventional x86 servers that powers it today.

If these predictions hold, serverless computing will become the default computing paradigm of the Cloud Era, largely replacing serverful computing and thereby closing the Client-Server Era, just as the smartphone brought the end of the PC Era.

**Acknowledgments.** We thank the reviewers for their thoughtful comments, as well as the many friends who gave feedback on early drafts. This work was conducted at UC Berkeley RISELab and it was supported by a National Science Foundation Expedition Project, Alibaba Group, Amazon Web Services, Ant Financial, Ericsson, Facebook, Futurewei, Google, Intel, Microsoft, Scotiabank, Splunk, and VMware. C

**References**

1. Agache, A., et al. Firecracker: Lightweight virtualization for serverless applications. In *Proceedings of the 17th USENIX Symp. Networked Systems Design and Implementation* (2020), 419–434.
2. Alcott, B. Jevons’ paradox. *Ecological Economics* 54, 1 (2005), 9–21.
3. Alvaro, P., et al. Consistency analysis in Bloom: A CALM and collected approach. *CIDR*, 249–260.
4. Armburst, M., et al. A view of cloud computing. *Commun. ACM* 53, 4 (Apr. 2010) 50–58.
5. Bernstein, P., et al. Orleans: Distributed virtual actors for programmability and scalability. MSR-TR-2014–41, 2014.
6. Brazeal, F. The business case for serverless, 2018;

- <https://www.trek10.com/blog/business-case-for-serverless>
7. Brooks, F. No silver bullet: essence and accidents of software engineering. In *Information Processing*. IEEE, 1986.
8. Castro, P., et al. The rise of serverless computing. *Commun. ACM* 66, 12 (Dec. 2019), 44–54.
9. Cheung, A., Crooks, N., Milano, M., and Hellerstein, J. New directions in cloud programming. *CIDR*, 2021.
10. Dean, J. Machine learning for systems and systems for machine learning. In *Proceedings of the 2017 Conf. Neural Info. Processing System*.
11. Dean, J. and Ghemawat, S. MapReduce: simplified data processing on large clusters. *Commun. ACM* 51, 1 (Jan. 2008), 107–113.
12. Goldreich, O. Towards a theory of software protection and simulation by oblivious RAMs. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, (1987) 182–194.
13. Hellerstein, J., et al. Serverless computing: One step forward, two steps back. *CIDR*, 2019.
14. Hennessy, J. and Patterson, D. A new golden age for computer architecture. *Commun. ACM* 62, 2 (Feb. 2019), 48–60.
15. Hewitt, C., Bishop, P., and Steiger, R. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd Intern. Joint Conf. Artificial Intelligence*, (1973), 235–245. Morgan Kaufmann Publishers Inc.
16. Irwin, D. and Urgaonkar, B. Research Challenges at the Intersection of Cloud Computing and Economics. National Science Foundation, 2018.
17. Jonas, E. et al. Cloud programming simplified: A Berkeley view on serverless computing. Tech. Rep. No. UCB/ECS-2019-3, 2019.
18. Jonas, E., Pu, Q., Venkataraman, S., Stoica, I., and Recht, B. Occupy the cloud: Distributed computing for the 99%. In *Proceedings of the ACM SoCC*, 2017.
19. Jouppi, N. et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual Intern. Symp. Computer Architecture*. (2017), 1–12.
20. Kim, Y., et al. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *Proceeding of the 42nd ISCA*. IEEE Press, 2014, 361–372.
21. Klimovic, A., et al. Pocket: Elastic ephemeral storage for serverless analytics. In *Proceedings of the 13th USENIX Symp. Operating Systems Design and Implementation* (2018), 427–444.
22. Kocher, P., et al. Spectre attacks: Exploiting speculative execution. *Commun. ACM* 63, 7 (July 2020), 93–101.
23. Liskov, B. Distributed programming in Argus. *Commun. ACM* 31, 3 (Mar. 1988), 300–312.
24. McIlroy, R., Sevcik, J., Tebbi, T., Titzer, B., and Verwaest, T. Spectre is here to stay: An analysis of side-channels and speculative execution. 2019; arXiv:1902.05178.
25. Moritz, P., et al. Ray: A distributed framework for emerging AI applications. In *Proceedings of the 13th USENIX Symp. Operating Systems Design and Implementation* (2018), 561–577.
26. Oakes, E., et al. SOCK: Rapid task provisioning with serverless-optimized containers. In *2018 USENIX Annual Technical Conf.* (2018), 57–70.
27. Passwater, A. 2018 serverless community survey: Huge growth in serverless usage; <https://serverless.com/blog/2018-serverless-community-survey-huge-growth-usage/>
28. Perron, M., Fernandez, R., DeWitt, D., and Madden, S. Starling: A scalable query engine on cloud functions. In *Proceedings of the 2020 ACM SIGMOD Intern. Conf. Management of Data* (2020), 131–141.
29. Sreekanti, V., et al. Cloudburst: Stateful functions-as-a-service. *Proc. VLDB* 13, 11 (2020), 2438–2452.
30. Wagner, T. Debunking serverless myths, 2018; <https://www.slideshare.net/TimWagner/serverlessconf-2018-keynote-debunking-serverless-myths>

**Johann Schleier-Smith, Vikram Sreekanti, Anurag Khandelwal, Joao Carreira, Neeraja J. Yadwadkar, Raluca Ada Popa, Joseph E. Gonzalez, Ion Stoica, and David A. Patterson.**

The authors are associated with the UC Berkeley RISELab (Real-Time Intelligence Secure Explainable Systems Lab).

Copyright held by authors/owners.



WebSci'21

# 13th ACM WEB SCIENCE CONFERENCE 2021

## June 21st - June 25th

Hosted by the University of Southampton, UK - delivered online



Join world leaders in Web Science research, technology, industry and policy-making at this global, interdisciplinary event to explore how Web Science can help us address the challenges and opportunities we face from the effects of the pandemic and other global threats.

Keynote Speakers include:

- Deen Freelon, Associate Professor at UNC Chapel Hill
- Baroness Martha Lane Fox, Entrepreneur and internet activist
- Matthew Weber, Associate Professor at Rutgers University
- Daniel J. Weitzner, Director of the MIT Internet Policy Research Initiative
- Jennifer Zhu Scott, Forbes World's Top 50 Women in Tech in 2018

*You can also take part in:*

*Workshops • Tutorials • Spotlight Panels • Paper Sessions • PhD Symposium • Brave Conversations • Meet the Authors*



Registration is £50 to attend all 5 days

UNIVERSITY OF  
Southampton

# [websci21.webscience.org/register](http://websci21.webscience.org/register)

## Symbolic automata better balances how automata are implemented in practice.

BY LORIS D'ANTONI AND MARGUS VEANES

# Automata Modulo Theories

FINITE AUTOMATA ARE ONE of the most fundamental models of computation and are taught in almost all undergraduate computer-science curricula. Although automata are typically presented as a theoretical model of computation, they have found their place in a variety of practical applications, such as natural language processing, networking, program verification, and regular-expression matching. In this article, we argue that the classical definition of finite automata is not well suited for practical implementation and present symbolic automata, a model that addresses this limitation.

In most of the literature, finite automata are intuitively presented as follows. A finite automaton is a graph that describes a set of strings over a finite alphabet  $\Sigma$ . In the graph, nodes are called states, some states are initial, some states are final, and each edge “reads” a symbol from  $\Sigma$ . Every path from an initial to a final state describes a string accepted by the automaton. For example, the automaton in Figure 1(a), given  $\Sigma = \{0, 1\}$ , describes

the set of all binary strings starting with 0. Researchers have designed a number of decision procedures and extensions for this simple but very powerful model.

As discussed in textbooks, example Hopcroft and Ullman,<sup>22</sup> the aforementioned way of defining automata is simple and easy to implement.

*Any automaton can be represented by a set of edges, where an edge  $(p, a, q)$  goes from state  $p$  to  $q$  reading symbol  $a$ .*

For certain applications, such as natural language processing, this type of implementation is appropriate. However, in many applications (particularly in software verification), the alphabet over which the automaton has to operate is so big (sometimes infinite) that an explicit representation of all its symbols is infeasible.

The following two applications of finite automata are examples of this problem.

**Regular expressions.** Automata are used in most implementations of regular-expression matching and analysis algorithms. In this domain, the alphabet  $\Sigma_r$  is typically very large, ranging between

### » key insights

- **Gap:** Real implementation of finite automata use the structure of the alphabet, for example, character classes, to enable efficient data structures and algorithms. These aspects are missing in the way automata are taught and defined in the literature.
- **Innovation:** Symbolic automata match how automata are implemented in practice by representing the alphabet and automaton structures separately. This separation yields a more general model that can handle complex and infinite alphabets while retaining all the desirable properties of finite automata and enabling new practical applications.
- **Opportunity:** Separating the alphabet and automaton representations opens opportunities for redesigning the ways in which we teach automata and design algorithms for them.





hundreds and millions of characters, for example, extended ASCII has 256 characters and Unicode has over one million characters. A quick Web search of finite-automata implementations for regular expressions shows that in all the performing implementations, transitions (that is, the edges of the graph) *do not* “read” a single character in  $\Sigma_R$  as done in the classical definition.<sup>a</sup> Instead, practical implementations allow transitions to carry (that is, “read”) *sets of characters*, that is, an edge in the graph denotes a transition of the form  $(p, S, q)$  where  $S \subseteq \Sigma_R$ , see Figure 1(b). Moreover, in most implementations,  $S$  is represented using a dedicated data structure, for example, an interval  $[‘0’, ‘9’]$  representing the set of all ASCII decimal digits  $\{‘0’, ‘1’, \dots, ‘9’\}$ , see Figure 1(c). Use of intervals assumes that all characters have a numeric code that implies a total order over  $\Sigma_R$ —for example, the  $i$ th decimal ASCII digit has code  $48 + i$ .

**Model checking.** Automata are the backbone of many model checking

algorithms,<sup>35</sup> where these models are used to describe properties a system must obey. In model checking, the alphabet is typically defined with respect to a finite set  $AP_k = \{b_i\}_{i=0}^{k-1}$  of Boolean properties of interest, often called *atomic propositions*. The alphabet is  $\Sigma_{MC} = 2^{AP_k}$ , where a character  $\alpha \in \Sigma_{MC}$  represents the assignment  $\{b = true\}_{b \in \alpha} \cup \{b = false\}_{b \in AP \setminus \alpha}$ . In practical implementations of automata for model checking, transitions do not read individual characters and instead read *sets of characters*.<sup>18</sup> The tools assume a particular representation of the sets, for example as binary decision diagrams, see Figure 1(d), or as Boolean formulas  $\varphi$  over the variables  $AP$  where each satisfying assignment defines a character  $\alpha$ , denoted by  $\alpha \models \varphi$ . For example, given  $k = 7$ , the predicate  $\varphi = b_6 \wedge b_5 \wedge b_4 \wedge (b_3 \vee (\overline{b_2} \wedge \overline{b_1}))$  in Figure 1(f) has exactly 10 satisfying assignments, such as  $\{b_0, b_4, b_5\} \models \varphi$ .

The above applications clarify the following point.

*Practical implementations of automata allow transitions to carry sets of characters instead of individual characters and take advantage of the structure of the input alphabet.*

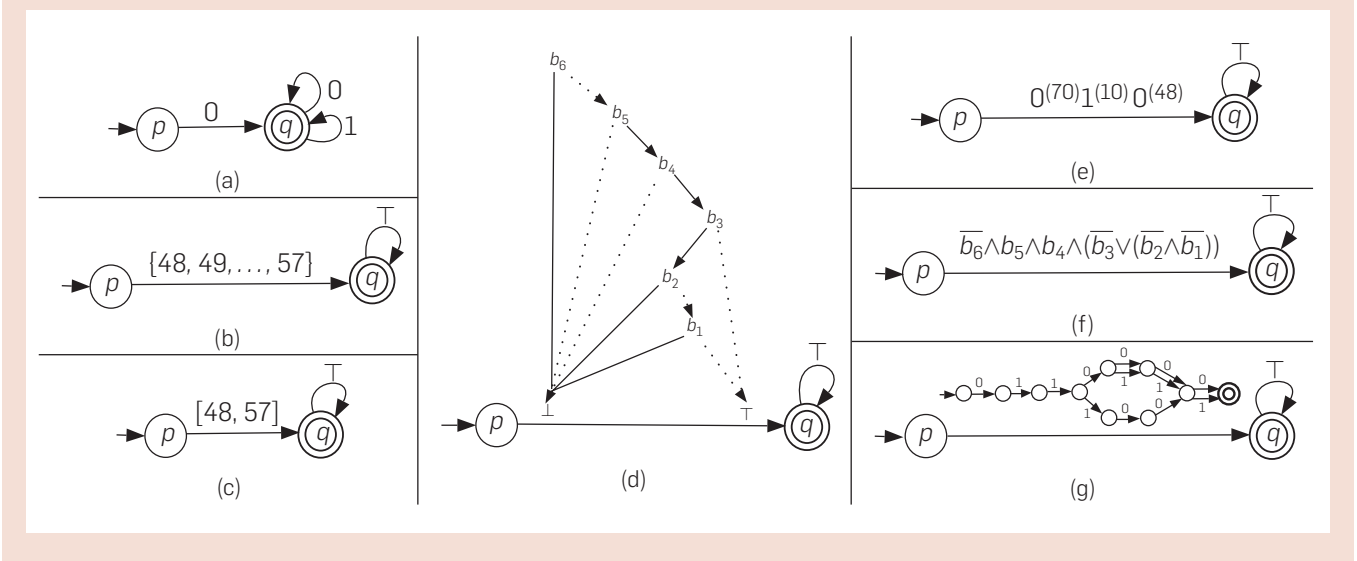
These examples show that the classical theory of automata does not do a good job in capturing the ways in which automata are implemented in practice. Therefore, all the theoretical results—for example, decision procedures and closure properties—do not directly apply to such implementations and cannot “take advantage” of the structure present in how automata are implemented. *Symbolic automata* and their variants (the models presented in this article) address this problem.

*Symbolic automata explicitly model how the alphabet is represented/implemented in practical applications and allow us to design automata decision procedures that take advantage of the alphabet representation.*

Symbolic automata extend finite automata by allowing transitions to carry predicates over rich alphabet theories (for example, intervals or binary decision diagrams). Unlike what is done in existing implementations of automata, where the alphabet representation is chosen a priori and hard-coded in the model, symbolic automata explicitly separate the representation of the alphabet structure from one of the finite state graph structures. Concretely, symbolic automata provide a unifying approach for different alphabet

a regex-automata (<https://github.com/BurntSushi/regexautomata/blob/master/src/nfa.rs>), Brics library (<https://www.brics.dk/automaton/>), Dregex - Deterministic Regular Expression Engine (<https://github.com/marianobarrios/dregex/blob/master/src/main/scala/dregex/impl/Dfa.scala>).

**Figure 1.** (a) classical automaton over the alphabet  $\{0, 1\}$  accepting all strings that start with 0; (b–g) symbolic automata accepting all ASCII strings starting with a decimal digit with predicates represented by: (b) hashsets; (c) intervals; (d) binary decision diagrams where dashed arrow is case  $b_i = 0$ ; (e) bitvectors; (f) Boolean formulas; and (g) finite automata over the alphabet  $\{0, 1\}$  restricted to  $\{0, 1\}^{(7)}$ .



representations by representing the alphabet using an effective Boolean algebra  $\mathcal{A}$ , also called the alphabet theory. The automata algorithms are now designed *modulo*  $\mathcal{A}$ . This separation of concerns allows one to seamlessly change the alphabet representation (for example, choosing any of the representations in Figure 1), without changing any of the underlying automata algorithms (that is, each decision procedure is implemented in a way that is agnostic to the choice of the alphabet theory).

Not only symbolic automata better reflect how automata are implemented in practice, but they also allow one to represent strings over infinite alphabets, for example, the set of rational numbers. Despite this increase in expressiveness, symbolic automata enjoy many of the desirable closure and decidability properties of finite automata and have been in fact used in a variety of applications: verification of functional programs operating over lists and trees,<sup>13</sup> analysis of complex implementations of BASE64 and UTF encoders,<sup>10</sup> automatic synthesis of inverses of complex string-manipulating programs,<sup>23</sup> analysis of programs involving regular expressions over large alphabets,<sup>8,9,36</sup> automatic parallelization of list-processing code,<sup>31</sup> solving constraints over sequences,<sup>33</sup> vulnerability detection in Web-applications,<sup>4</sup> analysis of program binaries,<sup>5</sup> and fast execution of string transformations.<sup>1,28</sup>

The goal of this article is to give an overview of what is currently known about symbolic automata and their variants, and what applications these models have enabled. The concept of automata with predicates instead of concrete symbols was first mentioned in Watson.<sup>39</sup> This article focuses on work done following the definition of symbolic finite automata presented in Veanes et al.,<sup>36</sup> where predicates have to be drawn from a decidable Boolean algebra. The term symbolic automata is sometimes used to refer to automata over finite alphabets where the state space is represented using binary decision diagrams. This meaning is different from the one described in this article.

It is hard to describe all the works related to symbolic automata in one

## Symbolic automata extend finite automata by allowing transitions to carry predicates over rich alphabet theories.

article, and the authors curate an updated list of papers on symbolic automata and transducers (<http://pages.cs.wisc.edu/~loris/symbolic-automata.html>). Moreover, the algorithms discussed in this article are implemented in the open source libraries AutomataDotNet (C#, <https://github.com/AutomataDotNet/>) and symbolic automata (Java, <https://github.com/lorisdanto/symbolicautomata/>).

### Symbolic Automata

**Structured alphabets and Boolean algebras.** As we illustrated in Figure 1, in symbolic automata, transitions carry predicates, and to formally describe predicates, we need to define Boolean algebras. Formally, an *effective Boolean algebra*  $\mathcal{A}$  is a tuple  $(\mathcal{D}, \Psi, \llbracket \_ \rrbracket, \perp, \top, \vee, \wedge, \neg)$ , where  $\mathcal{D}$  is a set of *domain elements*;  $\Psi$  is a set of *predicates* closed under the Boolean connectives, with  $\perp, \top \in \Psi$ ; and the component  $\llbracket \_ \rrbracket: \Psi \rightarrow 2^{\mathcal{D}}$  is a *denotation function* such that (i)  $\llbracket \perp \rrbracket = \emptyset$ , (ii)  $\llbracket \top \rrbracket = \mathcal{D}$ , and (iii) for all  $\varphi, \psi \in \Psi$ ,  $\llbracket \varphi \vee \psi \rrbracket = \llbracket \varphi \rrbracket \cup \llbracket \psi \rrbracket$ ,  $\llbracket \varphi \wedge \psi \rrbracket = \llbracket \varphi \rrbracket \cap \llbracket \psi \rrbracket$ , and  $\llbracket \neg \varphi \rrbracket = \mathcal{D} \setminus \llbracket \varphi \rrbracket$ . We also require that checking *satisfiability* of  $\varphi$ —that is, whether  $\llbracket \varphi \rrbracket \neq \emptyset$ —is *decidable*.  $\mathcal{A}$  is *extensional* when  $\llbracket \varphi \rrbracket = \llbracket \psi \rrbracket$  implies that  $\varphi = \psi$ —that is, semantic equivalence coincides with syntactic equivalence.

The following example shows how some of the character representations described in Figure 1 are Boolean algebras.

*Example 2.1 (ASCII Algebras).* Regular expressions in modern programming languages use *character classes* as basic building blocks. Character classes can be unioned and complemented. For example, the character classes  $[A-Z]$  and  $[a-z]$  denote the set of all ASCII upper and lower case letters, respectively,  $[A-Za-z]$  denotes their union, and  $[^A-Za-z]$  denotes the complement of  $[A-Za-z]$ . A natural choice for domain  $\mathcal{D}$  is the set of all ASCII codes  $\{n \mid 0 \leq n \leq 127\}$ . Figure 1(b–g) shows several different options for representing  $\Psi$ . Consider the following representations of  $\varphi \in \Psi$ :

- (a)  $\varphi$  is a binary decision diagram (BDD) over  $AP_7$  (Figure 1(d));  $\wedge$  is AND-product of BDDs; and  $\perp$  is the false-leaf BDD;
- (b)  $\varphi$  is a 128-bit bitvector whose  $n$ th bit is 1 iff  $n \in \llbracket \varphi \rrbracket$  (Figure 1(e));  $\wedge$

- is bitwise-AND; and  $\perp$  is  $0^{(128)}$ —that is, a sequence of 128 zeroes;
- (c)  $\varphi$  is a Boolean formula over  $AP_7$  (Figure 1(f));  $\wedge$  is syntactic conjunction; and  $\top$  is the true predicate.
- (d)  $\varphi$  is an automaton over  $\Sigma = \{0, 1\}$  restricted to  $\Sigma^{(7)}$  (Figure 1(g));  $\wedge$  is product of automata; and  $\top$  is the fixed automaton accepting all strings in  $\Sigma^{(7)}$ .

We identify the underlying semantic domain  $\mathcal{D}$  in this example with character codes. This reflects the fact that implementations that use ASCII often use expressions such as `A` and `\x41` as equivalent notations of the same underlying character (code). In other words,  $\llbracket \varphi \rrbracket \subseteq \mathcal{D}$  is a set of natural numbers, namely the set of all character codes denoted by  $\varphi$ . In particular, in (a) and (c),  $\llbracket \varphi \rrbracket = \{\sum_{b_i \in \alpha} 2^i \mid \alpha \models \varphi\}$ , and in (d),  $\llbracket \varphi \rrbracket = \{\sum_{1 \leq i \leq 7, v_i = 1} 2^{7-i} \mid v \in \mathcal{L}(\varphi)\}$ . Observe that the algebras in (a) and (b) are extensional where satisfiability is trivial (nonequality to  $\perp$ ). For example, in (b), the disjunction  $0^{(64)}1^{(64)} \vee 1^{(64)}0^{(64)}$  (where  $\vee$  is bitwise-OR) is the predicate  $1^{(128)}$ , that is,  $\top$ . The algebra in (c) is nonextensional and satisfiability can be decided using a satisfiability (SAT) solver. The algebra in (d) is also nonextensional but its satisfiability is trivial, assuming unreachable states and deadends are always removed from (the automaton)  $\varphi$ .  $\boxtimes$

The following Boolean algebra mimics how finite alphabets are represented in traditional automata implementations.<sup>b</sup>

<sup>b</sup> Technically, s-FAs over the equality algebra are still slightly more general than traditional automata as s-FAs can still allow individual transitions to carry multiple characters.

*Example 2.2 (Equality Algebra).* The equality algebra over an arbitrary set  $\mathcal{D}$  has an atomic predicate  $\varphi_a$  for every  $a \in \mathcal{D}$  such that  $\llbracket \varphi_a \rrbracket = \{a\}$  as well as predicates  $\perp$  and  $\top$ . There are no formal requirements on  $\mathcal{D}$ , but one case is that  $\mathcal{D} = \Sigma_{MC}$  is a powerset of a finite set of Boolean properties as discussed above. The set of predicates  $\Psi$  is the Boolean closure generated from the atomic predicates—for example,  $\varphi_a \vee \varphi_b$  and  $\neg \varphi_a$ , where  $a, b \in \mathcal{D}$  are predicates in  $\Psi$ . Intuitively, one can think of each predicate in this algebra to be of the form  $\lambda x. x = a_1 \vee \dots \vee x = a_n$ , where each  $a_i$  is an element of  $\mathcal{D}$ . Hence, the name equality algebra.  $\boxtimes$

The following example shows how the predicate representations employed by modern SMT solvers<sup>14</sup> can be used to design Boolean algebras for arbitrarily complex domains.

*Example 2.3 (SMT Algebra).* Let  $\Psi$  be the set of all quantifier-free formulas with one fixed free variable  $x$  of a fixed type  $\tau$ . Formally,  $SMT_\tau = (\mathcal{D}, \Psi, \llbracket \_ \rrbracket, \perp, \top, \vee, \wedge, \neg)$ , where  $\mathcal{D}$  is the domain of  $\tau$ , and the Boolean operations are the corresponding connectives in SMT formulas. Intuitively,  $SMT_\tau$  represents a restricted use of an SMT solver. The interpretation function  $\llbracket \_ \rrbracket$  is defined using the operations of satisfiability checking and witness generation of an SMT solver. For example, in  $SMT_{\mathbb{Z}}$ , elements are integers and predicates are linear arithmetic formulas, such as  $\varphi_{>0} \stackrel{\text{def}}{=} x > 0$  and  $\varphi_{\text{odd}} \stackrel{\text{def}}{=} x \% 2 = 1$ .  $\boxtimes$

**Symbolic finite automata.** We can now define symbolic finite automata, which are finite automata where edge labels are replaced by predicates in a Boolean algebra.

*Definition 2.4.* A *symbolic finite automaton* (s-FA) is a tuple  $M = (\mathcal{A}, Q, q^0, F, \Delta)$ , where  $\mathcal{A}$  is an effective Boolean algebra,

$Q$  is a finite set of states,  $q^0 \in Q$  is the initial state,  $F \subseteq Q$  is the set of final states, and  $\Delta \subseteq Q \times \Psi_{\mathcal{A}} \times Q$  is a finite set of *transitions*.

Elements of the domain  $\mathcal{D}$  of  $\mathcal{A}$  are called *characters*, and elements of  $\mathcal{D}^*$  are called *strings*. A transition  $\rho = (q_1, \varphi, q_2) \in \Delta$ , also denoted  $q_1 \xrightarrow{\varphi} q_2$ , has *source state*  $q_1$ , *target state*  $q_2$ , and *guard*  $\varphi$ . For  $a \in \mathcal{D}$ , the concrete  $a$ -transition  $q_1 \xrightarrow{a} q_2$  denotes that  $q_1 \xrightarrow{\varphi} q_2$  and  $a \in \llbracket \varphi \rrbracket$  for some  $\varphi$ ;  $M$  is *deterministic* when there is at most one concrete  $a$ -transition from any source  $q_1$  and character  $a$ .

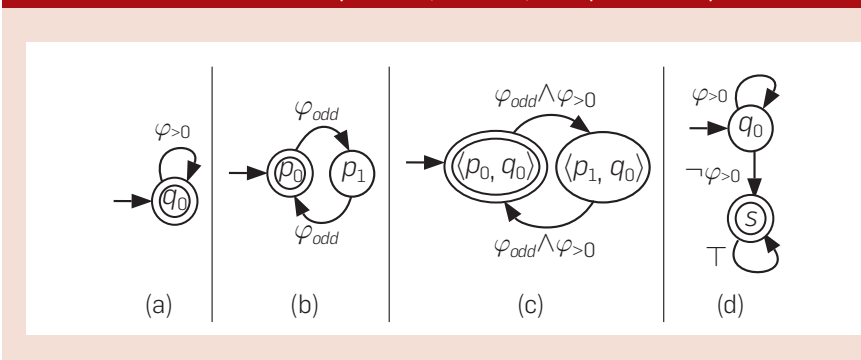
A string  $w = a_1 a_2 \dots a_k$  is *accepted at state*  $q$  iff, for  $1 \leq i \leq k$ , there exist  $a_i$ -transitions  $q_{i-1} \xrightarrow{a_i} q_i$  such that  $q_0 = q$  and  $q_k \in F$ . The set of strings accepted at  $q$  is denoted by  $\mathcal{L}_q(M)$  and the *language of*  $M$  is  $\mathcal{L}(M) = \mathcal{L}_{q^0}(M)$ .

It is convenient to work with s-FAs that are *normalized* by treating  $\Delta$  as a function from  $Q \times Q$  to  $\Psi$  with  $\Delta(p, q) = \perp$  when there is no transition from  $p$  to  $q$ . To this end, let  $\Delta(p, q) = \vee \{\varphi \mid (p, \varphi, q) \in \Delta\}$ , where  $\vee \emptyset = \perp$ . We also define  $\text{dom}(p) \stackrel{\text{def}}{=} \vee \{\varphi \mid \exists q. (p, \varphi, q) \in \Delta\}$ , as the domain-predicate of  $p$ , and say that  $p$  is *complete* if  $\llbracket \text{dom}(p) \rrbracket = \mathcal{D}_{\mathcal{A}}$ ;  $p$  is *partial* otherwise. Observe that  $p$  is partial iff  $\neg \text{dom}(p)$  is satisfiable.  $M$  is *complete* if all states of  $M$  are complete;  $M$  is *partial* otherwise.

*Example 2.5.* Examples of s-FAs are  $\mathbf{M}_{\text{pos}}$  and  $\mathbf{M}_{\text{ev/odd}}$  in Figure 2(a) and (b), respectively. These two s-FAs have 1 and 2 states, respectively, and they both operate over the Boolean algebra  $SMT_{\mathbb{Z}}$  from Example 2.3. The s-FA  $\mathbf{M}_{\text{pos}}$  accepts all strings consisting only of positive numbers, whereas the s-FA  $\mathbf{M}_{\text{ev/odd}}$  accepts all strings of even length consisting only of odd numbers. For example,  $\mathbf{M}_{\text{ev/odd}}$  accepts the string  $[1, 3, 5, 3]$  and rejects strings  $[1, 3, 5]$  and  $[51, 26]$ . The product automaton of  $\mathbf{M}_{\text{pos}}$  and  $\mathbf{M}_{\text{ev/odd}}$ ,  $\mathbf{M}_{\text{ev/odd}} \times \mathbf{M}_{\text{pos}}$ , accepts the language  $\mathcal{L}(\mathbf{M}_{\text{pos}}) \cap \mathcal{L}(\mathbf{M}_{\text{ev/odd}})$  (Figure 2(c)). Both s-FAs are partial—for example, neither of them has transitions for character 0. Finally,  $\mathbf{M}_{\text{pos}}^c$  accepts the complement of the language of  $\mathbf{M}_{\text{pos}}$  (Figure 2(d)).  $\boxtimes$

**Properties of symbolic automata.** In this section, we illustrate some basic properties of s-FAs and show how, although these models are more expressive (they can model infinite alphabets) and succinct (they allow multiple characters on individual transitions) than finite automata,

**Figure 2. Symbolic automata: (a)  $\mathbf{M}_{\text{pos}}$ ; (b)  $\mathbf{M}_{\text{ev/odd}}$ ; (c)  $\mathbf{M}_{\text{ev/odd}} \times \mathbf{M}_{\text{pos}}$ ; and (d)  $\mathbf{M}_{\text{pos}}^c$ .**



they still enjoy many desirable decidability and closure properties. A key characteristic of all s-FAs algorithms is that there is no explicit use of characters because  $\mathcal{D}$  may be infinite and the interface to the Boolean algebra does not directly support use of individual characters. This aspect is in sharp contrast to traditional finite automata algorithms.

*Closure properties.* First, such as for finite automata, nondeterminism does not add expressiveness for s-FAs.

**THEOREM 2.6 (DETERMINIZABILITY<sup>36</sup>).** *Given an s-FA  $M$  one can effectively construct a deterministic s-FA  $M_{\text{det}}$  such that  $\mathcal{L}(M) = \mathcal{L}(M_{\text{det}})$ .*

The determinization algorithm is similar to the subset construction for automata over finite alphabets, but also requires combining predicates appearing in different transitions. If  $M$  contains  $k$  inequivalent predicates and  $n$  states, then the number of distinct predicates (other than  $\top$ ) in  $M_{\text{det}}$  is at most  $2^k$  and the number of states is at most  $2^n$ . In other words, in addition to the classical state space explosion risk, there is also a predicate space explosion risk. Figure 3 illustrates such explosion for  $k = 2$  and  $n = 2$ .

Because s-FAs can be determinized, we can show that s-FAs are closed under Boolean operations using variations of traditional automata constructions.

**THEOREM 2.7 (BOOLEAN OPERATIONS<sup>36</sup>).** *Given s-FAs  $M_1$  and  $M_2$  one can effectively construct s-FAs  $M_1^c$  and  $M_1 \times M_2$  such that  $\mathcal{L}(M_1^c) = \mathcal{D}_{\mathcal{A}}^* \setminus \mathcal{L}(M_1)$  and  $\mathcal{L}(M_1 \times M_2) = \mathcal{L}(M_1) \cap \mathcal{L}(M_2)$ .*

The intersection of two s-FAs is computed using a variation of the classical product construction in which transitions are “synchronized” using conjunction. For example, the intersection of  $M_{\text{pos}}$  and  $M_{\text{ev/odd}}$  from Example 2.5 is shown in Figure 2(c). To complement a deterministic partial s-FA  $M$ ,  $M$  is first *completed* by adding a new nonfinal state  $s$  with loop  $s \xrightarrow{\top} s$  and for each partial state  $p$  a transition  $p \xrightarrow{-\text{dom}(p)} s$ . Then, the final states and the nonfinal states are swapped in  $M^c$ . Following this procedure, the complement of  $M_{\text{pos}}$  from Example 2.5 is shown in Figure 2(d).

*Decision procedures.* s-FAs enjoy the same decidability properties of finite automata:

**THEOREM 2.8 (DECIDABILITY<sup>36</sup>).** *Given s-FAs  $M_1$  and  $M_2$  it is decidable to check if  $M_1$  is empty—that is, whether  $\mathcal{L}(M_1) = \emptyset$ —and if  $M_1$  and  $M_2$  are language-equivalent—that is, whether  $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ .*

Checking emptiness requires checking what transitions are satisfiable and, once unsatisfiable transitions are removed, any path reaching a final state from an initial state represents at least one accepting string. Equivalence can be reduced to emptiness using closure under Boolean operations—that is,  $\mathcal{L}(M_1) = \mathcal{L}(M_2) \Leftrightarrow \mathcal{L}(M_1^c \times M_2) = \emptyset \wedge \mathcal{L}(M_1 \times M_2^c) = \emptyset$ .

Algorithms have also been proposed for minimizing *deterministic* s-FAs,<sup>9</sup> for checking language inclusion,<sup>25</sup> and for learning s-FAs from membership and equivalence queries.<sup>3,26,27</sup>

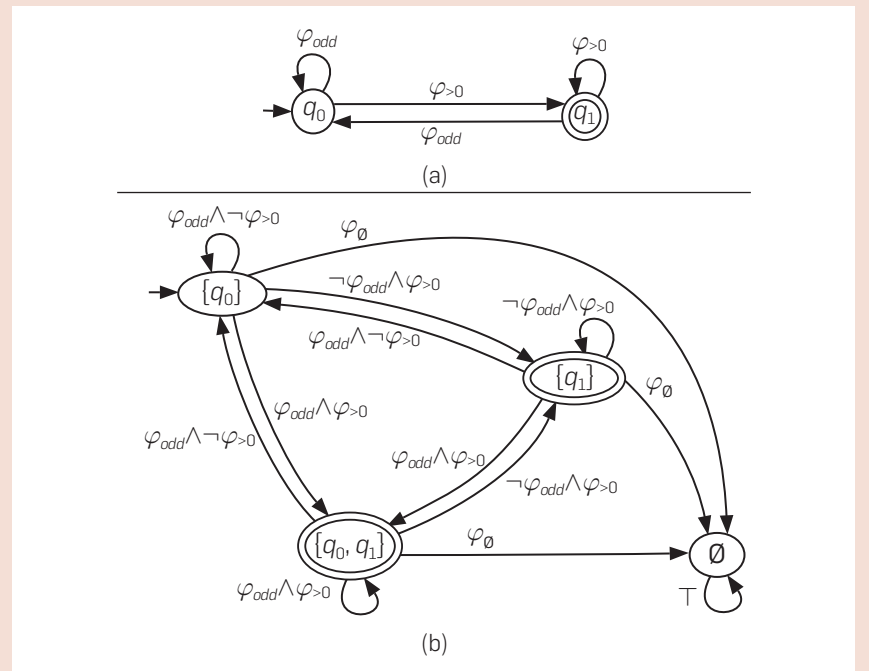
*Parametric complexities.* Because s-FAs are parametric in an underlying alphabet theory  $\mathcal{A}$ , the complexities of the aforementioned algorithms must in some way depend on the complexities of performing certain operations in  $\mathcal{A}$ . For example, the cost of checking emptiness of an s-FA depends on the cost of checking satisfiability

in  $\mathcal{A}$ . Another issue is representation of  $\Psi$ , how to measure the size  $|\varphi|$  of  $\varphi \in \Psi$ , and the cost of Boolean operations whose repeated applications may cause predicates to grow in size and thus increase related costs, in particular when  $\mathcal{A}$  is not extensional. This peculiar aspect of s-FAs opens up a new set of complexity questions that have not been studied in automata theory.

We summarize the known complexity results for the procedures we described in this section. Let  $f_{\text{sat}}(|\varphi|)$  denote the cost of checking satisfiability of a predicate  $\varphi \in \Psi$  (of size  $|\varphi|$ ) and  $M$  and  $M'$  be two *deterministic* s-FAs such that  $M$  has  $n$  states and  $m$  transitions,  $M'$  has  $n'$  states and  $m'$  transitions, and the largest predicate in  $M$  and  $M'$  has size  $\ell$ . Then, (i) checking whether  $M$  is empty has complexity  $\mathcal{O}(n + mf(\ell))$ , and (ii) checking whether  $M$  and  $M'$  are language equivalent has complexity  $\mathcal{O}(mm'f(\ell)(n + n') + \alpha(n + m))$ , where  $\alpha(\cdot)$  is related to a functional inverse of the Ackermann function. This last complexity is obtained by a straightforward adaptation of Hopcroft and Karp’s algorithm for DFA equivalence.<sup>21</sup>

For certain classes of problems, different algorithms can have different incomparable complexities. Consider,

**Figure 3: (a) Nondeterministic partial s-FA with predicates  $\varphi_{\text{odd}}$  and  $\varphi_{>0}$  from Example 2.3. (b) Equivalent complete s-FA after determinization where  $\varphi_{\emptyset}$  denotes the predicate  $\neg\varphi_{\text{odd}} \wedge \neg\varphi_{>0}$  on transitions to the “sink” state  $\emptyset$ .**



for example, the problem of minimizing a deterministic s-FA  $M$  and let  $f_{\text{sat}}(|\varphi|)$  denote the cost of checking satisfiability of a predicate  $\varphi \in \Psi$ . If  $M$  has  $n$  states and  $m$  transitions, and the largest predicate in  $M$  has size  $\ell$ , then the symbolic adaptation of Moore’s algorithm for minimizing DFAs has complexity  $\mathcal{O}(mn \cdot f_{\text{sat}}(\ell))$ , whereas the symbolic adaptation of Hopcroft’s algorithm for minimizing DFAs has complexity  $\mathcal{O}(m \log n \cdot f_{\text{sat}}(n\ell))$ , if the cost of Boolean operations is linear.<sup>9</sup> Although in the world of DFAs, Hopcroft’s algorithm<sup>20</sup> is provably better than Moore’s,<sup>29</sup> in the world of s-FAs, the two algorithms have orthogonal complexities: Hopcroft’s algorithm saves a logarithmic factor in terms of  $n$ , but this saving comes at the increased cost of satisfiability queries.

*Reduction to classical automata.* Although the set  $S$  of predicates appearing in a given s-FA (or finitely many s-FAs over the same alphabet algebra) operates generally over an infinite domain, the set of *minterms*,  $\text{Minterms}(S)$ , the maximal satisfiable Boolean combinations of  $S$  induces a finite set of equivalence classes. In general, every s-FA  $M$  can be compiled into a symbolically equivalent finite automaton over alphabet  $\text{Minterms}(S)$ , where  $\text{Predicates}(M) \subseteq S$  and  $S$  is a finite subset of  $\Psi$ . This idea, also called predicate abstraction, is often used in program verification.<sup>16</sup> However, note that the size of  $\text{Minterms}(S)$  is in general exponential in the size of  $S$ .

### Extensions of the Basic s-FA Model

Symbolic automata have been extended in various ways, as summarized in D’Antoni and Veanes<sup>12</sup>:

- Symbolic alternating automata (s-AFA)<sup>8</sup> are equivalent in expressiveness to s-FAs, but achieve succinctness by extending s-FAs with alternation and, despite the high theoretical complexity, this model can at times be more practical than s-FAs. For example, s-AFAs can efficiently check whether the intersection of multiple ASCII regular expressions is empty in cases where traditional automata suffer from state explosion.
- Symbolic extended finite automata (s-EFA)<sup>10</sup> allow s-FAs to read multiple input characters in a single transition and can be used to

model relations between adjacent input symbols.

- Symbolic tree automata (s-TA)<sup>13</sup> operate over trees instead of strings. s-FAs are a special case of s-TAs in which all nonleaf nodes in the tree have one child. s-TAs have the same closure and decidability properties as s-FAs and can also be minimized efficiently.
- Symbolic visibly pushdown automata (s-VPA)<sup>6</sup> operate over nested words,<sup>2</sup> which are used to model data with both linear and hierarchical structure—for example, XML documents and recursive program traces. s-VPAs can be determinized and have the same closure and decidability properties as s-FAs.

**Symbolic transducers.** Automata that also emit string outputs are generally called *transducers*. Perhaps the most practical extension of s-FAs is Symbolic Finite (state) Transducers (s-FTs), which extend finite transducers by allowing outputs to functionally depend on inputs, in addition to allowing predicates over those inputs.

*Example 3.1.* Consider an HTML encoder that is often used as a *string sanitizer* in a Web browser. At a high level, its purpose is to replace all nonwhitelisted characters by their equivalent HTML encoded substrings in an HTML document, for example, the character  $<$  can be encoded as the string “&lt;” (or as the string “&#000060;” using its character code 60). It is prohibitively expensive to view such an encoder as a classical Finite (state) Transducer (FT) because the standard *Unicode alphabet*<sup>c</sup> has 1,112,064 characters. So the FT would in general need over a million transitions, one for each character  $c$ : namely, a transition  $q \xrightarrow{c/\epsilon} q$  if  $c$  is whitelisted (safe), and a transition  $q \xrightarrow{c/\text{encode}(c)} q$  otherwise where  $\text{encode}(c)$  denotes the HTML encoding of  $c$ .

Instead, the corresponding s-FT would need only *two* transitions: a transition  $q \xrightarrow{\varphi_{\text{safe}}[\lambda x.x]} q$  for *safe* characters (this includes all ASCII uppercase and lowercase letters and digits), and a transition

$$q \xrightarrow{\neg\varphi_{\text{safe}}[\lambda x.'\&'; \lambda x.'\#'; \pi_6, \pi_5, \pi_4, \pi_3, \pi_2, \pi_1, \pi_0, \lambda x.';']} q$$

where  $\pi_k = \lambda x.((x \div 10^k) \bmod 10) + 48$

<sup>c</sup> <https://home.unicode.org/>

yields the  $k$ th decimal digit from the character  $x$  (recall that the standard character code of a decimal digit  $d$  is  $d + 48$ ). Note that the output of an s-FT transition is a sequence of functions—in the safe case, the sequence contains a single identity function (outputting the input character).  $\square$

We discuss next some of the main properties of s-FTs and the kinds of analysis they enable. A more formal treatment is covered in D’Antoni and Veanes.<sup>12</sup>

A transducer  $T$  denotes a relation  $\mathcal{T}_T \subseteq \mathcal{D}^* \times \mathcal{D}^*$  from input to output sequences. When  $T$  is deterministic (at most one transition can be triggered by an input symbol), this relation is a function. If  $T$  is nondeterministic but each input sequence is mapped to only one output sequence, the relation is also a function. In these last two cases, we call the transducer *functional* or *single-valued*. We call the first and second projections of the relation describing the semantics of the transducer  $T$ , its domain ( $\text{dom}(T)$ ) and range ( $\text{ran}(T)$ ), respectively. We write  $\mathcal{T}_T(u)$  for  $\{v \mid (u, v) \in \mathcal{T}_T\}$ .

Although both the domain and the range of a finite state transducer are regular, this is not true for s-FTs. By a *regular language*, here we mean a language accepted by an s-FA. Given an s-FT  $T$ , one can compute an s-FA  $\text{DOM}(T)$  such that  $\mathcal{L}(\text{DOM}(T)) = \text{dom}(T)$ . The range of an s-FT is in general not regular: Take an s-FT  $T$  with a single transition  $q \xrightarrow{\varphi_{\text{odd}}(x)/[x,x]} q$  that duplicates its input if the input is odd. The range  $\text{ran}(T)$  of the s-FT  $T$  only accepts sequences of numbers where each number at position 0 (respectively, 2, 4, ...) has to be the same as the number at position 1 (respectively, 3, 5, ...). Then,  $\text{ran}(T)$  is not regular because an s-FA cannot enforce the equality constraint between two symbols at different positions in the sequence. Also, because the alphabet is *infinite*, states cannot be used to remember what symbol is at each position such as it is normally done for finite automata over finite alphabets.

The most important property of s-FTs is that they are closed under *sequential composition*: Given two s-FTs  $S$  and  $T$ , one can compute an s-FT  $S(T)$  such that:

$$\forall u, v \in \mathcal{D}^* : (u, v) \in \mathcal{T}_{S(T)} \Leftrightarrow$$

$$\exists w \in \mathcal{D}^* : (u, w) \in \mathcal{T}_T \wedge (w, v) \in \mathcal{T}_S.$$

This enables several interesting program analyses<sup>19</sup> and optimizations, such as (*symbolic regular*) *type-checking* that is decidable when combined with closure properties of s-FAs:

Given an s-FT  $T$  and s-FAs  $M_1$  and  $M_o$ , decide if for all  $v \in \mathcal{L}(M_1)$ :  $\mathcal{T}_T(v) \subseteq \mathcal{L}(M_o)$ .

Treat any s-FA  $M$  implicitly as an s-FT by treating each transition  $p \xrightarrow{\phi} q$  as  $p \xrightarrow{\phi(x)/[x]} q$ . Then,  $M(T)$  restricts outputs of  $T$  to  $\mathcal{L}(M)$  and  $T(M)$  restricts inputs of  $T$  to  $\mathcal{L}(M)$ . So type-checking reduces to emptiness of  $\mathcal{T}_{M_o^c(T(M_1))}$ .

*Example 3.2.* By using the type-checking algorithm one can prove that, for every input sequence  $v \in \mathcal{D}^*$ , an HTML-Encoder  $T$  always produces a *valid* output:  $\mathcal{T}_T(v) \subseteq \mathcal{L}(M_o)$ , where  $M_o$  disallows all unsafe characters besides '&', and allows '&' only as an encoding prefix—for example, as described by the regular expression “( $[ \#0\text{-}\sim ] | \&(\text{amp} | \text{lt} | \text{gt} | \#[0\text{-}9]^+)$ );\*”. Observe that  $M_o^c$  accepts all invalid outputs and thus  $M_o^c(T)$  represents the restriction of  $T$  to unwanted behaviors.  $\square$

Although equivalence of finite transducers is in general undecidable,<sup>17</sup> equivalence of single-valued s-FTs is decidable and single-valuedness is itself a decidable property.<sup>37</sup> When combined with closure under composition, decidable equivalence is a powerful verification tool. For example, one can check whether composing two s-FTs  $E$  and  $D$ , representing a string encoder and decoder, respectively, yields the identity function  $I$ —that is, if  $\mathcal{T}_{D(E)} = \mathcal{T}_I$ .

A transducer  $T$  is *injective* if for all distinct  $u, v \in \mathcal{D}^*$ , we have  $\mathcal{T}_T(u) \cap \mathcal{T}_T(v) = \emptyset$ . Although injectivity of finite transducers is decidable, injectivity is *undecidable* for s-FTs.<sup>23</sup>

*Extensions of symbolic transducers.* Similarly to how s-EFAs extend s-FAs, Symbolic Extended Finite Transducers (s-EFT)<sup>10</sup> are symbolic transducers in which each transition can read more than a single character. A similar extension, called s-RTs, incorporates the notion of bounded *look-back* and *roll-back* in form of roll-backtransitions, not present in any other transducer formalisms, to accommodate default or exceptional behavior.

s-FTs have also been extended with *registers* and are called *symbolic transducers*.<sup>10, 38</sup> The key motivation is to support loop-carried data state, such as the maximal number seen so far. This model is closed under composition,

## Sanitizers provide a first line of defense against cross site scripting attacks.

but most decision problems for it are undecidable, even emptiness.

Symbolic *tree* transducers (s-TT)<sup>13</sup> operate over trees instead of strings. s-FTs are a special case of s-TTs in which all nodes in the tree have one child or are leaves. s-TTs are only closed under composition when certain assumptions hold.

### Applications of Symbolic Automata and Their Variants

*Analysis of regular expressions.* The connection between automata and regular expressions has been studied for more than 50 years. However, real-world regular expressions are much more complex than the simple model described in a typical theory of computation course. In particular, in practical regular expressions, the alphabet can contain upward of  $2^{16}$  characters due to the widely adopted UTF16 standard of Unicode characters. s-FAs can model characters as bitvectors and use various representations for predicates—for example, Binary Decision Diagrams (BDDs) or bitvector arithmetic. These representations turned out to be a viable way to model regular expression constraints in parameterized unit tests<sup>36</sup> and for random password generation.<sup>9</sup>

*Analysis of string encoders and sanitizers.* The original motivation for s-FTs was to enable static analysis for string *sanitizers*.<sup>19</sup> String sanitizers are particular string to string functions over Unicode (a very large alphabet that cannot be handled by traditional automata models) designed to encode special characters in text that may otherwise trigger malicious code execution in certain sensitive contexts, primarily in HTML pages. Thus, sanitizers provide a first line of defense against cross site scripting (XSS) attacks. When sanitizers can be represented as s-FTs, one can, for example, decide if two sanitizers  $A$  and  $B$  commute—that is, if  $\mathcal{T}_{A(B)} = \mathcal{T}_{B(A)}$ —if a sanitizer  $A$  is idempotent—that is, if  $\mathcal{T}_{A(A)} = \mathcal{T}_A$ —or if  $A$  cannot be compromised with an input attack vector—that is, if  $\text{ran}(A) \subseteq \text{SafeSet}$ . Checking such properties can help ensure the correct usage of sanitizers. s-EFTs have been used to prove that efficient implementations of BASE64 or UTF encoders and decoders correctly

invert each other.<sup>10</sup> Recently, s-EFTs have been used to automatically synthesize inverses of encoders that are correct by construction.<sup>23</sup>

*Analysis of functional programs.* Symbolic transducers have been used to perform static analysis of functional programs that operate over lists and trees.<sup>13</sup> In particular, symbolic tree transducers were used to verify HTML sanitizers and to perform deforestation, a technique used to speedup function composition in functional language compilation. These programs cannot be modeled using traditional automata models because they operate over infinite alphabets—for example, lists of integers.

*Code generation and parallelization.* Symbolic transducers can be used to expose data parallelism in computations that may otherwise seem inherently sequential. A DFA transition function can be viewed as a particular kind of *matrix multiplication*. Therefore, DFA-based pattern matching can be executed in parallel by virtue of associativity of multiplication. This idea can be lifted to symbolic transducers and applied to many common string transformations. Symbolic transducers can also be extended with *registers* and *branching rules*, which are transitions with multiple target states in the form of if-then-else statements. The main purpose of a branching rule is to support built-in determinism and to enable a way to control evaluation order of transition guards for serial code generation, so that *hot paths* can be optimized. Moreover, symbolic transducers can be composed in a manner that is similar to *loop fusion* in order to avoid intermediate data structures. The main context where these ideas have been evaluated is in log/data processing pipelines.<sup>31</sup>

*Symbolic regex matcher (SRM).* Analogously to s-FAs, *regular expressions* can also be defined *modulo* an alphabet theory  $\mathcal{A}$ , instead of using a finite alphabet. One can develop a corresponding theory of *symbolic derivatives* of such symbolic regular expressions. Symbolic derivatives enable lazy unfolding and on-the-fly creation of s-FAs that leads to a new class of more predictable matching algorithms that avoid backtracking. Such algorithms and a tool called SRM<sup>32</sup> have been developed at Microsoft Research and are being deployed daily in Azure for scanning

credentials and other sensitive content in cloud service software. The input to SRM is a .NET regular expression over a restricted set of features. SRM's linear matching complexity has helped avoid unpredictable performance in the built-in .NET regular expression engine that was susceptible to catastrophic backtracking on files with long lines, such as minified JavaScript and SQL server seeding files.

### Open Problems and Future Directions

We conclude this article with a list of open questions that are unique to symbolic automata and transducers, as well as a summary of what unexplored applications could benefit from these models.

**Theoretical directions.** *Adapting DFA algorithms to s-FAs.* Several algorithms for finite automata are based on efficient data structures that take advantage of the alphabet being finite. For example, Hopcroft's and Karp's algorithm for DFA minimization iterates over the alphabet to refine state partitions through splitting. This iteration can be avoided in s-FAs using satisfiability checks on certain carefully crafted predicates.<sup>9</sup> Paige-Tarjan's algorithm for computing forward bisimulations of NFAs is similar to Hopcroft's algorithm.<sup>30</sup> The algorithm can compute the partition of forward-bisimilar states in time  $O(km \log n)$ . However, unlike Hopcroft's algorithm, Paige-Tarjan's algorithm is hard to adapt to the symbolic setting and the current adaptation has complexity  $O(2^m \log n + 2^m f_{\text{sat}}(n\ell))$ .<sup>11</sup> By contrast, the simpler  $O(km^2)$  algorithm for forward bisimulations can be easily turned into a symbolic  $O(m^2 f_{\text{sat}}(\ell))$  algorithm.<sup>11</sup> Another example of this complexity of adaptation arises in checking equivalence of two unambiguous NFAs.<sup>34</sup>

The problem of learning symbolic automata has only received limited attention.<sup>3, 15, 26, 27</sup> Classical learning algorithms require querying an oracle for all characters in the alphabet and this is impossible for symbolic automata. On the other hand, the learner simply needs to learn the predicates on each transition of the s-FA, which might require a finite number of queries to the oracle. This is a common problem in computational learning theory and there is an opportunity to

apply concepts from this domain to the problem of learning symbolic automata.

*Properties of new models.* Some symbolic models are still not well understood because they do not have finite automata counterparts. In particular, s-EFAs<sup>10</sup> do not enjoy many good properties, but it is possible that they have practical subclasses—for example deterministic, unambiguous, etc.—with good properties.

A new model, called Symbolic Register Automata (s-RA),<sup>7</sup> combines the symbolic aspect of s-FA with the ability of comparing elements at different positions in a string—for example, equality. This model is strictly more expressive than its components and it enjoys the same decidability properties of its nonsymbolic counterpart<sup>24</sup>—for example, equivalence is decidable for deterministic s-RAs. s-RAs could be used to represent complex list properties and improve static-analysis procedures. However, their theoretical treatment is limited so far and an exciting avenue for those interested in symbolic automata.

*Complexity and expressiveness.* In classical automata theory, the complexities of the algorithms are given with respect to the number of states and transitions in the automaton. We discussed previously how the algorithm complexities in the symbolic setting depend on trade-offs made in the alphabet theory. Exactly understanding these trade-offs is an interesting research question.

**New potential applications.** *Extracting automata from recurrent neural networks.* Due to the widespread adoption of machine learning modes, there has been renewed interest in the problem of extracting “explanations” from opaque black-box models. The work that is most relevant to ours is that of using automata learning to extract finite automata from recurrent neural networks.<sup>40</sup> Existing works have used automata learning for extracting automata from very simple synthetic RNNs. In particular, these techniques can only handle small-valued features (2-4 values) and cannot handle complex real-valued feature sets. s-FAs can be used to address this limitation. In particular, recent work on learning s-FAs<sup>3</sup> could potentially be used for



extracting automata from complex real-valued neural networks.

**SMT solving with sequences.** SMT solvers<sup>14</sup> have drastically changed the world of programming languages and turned previously unsolvable problems into feasible ones. The recent interest in verifying programs operating over sequences has created a need for extending existing SMT solving techniques to handle sequences over complex theories. Solvers that are able to handle strings typically use automata. Most solvers only handle strings over finite small alphabets and s-FAs have the potential to impact the way in which such solvers for SMT are built. Recently, some SMT solvers such as Z3 have started incorporating s-FAs in the context of supporting regular expressions in the sequence theory.<sup>33</sup>

**Static analysis.** Dalla Preda et al. recently investigated how to use s-FAs to model program binaries.<sup>5</sup> s-FAs can use their state space to capture the control flow of a program and their predicates to abstract the I/O semantics of basic blocks appearing in the programs. This approach unifies existing syntactic and semantic techniques for similarity of binaries and has the promise to lead us to better understand techniques for malware detection in low-level code. The same authors recently started investigating whether, using s-FTs, the same techniques could be extended to perform analysis of reflective code—that is, code that can self-modify itself at runtime.

Argyros et al.<sup>4</sup> have used s-FA learning to extract models of Web applications and detecting inconsistencies among different applications of the same logical application. Extending this approach to the more powerful extensions of s-FAs could lead to the ability to detect complex bugs in various kinds of software.

## Conclusion

Symbolic automata and their variants have proven to be a versatile and powerful model to reason about practical applications that were beyond the reach of finite-alphabet automata models. In this article, we summarized what theoretical results are known for symbolic models, described the numerous extensions of symbolic automata, and clarified why these

models are different from their finite-alphabet counterparts. We also presented the following list of open problems we hope that the research community will help us solve: Can we provide theoretical treatments of the complexities of the algorithms for symbolic models? Can we extend classical algorithms for automata over finite alphabets to the symbolic setting? Can we use symbolic automata algorithms to design decision procedures for the SMT theory of sequences? **□**

## References

- Alur, R., D'Antoni, L., Raghothaman, M. DReX: A declarative language for efficiently evaluating regular string transformations. *ACM SIGPLAN Notices – POPL'15 50*, 1 (2015), 125–137.
- Alur, R., Madhusudan, P. Adding nesting structure to words. In *Developments in Language Theory*, O.H. Ibarra and Z. Dang, eds. Springer, Berlin, Heidelberg, 2006, 1–13.
- Argyros, G., D'Antoni, L. The learnability of symbolic automata. In *Computer Aided Verification*, H. Chockler and G. Weissenbacher, eds. Springer International Publishing, Cham, 2018, 427–445.
- Argyros, G., Stais, I., Jana, S., Keromytis, A.D., Kiyias, A. SFADiff: Automated evasion attacks and fingerprinting using black-box differential automata learning. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria, October 24–28, 2016), 1690–1701.
- Dalla Preda, M., Giacobazzi, R., Lakhotia, A., Mastroeni, I. Abstract symbolic automata: Mixed syntactic/semantic similarity analysis of executables. *ACM SIGPLAN Notices – POPL'15 50*, 1 (2015), 329–341.
- D'Antoni, L., Alur, R. Symbolic visibly pushdown automata. In *CAV'14* (2014), Springer International Publishing, Cham, 209–225.
- D'Antoni, L., Ferreira, T., Sarmmartino, M., Silva, A. Symbolic register automata. In *Computer Aided Verification*, I. Dillig and S. Tasiran, eds. Springer International Publishing, Cham, 2019, 3–21.
- D'Antoni, L., Kincaid, Z., Wang, F. A Symbolic Decision Procedure for Symbolic Alternating Finite Automata. *Electron. Notes Theor. Comput. Sci.* 336 (2018), 79–99.
- D'Antoni, L., Veanes, M. Minimization of symbolic automata. *ACM SIGPLAN Notices – POPL'14 49*, 1 (2014), 541–553.
- D'Antoni, L., Veanes, M. Extended symbolic finite automata and transducers. *Formal Methods Syst. Des.* 47, 1 (Aug. 2015), 93–119.
- D'Antoni, L., Veanes, M. Forward bisimulations for nondeterministic symbolic finite automata. In *TACAS'17, LNCS* (2017), Springer, Berlin, Heidelberg, 518–534.
- D'Antoni, L., Veanes, M. The Power of Symbolic Automata and Transducers. In: *Computer Aided Verification. Computer Aided Verification. CAV. Lecture Notes in Computer Science*. R. Majumdar, V. Kunčak, eds. Springer, Cham, 2017, vol. 10426, 47–67.
- D'Antoni, L., Veanes, M., Livshits, B., Molnar, D. Fast: A transducer-based language for tree manipulation. *ACM TOPLAS* 38, 1 (2015), 1–32.
- de Moura, L., Björner, N. Satisfiability modulo theories: Introduction & applications. *Commun. ACM* 54, 9 (Sept. 2011), 69–77.
- Drews, S., D'Antoni, L. Learning symbolic automata. In *TACAS'17* (2017), 173–189.
- Flanagan, C., Qadeer, S. Predicate abstraction for software verification. In *Predicate Abstraction for Software Verification* (2002), vol. 37, ACM, 191–202.
- Griffiths, T. The unsolvability of the equivalence problem for  $\Lambda$ -free nondeterministic generalized machines. *J. ACM* 15, 1968, 409–413.
- Holzmann, G.J. The model checker spin. *IEEE Trans. Softw. Eng.* 23, 5 (May 1997), 279–295.
- Hooimeijer, P., Livshits, B., Molnar, D., Saxena, P., Veanes, M. Fast and precise sanitizer analysis with BEK. In *Proceedings of the 20th USENIX Conference on Security, SEC'11* (Berkeley, CA, USA, 2011), USENIX Association, San Francisco, CA.
- Hopcroft, J. An  $n \log n$  algorithm for minimizing states in a finite automaton. In *Proceedings of International Symposium on Theory of Machines and Computations*, (Technion, Haifa, 1971), Z. Kohavi, ed., 189–196.
- Hopcroft, J.E., Karp, R.M. A linear algorithm for testing equivalence of finite automata. Technical Report 114, Cornell University, Ithaca, NY, December 1971.
- Hopcroft, J.E., Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, Reading, Mass, 1979.
- Hu, Q., D'Antoni, L. Automatic program inversion using symbolic transducers. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation* (2017), ACM, New York, NY, USA, 376–389.
- Kaminski, M., Francez, N. Finite-memory automata. *TCS* 134, 2 (1994), 329–363.
- Keil, M., Thiemann, P. Symbolic solving of extended regular expression inequalities. In *FSTTCS'14* (2014), LIPIcs, Dagstuhl, Germany, 175–186.
- Maler, O., Mens, I.-E. A generic algorithm for learning symbolic automata from membership queries. *Logical Methods Comput. Sci.* 11, 3:13 (2014), 2015.
- Maler, O., Mens, I.-E. Learning regular languages over large ordered alphabets. *Log. Methods Comput. Sci.* 11, 3(2015)
- Mamouras, K., Raghothaman, M., Alur, R., Ives, Z.G., Khanna, S. StreamQRE: Modular specification and efficient evaluation of quantitative queries over streaming data. ACM, New York, NY, USA, 2017, 693–708.
- Moore, E.F. Gedanken-experiments on sequential machines. In: *Automata Studies, Annals of Mathematics Studies*. C. E. Shannon, J. McCarthy, eds. Litho-Printed, Princeton University Press, Princeton, vol. 34 (1956), 129–153.
- Paige, R., Tarjan, R.E. Three partition refinement algorithms. *SIAM J. Comput.* 16, 6 (1987), 973–989.
- Saarikivi, O., Veanes, M., Mytkowicz, T., Musuvathi, M. Fusing effectful comprehensions. In *ACM SIGPLAN Notices – PLDI'17* (2017), ACM, New York, NY.
- Saarikivi, O., Veanes, M., Wan, T., Xu, E. Symbolic regex matcher. In *TACAS'19, LNCS* (2019), Springer, Cham, Switzerland.
- C. Stanford, M. Veanes, and N. Björner. Symbolic Boolean Derivatives for Efficiently Solving Extended Regular Expression Constraints. Technical Report MSR-TR-2020-25, Microsoft, August 2020.
- Stearns, R.E., Hunt, H.B. On the equivalence and containment problems for unambiguous regular expressions, grammars, and automata. *SIAM J. Comput.* 14, 3, 598–611.
- Vardi, M.Y. Linear-time model checking: Automata theory in practice. In *Implementation and Application of Automata, 12th International Conference, CIAA 2007* (Prague, Czech Republic, July 16–18, 2007), Revised Selected Papers, 5–10.
- Veanes, M., de Halleux, P., Tillmann, N. Rex: symbolic regular expression explorer. In: *Third international conference on software testing, verification and validation* (2010), IEEE, Paris, 498–507.
- Veanes, M., Hooimeijer, P., Livshits, B., Molnar, D., Björner, N. Symbolic finite state transducers: Algorithms and applications. *ACM SIGPLAN Notices – POPL'12 47*, 1 (2012), 137–150.
- Veanes, M., Mytkowicz, T., Molnar, D., Livshits, B. Data-parallel string-manipulating programs. *ACM SIGPLAN Notices – POPL'15 50*, 1 (2015), 139–152.
- Watson, B.W. Implementing and using finite automata toolkits. In *Extended Finite State Models of Language*. Cambridge University Press, Cambridge, England, 1999, 19–36.
- Weiss, G., Goldberg, Y., Yahav, E. Extracting automata from recurrent neural networks using queries and counterexamples. In *Proceedings of the 35th International Conference on Machine Learning, J. Dy and A. Krause, eds. Volume 80, PMLR, Stockholm, Sweden, Stockholm Sweden, 10–15 Jul 2018, 5247–5256*

Loris D'Antoni, University of Wisconsin-Madison, WI, USA.

Margus Veanes, Microsoft Research, Redmond, WA, USA.

Copyright held by authors/owners.  
Publication rights licensed to ACM.

# research highlights

---

P. 97

**Technical  
Perspective**  
**A Logical Step  
Toward the Graph  
Isomorphism  
Problem**

By Pascal Schweitzer

P. 98

**Isomorphism, Canonization,  
and Definability for Graphs  
of Bounded Rank Width**

By Martin Grohe and Daniel Neuen

---

P. 106

**Technical  
Perspective**  
**Robust Statistics  
Tackle New  
Problems**

By Jacob S. Steinhardt

P. 107

**Robustness Meets Algorithms**

By Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane,  
Jerry Li, Ankur Moitra, and Alistair Stewart

---

# Technical Perspective

## A Logical Step Toward the Graph Isomorphism Problem

By Pascal Schweitzer

THE GRAPH ISOMORPHISM problem remains one of those mysteries in theoretical computer science that fascinates laypersons and experts alike. In 1979, Garey and Johnson mentioned the problem in their renowned book on computers and intractability but, in fact, it dates back even earlier and has been unresolved for over half a century. In 2015, a major advance hit the media: Babai's quasipolynomial algorithm. This was the first improvement for the general problem in over 30 years. And yet it remains an open problem. At its core, the problem captures the algorithmic detection of symmetries of combinatorial objects.

Maybe surprisingly, there are various and quite distinct areas in which the problem finds applications. We can find one such application domain in the context of logics. Indeed, the isomorphism problem appears to be closely linked to what logicians call the quest for a logic that captures PTIME, a major open problem in finite model theory. In a nutshell we want a logic that is powerful enough to precisely allow everything that is efficiently solvable to be captured in a formula but is not more powerful than that. In terms of databases, we want a query language expressive enough to allow all queries that can be efficiently answered but again not more than that.

For a logician on the quest, designing an efficient isomorphism algorithm for some class of structures is not particularly helpful. What is needed is to be able to “solve” the problem within a logic. In fact, while this is a necessary step, it might not be enough. What does suffice, however, is to be able to “solve” the canonization problem for the class within a logic. Intuitively, the canonization problem asks for an efficient normal form, some way of unambiguously (that is, canonically) representing a given input. The canonization problem is closely linked to the isomorphism problem, and considering practical applica-

tions of isomorphism, it turns out that a solution to the canonization problem is what users often actually want and need.


In the following paper, Grohe and Neuen consider the class of graphs of bounded rank width. The class is equivalent to graphs of bounded clique width, a term sometimes more familiar to graph theorists who do not have a focus on algorithmic aspects. The class is not readily defined, but it does play a central role in graph structure theory. Quintessentially, graphs in this class are recursively decomposable into small parts so that the interplay between the different parts is not too complicated. Several years ago, the class was essentially the only class of naturally decomposable graphs for which we had no efficient isomorphism test.

In 2015, Grohe and Schweitzer found an efficient test, but for the questing logician the answer was unsatisfactory. The algorithm involves a barrage of techniques from computational group theory, making the solution not only complicated but also not providing a sufficient answer to logical definability and canonization. In their new paper, Grohe and Neuen instead use a purely combinatorial approach: the Weisfeiler-Leman algorithm. This algorithm has its roots in algebraic graph theory, it is used in Babai's algorithm, and it recently found applications in machine learning in the form of graph kernels.

The authors now show, remarkably, that the well-established Weisfeiler-

Leman algorithm in its plain form solves the isomorphism problem for bounded-rank width graphs. In fact, the paper even goes the crucial step further to show that canonization can be solved in the logic corresponding to the algorithm (fixed-point logic with counting). Overall, they obtain a logic that captures PTIME on graphs of bounded rank width. This means, essentially, that everything that can be algorithmically solved efficiently on these graphs can alternatively be expressed by a logical formula of fairly simple composition.

Grohe had previously demonstrated this type of approach can be very fruitful. In fact, his 2017 book on the matter, spanning more than 500 pages, executes the approach for graph classes closed under taking minors. Yet, the class of graphs of bounded rank width is one of the most general classes for which the approach has been put to work successfully.

Overall, in comparison to the previously best result, the paper provides us with a stronger result based on a simpler algorithm. On top of that, the overall proof is—at least from my perspective—simpler and, for many, easier to understand. Finally, the constants in the running time of the algorithm, which translate in logic terms to the number of variables needed, are small (specifically, linear in the rank width) and in some sense as small as they can be. Central to the analysis is the introduction of the novel concepts of split pairs and flip functions, which cleanly facilitate many of the arguments. It is common to find that when it comes to matters of theory, hitting the right approach, careful choice of definitions, and a hunch can make all the difference. Overall, Grohe and Neuen have found a clean and neat way to complete the logicians' quest for graphs of bounded rank width. 

**Grohe and Neuen use a purely combinatorial approach: the Weisfeiler-Leman algorithm.**

Pascal Schweitzer is a professor at the Technische Universität Kaiserslautern, Germany.

Copyright held by author.

# Isomorphism, Canonization, and Definability for Graphs of Bounded Rank Width

By Martin Grohe and Daniel Neuen

## Abstract

We investigate the interplay between the graph isomorphism problem, logical definability, and structural graph theory on a rich family of dense graph classes: graph classes of bounded rank width. We prove that the combinatorial Weisfeiler-Leman algorithm of dimension  $(3k + 4)$  is a complete isomorphism test for the class of all graphs of rank width at most  $k$ . A consequence of our result is the first polynomial time canonization algorithm for graphs of bounded rank width.

Our second main result addresses an open problem in descriptive complexity theory: we show that fixed-point logic with counting expresses precisely the polynomial time properties of graphs of bounded rank width.

## 1. INTRODUCTION

The question of whether there is an efficient algorithm deciding whether two graphs are isomorphic is one of the oldest and best-known open problems in theoretical computer science. Already mentioned in Karp's<sup>18</sup> seminal article on NP-complete combinatorial problems, graph isomorphism (from now on: GI) has remained one of the few natural problems in NP that is neither known to be solvable in polynomial time nor known to be NP-complete. The problem has received considerable attention recently because of Babai's<sup>1</sup> breakthrough algorithm deciding GI in quasipolynomial time  $n^{\text{polylog}(n)}$ .

However, the question of whether GI is solvable in polynomial time remains wide open. Polynomial-time algorithms are known for the restrictions of GI to many interesting graph classes, for example the class of planar graphs<sup>14</sup>, classes of bounded degree<sup>19</sup>, even all classes excluding a fixed graph as a topological subgraph<sup>9</sup>, and only recently, graph classes of bounded rank width.<sup>11</sup>

*Rank width*, introduced by Oum and Seymour,<sup>21</sup> is a graph invariant that measures how well a graph can be decomposed hierarchically in a certain style. In this respect, it is similar to the better-known tree width, but where tree width measures the complexity, or width, of a separation in a hierarchical decomposition in terms of the “connectivity” between the two sides, rank width measures the complexity of a separation in terms of the rank of the adjacency matrix of the edges between the two sides of the separation (see Figure 1). As opposed to tree width, rank width is also a meaningful measure of simplicity for dense graphs. For example, the rank width of a complete graph, arguably the simplest dense graph, is 1. Rank width is closely related to clique width,<sup>5</sup> another well-studied graph invariant based on graph grammars. Many hard algorithmic problems can be solved efficiently on graphs of bounded rank

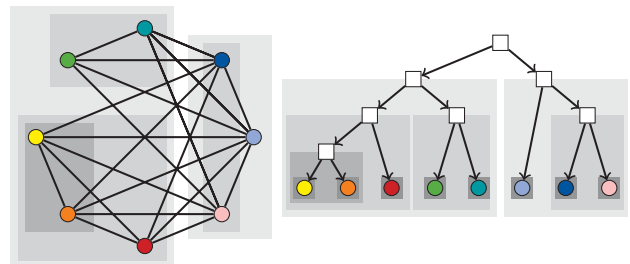
width, or equivalently, bounded clique width, among them all problems definable in monadic second-order logic.<sup>4</sup>

In this paper we study the graph isomorphism problem and the closely related graph canonization problem as well as logical definability and descriptive complexity on graph classes of bounded rank width. At the technical core of our work is a beautiful connection (from Cai et al.<sup>2</sup>) between the Weisfeiler-Leman algorithm, a generic combinatorial graph isomorphism algorithm, and an Ehrenfeucht-Fr ass e style game that is usually used to prove lower bounds in logic.

Although a polynomial time isomorphism algorithm for graph classes of bounded rank width was known<sup>11</sup> prior to this work, the running time of that algorithm is  $n^{f(k)}$ , where  $n$  is the number of vertices and  $k$  is the rank width of the input graph, and  $f$  is a *nonelementary* function. Moreover, the algorithm is extremely complicated, using both advanced techniques from structural graph theory<sup>12</sup> and the group-theoretic graph isomorphism machinery.<sup>19</sup>

Our first contribution is a simple isomorphism test for graphs of rank width at most  $k$  running in time  $n^{O(k)}$ . Indeed, the algorithm we use is a generic combinatorial isomorphism test known as the Weisfeiler-Leman algorithm.<sup>23, 1, 2</sup> The  $\ell$ -dimensional Weisfeiler-Leman algorithm ( $\ell$ -WL) iteratively colors  $\ell$ -tuples of vertices of the two input graphs and then compares the resulting color patterns. If they differ, we know that the two input graphs are nonisomorphic. If two graphs have the same color pattern, in general, they may still be nonisomorphic.<sup>2</sup> Thus  $\ell$ -WL is not a *complete* isomorphism

**Figure 1.** A dense graph of rank width 2 (left) and a hierarchical “rank decomposition” of this graph of width 2 (right). The rank width is low because the vertex cuts the decomposition on the right induces are very regular.



A previous version of this paper, entitled “Canonisation and Definability for Graphs of Bounded Rank Width” was published in *Proceedings of the 34<sup>th</sup> Annual ACM/IEEE Symp. on Logic in Computer Science* (Vancouver, BC, Canada, 2019), 1–13.

test for all graphs. However, we prove that it is for graphs of bounded rank width. We say that  $\ell$ -WL identifies a graph  $G$  if it distinguishes  $G$  from every graph  $H$  not isomorphic to  $G$ .

**THEOREM 1.** *The  $(3k + 4)$ -dimensional Weisfeiler-Leman algorithm identifies every graph of rank width at most  $k$ .*

Combining this theorem with a result due to Immerman and Lander on the running time of the WL algorithm, we obtain the following:

**COROLLARY 2.** *Isomorphism of graphs of rank width  $k$  can be decided in time  $O(n^{3k+5} \log n)$ .*

Another way of stating Theorem 1 is that the *Weisfeiler-Leman (WL) dimension*<sup>8</sup> of graphs of rank width  $k$  is at most  $3k + 4$ . It is known that many natural graph classes have bounded WL dimension, among them all classes of graphs excluding some fixed graph as a minor.<sup>8</sup> But most of these classes consist of sparse graphs with an edge number linear in the number of vertices. Our result adds a rich family of classes that include dense graphs to the picture.

In many applications of graph isomorphism, it is actually necessary to compute a canonical representation of a graph, a so-called *canonical form*, rather than just decide if two graphs are isomorphic. A *canonical form* for a graph class  $\mathcal{C}$  is a function  $\kappa: \mathcal{C} \rightarrow \mathcal{C}$  such that

1.  $\kappa(G) \cong G$  for all  $G \in \mathcal{C}$ , and
2.  $\kappa(G) = \kappa(H)$  for all isomorphic graphs  $G, H \in \mathcal{C}$ .

Note that the isomorphism problem for a class  $\mathcal{C}$  easily reduces to computing a canonical form for  $\mathcal{C}$  (i.e., given a graph  $G \in \mathcal{C}$ , compute  $\kappa(G)$ ). A reduction in the other direction is not known. However, it is known that if a class of graphs has WL dimension at most  $\ell$  then there is an algorithm computing a canonical form for this class running in time  $O(n^{\ell+3} \log n)$ . Hence, as another corollary to Theorem 1, we obtain the first polynomial-time canonization algorithm for graphs of bounded rank width.

**COROLLARY 3.** *There is an algorithm computing a canonical form for the class of graphs of rank width at most  $k$  in time  $O(n^{3k+7} \log n)$ .*

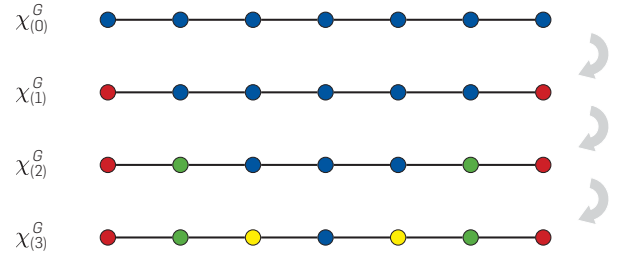
The second part of our paper is concerned with descriptive complexity theory, which aims to connect computational complexity and descriptive (or logical) complexity. The central open question of the field is whether there is a logic that *captures polynomial time*.<sup>3, 13</sup> We will defer a more detailed discussion of this question and its background to Section 4 and at this point only state our main result.

**THEOREM 4.** *Fixed-point logic with counting FP+C captures polynomial time on every class of graphs of bounded rank width.*

On a very high level, the connection between the two theorems is that to prove Theorem 4 we mimic the proof of Theorem 1 within the realms of the logic FP+C.

The rest of this article is organized as follows: Section 2 is

**Figure 2.** The iterations of the color refinement algorithm on a path of length 6.



devoted to a thorough introduction to the Weisfeiler-Leman algorithm. In Section 3, we formally introduce rank width and informally sketch the proof of Theorem 1. Finally, Section 4 is devoted to descriptive complexity theory and Theorem 4.

## 2. WEISFEILER-LEMAN ALGORITHM

### 2.1. The color refinement algorithm

One of the simplest combinatorial procedures to tackle the graph isomorphism problem is the color refinement algorithm. In a nutshell, the basic idea is to label vertices of the graph with their iterated degree sequence. More precisely, an initially uniform coloring is repeatedly refined by counting, for each color, the number of neighbors of that color.

Let  $G$  be a graph. Let  $\chi_1, \chi_2: V(G) \rightarrow C$  be colorings of vertices where  $C$  is some finite set of colors. The coloring  $\chi_1$  *refines*  $\chi_2$ , denoted by  $\chi_1 \preceq \chi_2$ , if  $\chi_1(v) = \chi_1(w)$  implies  $\chi_2(v) = \chi_2(w)$  for all  $v, w \in V(G)$ . The colorings  $\chi_1$  and  $\chi_2$  are *equivalent*, denoted by  $\chi_1 \equiv \chi_2$ , if  $\chi_1 \preceq \chi_2$  and  $\chi_2 \preceq \chi_1$ .

We give a formal description of the color refinement algorithm. Initially, all vertices are assigned the same color. Formally, we define  $\chi_{(0)}^G(v) := 0$  for all  $v \in V(G)$ . Then, the coloring is iteratively refined as follows: for  $i > 0$ , we define  $\chi_{(i)}^G(v) := (\chi_{(i-1)}^G(v); \mathcal{M}_i(v))$  where

$$\mathcal{M}_i(v) := \{\{\chi_{(i-1)}^G(w) \mid w \in N_c(v)\}\}$$

is the multiset of colors for the neighbors computed in the previous round.

In each round, the algorithm computes a coloring that is finer than the one computed in the previous round, that is,  $\chi_{(i)}^G \preceq \chi_{(i-1)}^G$ . At some point, this procedure stabilizes meaning the coloring does not become strictly finer anymore. In other words, there is some minimal  $i^* \in \mathbb{N}$  such that  $\chi_{(i^*)}^G \equiv \chi_{(i^*+1)}^G$ . We denote the corresponding coloring as the *stable* coloring and define  $\chi^G := \chi_{(i^*)}^G$ . As an example, the sequence of colorings for a path of length 6 is depicted in Figure 2.

The color refinement algorithm takes as input a graph  $G$  and outputs (a coloring that is equivalent to)  $\chi^G$ . It can be implemented in almost linear time, that is, in time  $O((n + m) \log n)$  where  $n$  denotes the number of vertices and  $m$  the number of edges of  $G$ .

One of the most common applications of the color refinement algorithm is to serve as a heuristic for GI. As the coloring computed by the color refinement algorithm is defined in an isomorphism-invariant manner, every isomorphism

$\varphi: G \cong H$  between two graphs  $G$  and  $H$  satisfies that  $\chi^G(v) = \chi^H(\varphi(v))$  for all  $v \in V(G)$ . The color refinement algorithm *distinguishes*  $G$  and  $H$  if there exists a color  $c$  such that

$$\left| (\chi^G)^{-1}(c) \right| \neq \left| (\chi^H)^{-1}(c) \right|.$$

In order to determine algorithmically whether the color refinement algorithm *distinguishes*  $G$  and  $H$ , one typically runs the color refinement algorithm on the disjoint union of  $G$  and  $H$  and determines whether there is some color  $c$  that appears a different number of times in the two graphs.

If the color refinement algorithm does not distinguish  $G$  and  $H$ , we write  $G \simeq H$ . Note that

$$G \cong H \Rightarrow G \simeq H$$

for all graphs  $G$  and  $H$ . Unfortunately, the backward direction of the implication does not hold. For example, the color refinement algorithm does not distinguish the disjoint union of two triangles from a cycle of length six although the two graphs are nonisomorphic.

Still, despite its simplicity, the color refinement algorithm serves as a complete isomorphism test for a large number of graphs. We say the color refinement algorithm *identifies* a graph  $G$  if it distinguishes  $G$  from every other nonisomorphic graph  $H$ . For example, it is known that the color refinement algorithm identifies random graphs asymptotically almost surely.

In combination with its efficiency, this makes the color refinement algorithm a popular subroutine in the context of GI, which is used in many practical and theoretical algorithms.

## 2.2. The $k$ -dimensional algorithm

Although the color refinement algorithm often serves as a complete isomorphism test, it is still quite restricted in its power. For example, given any two  $d$ -regular graphs  $G$  and  $H$ , the color refinement algorithm does not distinguish  $G$  and  $H$ . The *Weisfeiler-Leman algorithm* provides an extension to the color refinement algorithm, which, instead of only coloring single vertices, colors  $k$ -tuples of vertices for some fixed number  $k$ . This gives the algorithm additional expressive power for increasing values of  $k$ , but comes at the price of higher computational complexity.

Let  $G$  be a graph and let  $k \in \mathbb{N}$ . The  $k$ -dimensional Weisfeiler-Leman algorithm ( $k$ -WL) is a procedure that, given a graph  $G$ , first computes an isomorphism-invariant initial coloring of the  $k$ -tuples of vertices and then iteratively refines this coloring in an isomorphism-invariant way. The 1-dimensional Weisfeiler-Leman algorithm corresponds exactly to the color refinement algorithm described above.

For the description of the algorithm for higher dimensions, fix  $k \geq 2$  and let  $G = (V, E)$  be a graph. The initial coloring  $\chi_{(0)}^{G,k}$  computed by  $k$ -WL determines for each  $\bar{v} \in V^k$  the isomorphism-type of the underlying ordered induced subgraph. More precisely,  $\chi_{(0)}^{G,k}(v_1, \dots, v_k) = \chi_{(0)}^{G,k}(w_1, \dots, w_k)$  if for all  $i, j \in [k]$  it holds  $v_i = v_j$  if and only if  $w_i = w_j$ , and  $v_i v_j \in E$  if and only if  $w_i w_j \in E$ . The initial coloring is refined by iteratively computing colorings  $\chi_{(i)}^{G,k}$  for  $i > 0$ . For  $\bar{v} = (v_1, \dots, v_k)$  and  $w \in V$ , let  $\bar{v}[i/w] := (v_1, \dots, v_{i-1}, w, v_{i+1}, \dots, v_k)$  be the tuple

obtained from  $\bar{v}$  by replacing the  $i$ -th entry with vertex  $w$ . Now we define  $\chi_{(i)}^{G,k}(\bar{v}) := (\chi_{(i-1)}^{G,k}(\bar{v}); \mathcal{M}_i(\bar{v}))$  where

$$\mathcal{M}_i(\bar{v}) := \left\{ \left\{ \left( \chi_{(i-1)}^{G,k}(\bar{v}[1/w]), \dots, \chi_{(i-1)}^{G,k}(\bar{v}[k/w]) \right) \mid w \in V \right\} \right\}.$$

From the definition of the colorings, it is immediately clear that  $\chi_{(i)}^{G,k} \preceq \chi_{(i-1)}^{G,k}$ . Now let  $i^* \in \mathbb{N}$  be the minimal number such that  $\chi_{(i^*)}^{G,k} \equiv \chi_{(i^*+1)}^{G,k}$ . For this  $i^*$ , the coloring  $\chi_{(i^*)}^{G,k}$  is called the *stable* coloring of  $G$  with respect to  $k$ -WL and is denoted by  $\chi^{G,k}$ .

As an example, the coloring  $\chi^{G,2}$  where  $G$  is a cycle of length 9 is depicted in Figure 3.

The  $k$ -dimensional Weisfeiler-Leman algorithm takes as input a graph  $G$  and computes (a coloring that is equivalent to)  $\chi^{G,k}$ .

**THEOREM 5** (Immerman and Lander<sup>17</sup>). *Let  $G$  be a graph. Then, an isomorphism-invariant coloring that is equivalent to  $\chi^{G,k}$  can be computed in time  $O(n^{k+1} \log n)$ .*

We extend the notations introduced for the color refinement algorithm. The  $k$ -dimensional Weisfeiler-Leman algorithm *distinguishes* two graphs  $G$  and  $H$  if there is a color  $c$  such that  $\left| (\chi_{(i)}^{G,k})^{-1}(c) \right| \neq \left| (\chi_{(i)}^{H,k})^{-1}(c) \right|$ . In this case it follows that  $G$  and  $H$  are nonisomorphic. Two graphs  $G$  and  $H$  are *equivalent* with respect to  $k$ -WL, denoted by  $G \simeq_k H$ , if they are not distinguished by  $k$ -WL. A graph  $G$  is *identified* by  $k$ -WL if  $G \simeq_k H$  if and only if  $G \cong H$  for all graphs  $H$ . The *Weisfeiler-Leman dimension* of a graph  $G$  is the smallest number  $k \in \mathbb{N}$  such that  $k$ -WL identifies  $G$ .

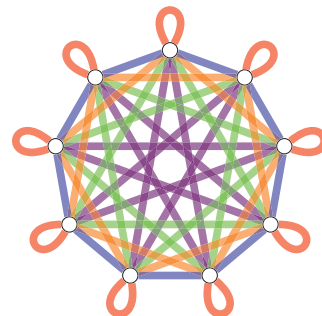
## 2.3. A pebble game

Providing upper bounds on the Weisfeiler-Leman dimension of certain graphs often turns out to be rather complicated when arguing directly with the definition of the Weisfeiler-Leman algorithm. Indeed, it is often more convenient to exploit other characterizations of the power of the Weisfeiler-Leman algorithm for this task. In particular, the following pebble game is known to capture the same information as the Weisfeiler-Leman algorithm.

Let  $k \in \mathbb{N}$ . For graphs  $G, H$  on the same number of vertices, we define the *bijective  $k$ -pebble game*  $\text{BP}_k(G, H)$  as follows:

- The game has two players called Spoiler and Duplicator.
- The game proceeds in rounds. Each round is associated

**Figure 3. The stable coloring of 2-WL on a cycle of length 9. In this example,  $\chi^{G,2}(v, w) = \chi^{G,2}(w, v)$  for all  $v, w \in V(G)$  and thus, all colors are represented by undirected edges.**



with a pair of positions  $(\bar{v}, \bar{w})$  with  $\bar{v} \in V(G)^\ell$  and  $\bar{w} \in V(H)^\ell$  where  $0 \leq \ell \leq k$ .

- The initial position of the game is  $((), ())$  (the pair of empty tuples).
- Each round consists of the following steps: Suppose the game is in position  $(\bar{v}, \bar{w}) = ((v_1, \dots, v_\ell), (w_1, \dots, w_\ell))$ . First, Spoiler chooses whether to remove a pair of pebbles or to play a new pair of pebbles. The first option is only possible if  $\ell > 0$  and the latter option is only possible if  $\ell < k$ .

If Spoiler wishes to remove a pair of pebbles, he picks some  $i \in [\ell]$  and the game moves to position  $(\bar{v} \setminus i, \bar{w} \setminus i)$  where  $\bar{v} \setminus i := (v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_\ell)$  (and  $\bar{w} \setminus i$  is defined in the same way).

Otherwise, a new pair of pebbles is played and the following steps are performed:

(D) Duplicator picks a bijection  $f: V(G) \rightarrow V(H)$ .

(S) Spoiler chooses  $v \in V(G)$  and sets  $w := f(v)$ .

The new position is  $((v_1, \dots, v_\ell, v), (w_1, \dots, w_\ell, w))$ .

Spoiler wins the play if for the current position  $(\bar{v}, \bar{w}) = ((v_1, \dots, v_\ell), (w_1, \dots, w_\ell))$  the induced graphs are not isomorphic. More precisely, Spoiler wins if there are  $i, j \in [\ell]$  such that  $v_i = v_j \not\leftrightarrow w_i = w_j$  or  $v_i v_j \in E(G) \not\leftrightarrow w_i w_j \in E(H)$ . If the play never ends, Duplicator wins.

We say that Spoiler (resp. Duplicator) *wins the bijective  $k$ -pebble game*  $\text{BP}_k(G, H)$  if Spoiler (resp. Duplicator) has a winning strategy for the game.

Moreover, for positions  $(\bar{v}, \bar{w}) \in V(G)^\ell \times V(H)^\ell$ ,  $\ell \leq k$ , Spoiler (resp. Duplicator) *wins the game*  $\text{BP}_k(G, H)$  *from position*  $(\bar{v}, \bar{w})$  if Spoiler (resp. Duplicator) has a winning strategy in the game  $\text{BP}_k(G, H)$  starting from initial position  $(\bar{v}, \bar{w})$ .

The next theorem connects the bijective  $k$ -pebble game to the Weisfeiler-Leman algorithm.

**THEOREM 6** (Cai et al.<sup>2</sup>). *Let  $G, H$  be two graphs and let  $\bar{v} \in V(G)^k$  and  $\bar{w} \in V(H)^k$ . Then  $\chi^{G,k}(\bar{v}) = \chi^{H,k}(\bar{w})$  if and only if Duplicator wins the pebble game  $\text{BP}_{k+1}(G, H)$  from the position  $(\bar{v}, \bar{w})$ .*

**COROLLARY 7.** *Let  $G, H$  be two graphs. Then  $G \simeq_k H$  if and only if Duplicator wins the pebble game  $\text{BP}_{k+1}(G, H)$ .*

### 3. RANK WIDTH

Rank width is a graph invariant that was first introduced by Oum and Seymour<sup>21</sup> and that measures the width of a certain style of hierarchical decomposition of graphs. Intuitively, the aim is to repeatedly split the vertex set of the graph along cuts of low complexity in a hierarchical fashion. For rank width, the complexity of a cut is measured in terms of the rank of the matrix capturing the adjacencies between the two sides of the cut over the 2-element field  $\mathbb{F}_2$ .

Let  $G$  be a graph. For  $X, Y \subseteq V(G)$ , we define  $M(X, Y) \in \mathbb{F}_2^{X \times Y}$  where  $(M(X, Y))_{x,y} = 1$  if and only if  $xy \in E(G)$ . Furthermore,  $\rho_2(X) = \text{rk}_2(M(X, X))$  where  $\bar{X} = V(G) \setminus X$  and  $\text{rk}_2(A)$  denotes the  $\mathbb{F}_2$ -rank of a matrix  $A$ .

A *rank decomposition* of  $G$  is a tuple  $(T, \gamma)$  consisting of a binary directed tree  $T$  and a mapping  $\gamma: V(T) \rightarrow 2^{V(G)}$  such that

- (R.1)  $\gamma(r) = V(G)$  where  $r$  is the root of  $T$ ,
- (R.2)  $\gamma(t) = \gamma(t_1) \cup \gamma(t_2)$  and  $\gamma(t_1) \cap \gamma(t_2) = \emptyset$  for all internal nodes  $t \in V(T)$  with children  $t_1$  and  $t_2$ , and

- (R.3)  $|\gamma(t)| = 1$  for all  $t \in L(T)$ , where  $L(T)$  denotes the set of leaves of the tree  $T$ .

Note that, instead of giving  $\gamma$ , we can equivalently specify a bijection  $f: L(T) \rightarrow V(G)$  (this completely specifies  $\gamma$  by Condition (R.2)). The *width* of a rank decomposition  $(T, \gamma)$  is

$$\text{wd}(T, \gamma) := \max\{\rho_2(\gamma(t)) \mid t \in V(T)\}.$$

The rank width of a graph  $G$  is

$$\text{rw}(G) := \min\{\text{wd}(T, \gamma) \mid (T, \gamma) \text{ is a rank decomp. of } G\}.$$

Examples of graphs  $G$  and rank decompositions of  $G$  are provided in Figure 1 and (more detailed) in Figure 4.

Clique width<sup>5</sup> is another measure aiming to describe the structural complexity of a graph. It is based on a natural graph grammar.

For  $k \in \mathbb{N}$ , a  *$k$ -graph* is a pair  $(G, \text{lab})$  where  $G$  is a graph and  $\text{lab}: V(G) \rightarrow [k]$  is a labeling of vertices. We define the following four operations for  $k$ -graphs:

1. For  $i \in [k]$ , let  $\bullet_i$  denote an isolated vertex with label  $i$ .
2. For  $i, j \in [k]$  with  $i \neq j$ , we define  $\eta_{i,j}(G, \text{lab}) := (G', \text{lab})$  where  $V(G') := V(G)$  and  $E(G') := E(G) \cup \{vw \mid \text{lab}(v) = i \wedge \text{lab}(w) = j\}$ .
3. For  $i, j \in [k]$ , we define  $\rho_{i \rightarrow j}(G, \text{lab}) := (G, \text{lab}')$  where  $\text{lab}'(v) := \begin{cases} j & \text{if } \text{lab}(v) = i \\ \text{lab}(v) & \text{otherwise} \end{cases}$ .
4. For two  $k$ -graphs  $(G, \text{lab})$  and  $(G', \text{lab}')$ , we define  $(G, \text{lab}) \oplus (G', \text{lab}')$  to be the disjoint union of the two  $k$ -graphs.

A  *$k$ -expression*  $t$  is a well-formed expression in these symbols and defines a  $k$ -graph  $(G, \text{lab})$ . In this case,  $t$  is a  $k$ -expression for  $G$ . The clique width of a graph  $G$ , denoted by  $\text{cw}(G)$ , is the minimum  $k \in \mathbb{N}$  such that there is a  $k$ -expression for  $G$ .

*Example 1.* The expression

$$\eta_{1,2}(\rho_{2 \rightarrow 1}(\eta_{1,2}(\bullet_1 \oplus \bullet_2) \oplus \eta_{1,2}(\bullet_2 \oplus \rho_{2 \rightarrow 1}(\eta_{1,2}(\bullet_1 \oplus \bullet_2)))) \oplus \rho_{1 \rightarrow 2}(\eta_{1,2}((\bullet_1 \oplus \bullet_1) \oplus \bullet_2)))$$

is a 2-expression for the graph in Figure 1. The colors of the “constants”  $\bullet_i$  indicate which node in the figure they correspond to.

Comparing clique width and rank width, each parameter is bounded in terms of the other.<sup>21</sup> More precisely, for every graph  $G$ , it holds that

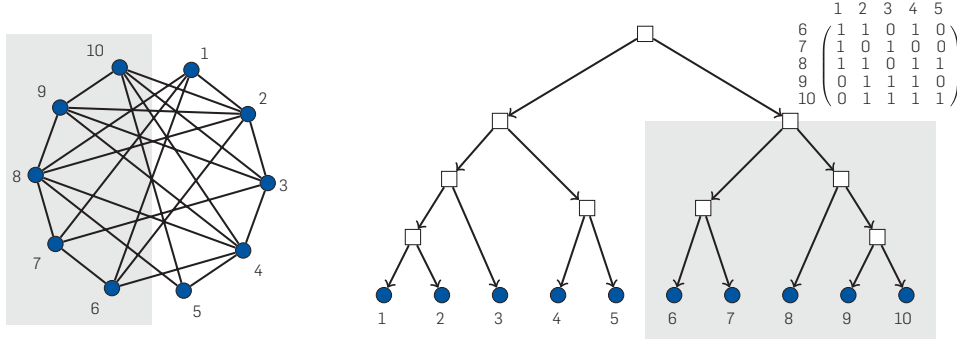
$$\text{rw}(G) \leq \text{cw}(G) \leq 2^{\text{rw}(G)+1} - 1. \quad (1)$$

Also, the rank width of a graph  $G$  is bounded by the tree width of  $G$ , denoted by  $\text{tw}(G)$ , by  $\text{rw}(G) \leq \text{tw}(G) + 1$ . Note that the tree width of a graph cannot be bounded in terms of its rank width. For example, the complete graph on  $n$  vertices  $K_n$  has rank width  $\text{rw}(K_n) = 1$  and tree width  $\text{tw}(K_n) = n - 1$ .

#### 3.1 The WL dimension of graphs of bounded rank width

The first main result of this work bounds the Weisfeiler-Leman dimension of graphs of rank width at most  $k$  by a linear function in  $k$ .

**Figure 4.** A graph  $G$  on the left and a potential rank decomposition of  $G$  of width 3 on the right. As an example, the matrix  $M([6, 7, 8, 9, 10], [1, 2, 3, 4, 5])$  is provided. Note that  $\rho_6([6, 7, 8, 9, 10]) = 3$ .



**THEOREM 8.** *The  $(3k + 4)$ -dimensional Weisfeiler-Leman algorithm identifies every graph of rank width at most  $k$ .*

Note that, by Equation (1), the same result and all of its consequences also hold for graphs of clique width at most  $k$ .

For the remainder of this section, we provide a high-level sketch of the proof of Theorem 8. We need to argue that, for every two nonisomorphic graphs  $G$  and  $H$  such that  $\text{rw}(G) \leq k$ , the  $(3k + 4)$ -dimensional Weisfeiler-Leman algorithm distinguishes between  $G$  and  $H$ . Exploiting the characterization of  $k$ -WL in terms of pebble games (see Corollary 7), it actually suffices to show that Spoiler wins the bijective  $(3k + 5)$ -pebble game  $\text{BP}_{3k+5}(G, H)$ .

To describe Spoiler's strategy, let us fix the input graphs  $G$  and  $H$  as well as some rank decomposition  $(T, \gamma)$  of  $G$  of width at most  $k$ . The basic idea for Spoiler is to play along the rank decomposition of  $G$  in a downward fashion in order to confine the "difference" between the two input graphs to smaller and smaller regions of the graphs. More precisely, for his strategy, Spoiler maintains sets  $X_G \subseteq V(G)$  and  $X_H \subseteq V(H)$  and vertices  $(a_G^1, \dots, a_G^p, b_G^1, \dots, b_G^p) \in X_G^{2p}$  and  $(a_H^1, \dots, a_H^p, b_H^1, \dots, b_H^p) \in X_H^{2p}$  where  $p \leq k$  such that there is no isomorphism  $\varphi: G[X_G] \rightarrow H[X_H]$  with  $\varphi(a_G^i) = a_H^i$  and  $\varphi(b_G^i) = b_H^i$  (where  $G[X_G]$  and  $H[X_H]$  denote the induced subgraphs on vertex sets  $X_G$  and  $X_H$ ). By gradually decreasing the size of the set  $X_G$ , Spoiler eventually wins the game since, as soon as  $|X_G| \leq 2k + 1$ , the entire set  $X_G$  can be pebbled.

During the play, the set  $X_G$  essentially corresponds to the sets  $\gamma(t)$  where  $t \in V(T)$  is a node of the rank decomposition of  $G$  where, as the play progresses, Spoiler moves down the rank decomposition to enforce a decrease in the size of the set  $X_G$ . Initially,  $X_G := V(G)$ ,  $X_H := V(H)$ ,  $p = 0$ , and  $t$  is the root node of the rank decomposition of  $G$ .

In order to force a descend in the rank decomposition, Spoiler's goal is to move to one of the two children  $t_1, t_2$  of  $t$  in the rank decomposition  $(T, \gamma)$ . To maintain the invariant described above, it is necessary to "split"  $\gamma(t_1)$  from  $\gamma(t_2)$  in the graph  $G$  in such a way that there are no dependencies left between the two sets. This enables Spoiler to confine the "difference" between the two input graphs to one of the two sides. To achieve the "split", we introduce the notion of a *split pair*.

Let  $G$  be a graph and  $X \subseteq V(G)$ . For  $v, w \in X$ , we define  $v \approx_X w$  if  $N(v) \cap \overline{X} = N(w) \cap \overline{X}$ . For  $v \in X$ , we define the vector

$\text{vec}_X(v) = (a_{v,w})_{w \in \overline{X}} \in \mathbb{F}_2^{\overline{X}}$  where  $a_{v,w} = 1$  if and only if  $vw \in E(G)$ . Note that  $v \approx_X w$  if and only if  $\text{vec}_X(v) = \text{vec}_X(w)$ . Moreover, for  $A \subseteq X$ , we define  $\text{vec}_X(A) = \{\text{vec}_X(v) \mid v \in A\}$ .

For any set of vectors  $S \subseteq \mathbb{F}_2^n$ , we denote by  $\langle S \rangle$  the linear space spanned by  $S$ . A set  $B \subseteq \mathbb{F}_2^n$  is a *linear basis* for  $\langle S \rangle$  if  $B$  is linearly independent and  $\langle B \rangle = \langle S \rangle$ .

**Definition 1.** Let  $G$  be a graph and  $X \subseteq V(G)$ . A pair  $(A, B)$  is a *split pair* for  $X$  if

1.  $A \subseteq X$  and  $B \subseteq \overline{X}$ ,
2.  $\text{vec}_X(A)$  forms a linear basis for  $\langle \text{vec}_X(X) \rangle$ , and
3.  $\text{vec}_{\overline{X}}(B)$  forms a linear basis for  $\langle \text{vec}_{\overline{X}}(X) \rangle$ .

Note that  $|A| = \rho_G(X)$  and  $|B| = \rho_G(X)$ . Also observe that if  $(A, B)$  is a split pair for  $X$  then  $(B, A)$  is a split pair for  $\overline{X}$ . As a special case, the pair  $(/, /)$  is defined to be a split pair for  $X = V(G)$ . An *ordered split pair* for  $X$  is a pair  $(\overline{a}, \overline{b}) = ((a_1, \dots, a_p), (b_1, \dots, b_p))$  such that  $(\{a_1, \dots, a_p\}, \{b_1, \dots, b_p\})$  is a split pair for  $X$ .

Now the central claim for split pairs is that pebbling an ordered split pair  $(\overline{a}, \overline{b})$  for  $X$  renders  $X$  to be "independent" from its complement. In order to visualize this independence, we make use of the color refinement algorithm. Toward this end, we define  $\chi^{(\overline{a}, \overline{b})} := \chi^{G, (\overline{a}, \overline{b})}$  to be the coloring computed by the color refinement algorithm on the graph  $G$  where all vertices  $a_1, \dots, a_p, b_1, \dots, b_p$  are *individualized* initially (i.e., each of the listed vertices is assigned a fresh color that is distinct from the colors of all other vertices). Moreover, we consider the notion of a *flip function* that allows us to flip edges between certain color classes of the coloring  $\chi^{(\overline{a}, \overline{b})}$ .

**Definition 2.** Let  $G = (V, E)$  be a graph and  $\chi: V \rightarrow C$  a coloring of the vertices. A *flip function* for  $G$  is a mapping  $f: C \times C \rightarrow \{0, 1\}$  such that  $f(c, c') = f(c', c)$  for all  $c, c' \in C$ .

For a flip function  $f$ , we define the flipped graph  $G^f = (V, E^f)$  where

$$E^f = \{vw \mid vw \in E \wedge f(\chi(v), \chi(w)) = 0\} \cup \{vw \mid v \neq w \wedge vw \notin E \wedge f(\chi(v), \chi(w)) = 1\}.$$

The following lemma gives the key tool to split  $X$  from its complement. For a graph  $G$  and a flip function  $f$ , let



$\text{Comp}(G, f) \subseteq 2^{V(G)}$  be the set of vertex sets of the connected components of  $G^f$ . Observe that  $\text{Comp}(G, f)$  forms a partition of the vertex set of  $G$ .

**LEMMA 9.** *Let  $G = (V, E)$  be a graph and  $X \subseteq V$ . Also let  $(\bar{a}, \bar{b})$  be an ordered split pair for  $X$ .*

*Then there is a flip function  $f$  for the graph  $G$  with coloring  $\chi^{(\bar{a}, \bar{b})}$  such that for every  $C \in \text{Comp}(G, f)$  it holds that  $C \subseteq X$  or  $C \subseteq \bar{X}$ .*

This process is also visualized in Figure 5 taking as input the example from Figure 1.

As applying a flip function to both input graphs changes neither the isomorphism problem nor the outcome of the bijective pebble game, Spoiler can simply pretend he is playing on  $G^f$  and  $H^f$  instead of  $G$  and  $H$ . For  $G^f$  and  $H^f$ , Spoiler can now restrict the game to a pair of connected components of the two graphs  $X_G \subseteq V(G)$  and  $X_H \subseteq V(H)$ , marking both components by placing a single pebble on them.

Now, in order to force a descend step in the rank decomposition, Spoiler pebbles split pairs for the sets  $\gamma(t_1)$  and  $\gamma(t_2)$  in order to render these sets to be independent from the rest of the graph. This allows Spoiler to track the difference between both input graphs to one of the two sets.

This almost completes the descend step. Indeed, the only remaining task for Spoiler is to remove all pebbles from previous steps to ensure that the number of required pebbles does not exceed  $3k + 5$ . In particular, the pebbles placed on the vertices of a split pair for  $\gamma(t)$  need to be removed.

However, it is not possible to simply remove these pebbles since, by the invariant described above, the sets  $X_G$  and  $X_H$  are only assured to be nonisomorphic as long as  $(a_G^1, \dots, a_G^p, b_G^1, \dots, b_G^p)$  as well as  $(a_H^1, \dots, a_H^p, b_H^1, \dots, b_H^p)$  are pebbled (these vertices need to be temporarily fixed to allow

for the applications of Lemma 9). In order to circumvent this problem, we introduce the notion of *nice* triples of split pairs.

**Definition 3.** Let  $G$  be a graph and  $X, X_1, X_2 \subseteq V(G)$  such that  $X = X_1 \cup X_2$  and  $X_1 \cap X_2 = \emptyset$ . Let  $(A, B)$  be a split pair of  $X$  and let  $(A_i, B_i)$  be split pairs for  $X_i$ ,  $i \in \{1, 2\}$ . We say that  $(A_i, B_i)$ ,  $i \in \{1, 2\}$ , are *nice* (with respect to  $(A, B)$ ) if

1.  $A \cap X_i \subseteq A_i$ , and
2.  $B_{3-i} \cap X_i \subseteq A_i$

for both  $i \in \{1, 2\}$ .

Naturally, a triple of ordered split pairs is nice if the underlying unordered triple of split pairs is nice.

Intuitively speaking, nice triples of split pairs enforce a significant overlap of all considered split pairs when performing the descend step. This enables Spoiler to remove the pebbles from the previous descend step without unpebbling any vertex in  $X_G$  or  $X_H$ . The following lemma guarantees the existence of nice triples of split pairs.

**LEMMA 10.** *Let  $G$  be a graph and  $X, X_1, X_2 \subseteq V(G)$  such that  $X = X_1 \cup X_2$  and  $X_1 \cap X_2 = \emptyset$ . Let  $(A, B)$  be a split pair of  $X$ . Then there are nice split pairs  $(A_i, B_i)$  for  $X_i$  for both  $i \in \{1, 2\}$ .*

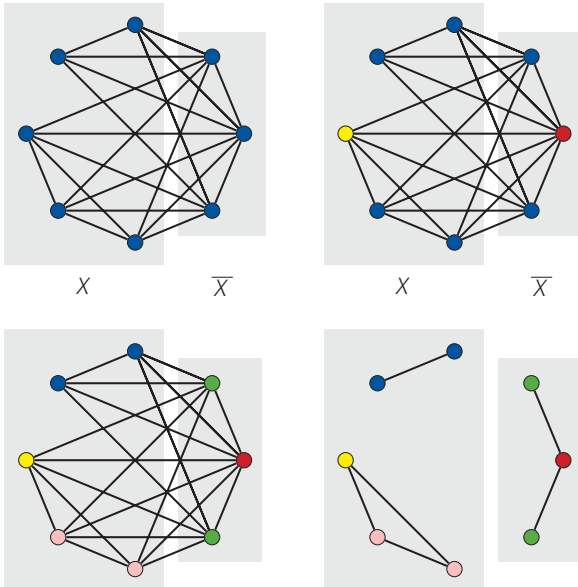
Hence, in all steps, Spoiler can play a nice triple of split pairs and simply remove any unwanted pebbles following the completion of a descend step while preserving the invariant given above. This completes the description of the strategy.

#### 4. DESCRIPTIVE COMPLEXITY

Descriptive complexity theory aims to characterize computational complexity in terms of logical definability, that is, relate computational resources such as space and time to language resources such as recursion mechanisms or quantifier alternation. The best-known result in the field is Fagin's theorem<sup>7</sup> stating existential second-order logic captures NP, that is, a property of graphs (or other structures) is decidable in nondeterministic polynomial time if and only if it is definable in existential second-order logic. The best-known open problem, going back to Chandra and Harel's influential paper<sup>3</sup> on database query languages and popularized by Gurevich,<sup>13</sup> asks whether there is a logic that captures PTIME (polynomial time). In the following, we give a very brief introduction to the relevant notions and results from descriptive complexity. For more background, we refer the reader to the existing literature (e.g., Ebbinghaus and Flum<sup>6</sup>).

In descriptive complexity theory, computational problems are typically modeled using finite relational structures, for example, graphs. A decision problem is simply a *property of structures*, that is, a class of structures that is closed under isomorphism. Closure under isomorphism is important, because it means that properties are isomorphism-invariant. For example, "a graph is connected" is a valid property, whereas "the first vertex has degree 3" is not, because a graph has no well-defined "first vertex". An *abstract logic*  $L$  is simply a set of sentences together with a satisfaction relation between structures and sentences. An abstract logic  $L$  *captures* PTIME if (i) it has a decidable syntax (the set of sentences is a decidable set of strings over a finite alphabet); (ii) it has a PTIME-decidable

**Figure 5.** In order to split  $X$  from its complement in the graph  $G$  (top left), we individualize a split pair (top right), compute the coloring  $\chi^{(\bar{a}, \bar{b})}$  using the color refinement algorithm (bottom left), and apply a suitable flip function to the graph (bottom right).



semantics, in the sense that there is an algorithm that, given a sentence  $\phi$  of  $L$  and a structure  $A$ , decides if  $A$  satisfies  $\phi$  in time polynomial in the size of  $A$ ; and (iii) every PTIME-decidable property  $\mathcal{P}$  of structures is definable in  $L$ , that is, there is a sentence of  $L$  that is satisfied by precisely those structures that have property  $\mathcal{P}$ . We say  $L$  captures PTIME on a class  $\mathcal{C}$  of structures if condition (iii) is satisfied for all properties  $\mathcal{P} \subseteq \mathcal{C}$ .

Although there are good reasons to define an abstract logic capturing PTIME this way (see Gurevich<sup>13</sup>), the reason we are interested in a logical characterization of PTIME is the hope that it might give us new insights into the mechanisms of efficient computation. For this, we are not so interested in an abstract logic, but rather in nice logical languages with few clearly understood operators. Logics that have proved to be natural and useful in this regard are extensions of classical first-order logic FO by fixed-point operators allowing it to formalize iterative or recursive definitions. *Inflationary fixed-point logic* FP is a robust extension of FO with a fixed-point operator that allows it to define a relation by iteratively adding tuples to the initially empty relation.

*Example 2.* The FP-sentence  $\text{conn}$  defined as

$$\forall x \forall y \text{ ifp} (Z \leftarrow z | z = x \vee \exists z' (Z(z') \wedge E(z', z)))(y).$$

expresses that a graph is connected. For any two nodes  $x, y$ , the  $\text{ifp}(\dots)$  operator computes the connected component  $X$  of  $x$  by adding  $x$  and then in each iteration all neighbors of nodes already in  $X$ . Then the sentence  $\text{conn}$  says that for all  $x, y$ ,  $y$  belongs to the connected component  $X$  of  $x$ .

Often, it is useful to also add rudimentary arithmetic to the logic, in particular, the ability to count. *Inflationary fixed-point logic with counting* FP+C is an extension of FP by the ability to speak about an initial segment of the natural numbers with basic arithmetic and a counting mechanism relating structures and numbers.

*Example 3.* The following FP+C-sentence defines the class of Eulerian graphs (i.e., graphs with a cyclic walk that traverses all edges, which are well-known to be exactly the connected graphs in which all vertices have even degree):

$$\text{eulerian} := \text{conn} \wedge \forall x \text{ even} (\#y E(x, y)).$$

Here  $\text{conn}$  is the sentence defined in the previous example,  $\#y E(x, y)$  defines the number of nodes  $y$  that are adjacent to node  $x$ , and  $\text{even}(n)$  is a formula expressing that  $n$  is an even number.

So FP and FP+C extend first-order logic, which provides the basic logical machinery, by two fundamental computational mechanisms: iteration and counting. At first sight, it may seem that counting the number of elements in a set, for example, the set of neighbors of a vertex in a graph, can be simulated by enumerating the elements in an iterative process implemented in FP. But this assumes that the elements of the set can be put in some order, and such an order is not always available in a structure. Logics operate in an isomorphism-invariant way, and there may be no isomorphism-invariant order on our set.

Although FP and FP+C fail to capture PTIME—this is easy to prove for FP and surprisingly hard for FP+C<sup>2</sup>—it has been

shown that they can express many natural PTIME-properties and they do capture PTIME restricted to a large variety of classes of structures. As a starting point, Immerman<sup>15</sup> and independently Vardi<sup>22</sup> proved that FP captures PTIME on the class of ordered structures, that is, structures that have one relation, which is a linear order of the universe. Note that on ordered structures we can simulate counting using iteration, and thus, FP and FP+C have the same expressive power. As soon as we leave ordered structures, we need the explicit counting mechanism of FP+C. In a series of papers that culminated in a monograph,<sup>8</sup> it was proved that FP+C captures PTIME on all graph classes that exclude a fixed graph as a minor. In particular, this includes the class of planar graphs and all graph classes of bounded tree width. Not too much is known beyond these classes, which all consist of sparse graphs. Indeed, looking at dense graphs, there are only very few examples of classes for which it is known that FP+C captures PTIME, most of which are fairly restricted classes of intersection graphs (e.g., interval graphs and permutation graphs). Our second main result, Theorem 4 stating that FP+C captures PTIME on all graph classes of bounded rank width, broadens the scope of these results into a new direction.

Like previous results of this type, Theorem 4 is proved using the framework of *definable canonization*.<sup>8,20</sup> Recall that by the Immerman-Vardi theorem,<sup>16,22</sup> FP captures polynomial time on the class of all ordered structures. A straightforward way of applying this theorem to a class  $\mathcal{C}$  of unordered structures is to define a linear order on this class: if there is a formula  $\text{ord}(x, y)$  of the logic FP that defines a linear order on all structures in  $\mathcal{C}$ , then FP still captures polynomial time on  $\mathcal{C}$ . Unfortunately, this observation is rarely applicable, because usually it is impossible to define linear orders. For example, it is impossible to define a linear order on a structure that has a nontrivial automorphism.

A more refined idea, known as *definable canonization*, is to define an *ordered copy* of the input structure. To implement this idea, FP+C is particularly well-suited, because it allows us to speak about an initial segment of the natural numbers that comes with a natural linear order. Technically, definable canonization is based on syntactical interpretations, which allow it to define a structure within another structure using logical formulas. (In database terminology, a syntactical interpretation would be a view.)

Without going into any details, let us just state that the main technical lemma toward the proof of Theorem 4 states that for all  $k$  the class of all graphs of rank width  $k$  admits FP+C-definable canonization. The idea to prove this lemma is to implement our canonization procedure for graphs of rank width  $k$  based on the  $(3k + 4)$ -dimensional Weisfeiler-Leman algorithm by a formula of the logic FP+C.

For all further details, we refer the reader to the full version of this article.<sup>10</sup>

## 5. CONCLUSION

In this article, we considered the isomorphism and canonization problem for graphs of bounded rank width. The first main result is that the Weisfeiler-Leman dimension of graphs of rank width at most  $k$  is at most  $3k + 4$ . This implies

that isomorphism testing and computing canonical forms for graphs of rank width at most  $k$  can be done in time  $n^{O(k)}$ .

Naturally, it would be desirable to further improve on the complexity of the two problems. Indeed, an important open question is whether isomorphism testing is also fixed-parameter tractable when parameterized by rank width (i.e., whether there is an algorithm testing isomorphism of graphs of rank width at most  $k$  in time  $f(k)n^c$  for some computable function  $f$  and an absolute constant  $c$ ).

The second main result states that, for every  $k \in \mathbb{N}$ , fixed-point logic with counting captures PTIME on the class of graphs of rank width at most  $k$ . □

## References

1. Babai, L. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the 48<sup>th</sup> Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18–21, 2016* (2016), D. Wichs and Y. Mansour, eds. ACM, 684–697.
2. Cai, J., Fürer, M., Immerman, N. An optimal lower bound on the number of variables for graph identifications. *Combinatorica* 4, 12 (1992), 389–410.
3. Chandra, A.K., Harel, D. Structure and complexity of relational queries. *J. Comput. Syst. Sci.* 1, 25 (1982), 99–128.
4. Courcelle, B., Makowsky, J.A., Rotics, U. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Syst.* 2, 33 (2000), 125–150.
5. Courcelle, B., Olariu, S. Upper bounds to the clique width of graphs. *Discrete Appl. Math.* 1–3, 101 (2000), 77–114.
6. Ebbinghaus, H., Flum, J. *Finite Model Theory*, 2<sup>nd</sup> edn. Springer Verlag, 1999.
7. Fagin, R. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation, SIAM-AMS Proceedings* (Vol. 7), R. Karp, ed. (1974), 43–73.
8. Grohe, M. Descriptive complexity, canonisation, and definable graph structure theory. Volume 47 of *Lecture Notes in Logic*. Cambridge University Press, 2017.
9. Grohe, M., Marx, D. Structure theorem and isomorphism test for graphs with excluded topological subgraphs. *SIAM J. Comput.* 1, 44 (2015), 114–159.
10. Grohe, M., Neuen, D. Canonisation and definability for graphs of bounded rank width. *CoRR*, abs/1901.10330, 2019.
11. Grohe, M., Schweitzer, P. Isomorphism testing for graphs of bounded rank width. In *IEEE 56<sup>th</sup> Annual Symposium on Foundations of Computer Science, FOCS 2015* (Berkeley, CA, USA, October 17–20, 2015), V. Guruswami, ed. IEEE Computer Society, 1010–1029.
12. Grohe, M., Schweitzer, P. Computing with tangles. *SIAM J. Discrete Math.* 2, 30 (2016), 1213–1247.
13. Gurevich, Y. Logic and the challenge of computer science. In *Current Trends in Theoretical Computer Science* (1988), E. Börger, ed. Computer Science Press, 1–57.
14. Hopcroft, J.E., Tarjan, R. Efficient planarity testing. *J. ACM*, 21 (1974), 549–568.
15. Immerman, N. Relational queries computable in polynomial time (extended abstract). In *Proceedings of the 14<sup>th</sup> ACM Symposium on Theory of Computing* (1982), 147–152.
16. Immerman, N. Languages that capture complexity classes. *SIAM J. Comput.* 4, 16 (1987), 760–778.
17. Immerman, N., Lander, E. Describing graphs: A first-order approach to graph canonization. In *Complexity Theory Retrospective: In Honor of Juris Hartmanis on the Occasion of His Sixtieth Birthday* (July 5, 1988), A.L. Selman, ed. Springer New York, New York, NY, 1990, 59–81.
18. Karp, R. On the complexity of combinatorial problems. *Networks*, 5 (1975), 45–68.
19. Luks, E.M. Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Comput. Syst. Sci.* 1, 25 (1982), 42–65.
20. Otto, M. Bounded variable logics and counting—A study in finite models. Volume 9 of *Lecture Notes in Logic*. Springer, 1997.
21. Oum, S., Seymour, P.D. Approximating clique-width and branch-width. *J. Comb. Theory, Ser. B* 4, 96 (2006), 514–528.
22. Vardi, M.Y. The complexity of relational query languages (extended abstract). In *Proceedings of the 14<sup>th</sup> Annual ACM Symposium on Theory of Computing* (San Francisco, California, USA, May 5–7, 1982), H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. H. Landweber, eds. ACM, 1982, 137–146.
23. Weisfeiler, B., Leman, A. The reduction of a graph to canonical form and the algebra which appears therein. *NTI Ser.* 2, 1968. Available at [https://www.iti.zcu.cz/wl2018/pdf/wl\\_paper\\_translation.pdf](https://www.iti.zcu.cz/wl2018/pdf/wl_paper_translation.pdf). English translation by G. Ryabov

Martin Grohe and Daniel Neuen  
 ([grohe,neuen]@informatik.rwth-aachen.de),  
 RWTH Aachen University,  
 Aachen, Germany.

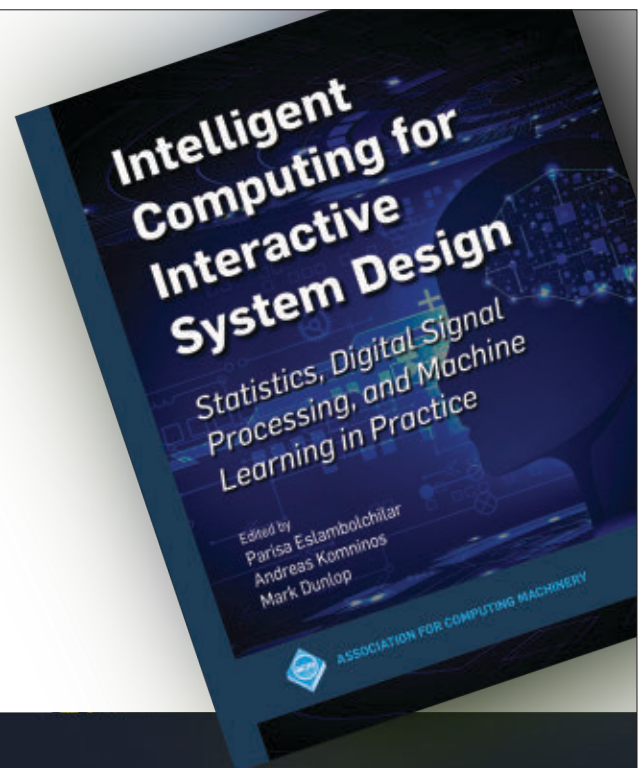
Copyright held by authors/owners.  
 Publication rights licensed to ACM.

# Intelligent Computing for Interactive System Design

*Statistics, Digital Signal Processing  
and Machine Learning in Practice*

Edited by  
**Parisa Eslambolchilar**  
**Mark Dunlop**  
**Andreas Komninos**

ISBN: 978-1-4503-9026-2  
 DOI: 10.1145/3447404  
<http://books.acm.org>  
<http://store.morganclaypool.com/acm>



 **ACM BOOKS**  
 Collection II

# Technical Perspective

## Robust Statistics Tackle New Problems

By Jacob S. Steinhardt

THE FOLLOWING PAPER represents the beginning of a long and productive line of work on robust statistics in high dimensions. While robust statistics has long been studied, going back at least to Tukey,<sup>6</sup> the recent revival centers on algorithmic questions that were largely unaddressed by the earlier statistical work.

Robust statistics centers on the question of how to extract information from data that may have been corrupted in some way. The most common form of robustness, also considered here, is robust to *outliers*: some fraction of the data has been removed and replaced with arbitrary, erroneous points. A familiar instance of robust statistics is using the median instead of the mean, since the median is less sensitive to extreme points, while in contrast a single overly large value could completely skew the mean.

There are several generalizations of the median to higher dimensions, each with their own robustness properties. These include the geometric median (which finds the point with minimum average Euclidean distance to the input dataset) and the Tukey median (which finds the “deepest” point in the dataset). However, the geometric median is fragile when the dimension is large, and the Tukey median is NP-hard to compute. In general, in the older statistics literature there were no estimators that were both efficient to compute and worked well in high-dimensional settings, although Donoho<sup>1</sup> and Donoho and Gasko<sup>2</sup> did construct several estimators toward this general end whose ideas are still relevant today.

This brings us to the current paper by Diakonikolas et al., which constructs efficient robust estimators for several problems including robust mean estimation of a Gaussian. For this problem, all prior robust estimators either had unfavorable error in high dimensions or could not be com-

puted in polynomial time (concurrent work by Lai et al.<sup>5</sup> also develops a polynomial-time estimator with slightly worse bounds; and earlier work by Klivans et al.<sup>4</sup> introduces related ideas for robust classification).


While there are a number of variations, there are two key techniques in this paper that underlie many of the results. The first, which has been most commonly used in subsequent work, is the “filtering algorithm.” The basic idea is to check if your distribution has some desirable property (such as bounded covariance), such that if it did we could perform robust recovery straightforwardly. If the property holds, then we are done; otherwise, we hope that a certificate of violation of the property will allow us to “filter out” bad points and then try again. In this way, we must eventually succeed: the desired property will eventually hold since there are only a finite number of bad points.

The second technique constructs an “approximate separation oracle” for a certain convex program, such that approximately solving the convex program leads to an accurate robust estimate. The twist is the convex program depends on the quantity that we are supposed to estimate in the first place! But this is where the approximate separation oracle comes in—it

**The following paper constructs efficient robust estimators for several problems including robust mean estimation of a Gaussian.**

turns out to be possible to construct this separation oracle without knowing the quantity itself. This particular technique has not been as widely employed as the filtering algorithm, but is spiritually related to later ideas such as the non-convex analysis of Zhu et al.<sup>7</sup>

Since these early papers, researchers have explored a number of algorithmic questions including robust classification and regression, robustness in list-decoding models, sum-of-squares proofs for robustness, and non-convex optimization for robust recovery. The recent work has also moved beyond algorithmic questions to unearth new statistical insights, such as new estimators that work under fairly weak statistical assumptions, connections to minimum distance functionals introduced by Donoho and Liu,<sup>3</sup> and robustness to new forms of corruptions defined by transportation metrics.

For those interested in learning more, there are now several tutorials, expositions, and courses on these topics, including the thesis of one of the authors, a related course at University of Washington, and my own lecture notes. 

### References

1. Donoho, D.L. Breakdown properties of multivariate location estimators. Ph.D. qualifying paper, 1982.
2. Donoho, D.L. and Gasko, M. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *Annals of Statistics* 20, 4 (1992), 1803–1827.
3. Donoho, D.L. and Liu, R.C. The “automatic” robustness of minimum distance functionals. *Annals of Statistics* 16, 2 (1988), 552–586.
4. Klivans, A.R., Long, P.M. and Servedio, R.A. Learning halfspaces with malicious noise. *J. ML Research* 10 (2009), 2715–2740.
5. Lai, K.A., Rao, A.B. and Vempala, S. Agnostic estimation of mean and covariance. *Foundations of Computer Science*, 2016.
6. Tukey, J.W. A survey of sampling from contaminated distributions. *Contributions to Probability and Statistics* 2 (1960), 448–485.
7. Zhu, B., Jiao, J. and Steinhardt, S. Robust estimation via generalized quasi-gradients. *arXiv preprint*, 2020.

**Jacob S. Steinhardt** is an assistant professor at the University of California Berkeley, CA, USA.

Copyright held by author.

# Robustness Meets Algorithms

By Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, and Alistair Stewart

## Abstract

**In every corner of machine learning and statistics, there is a need for estimators that work not just in an idealized model, but even when their assumptions are violated. Unfortunately, in high dimensions, being provably robust and being efficiently computable are often at odds with each other.**

**We give the first efficient algorithm for estimating the parameters of a high-dimensional Gaussian that is able to tolerate a constant fraction of corruptions that is independent of the dimension. Prior to our work, all known estimators either needed time exponential in the dimension to compute or could tolerate only an inverse-polynomial fraction of corruptions. Not only does our algorithm bridge the gap between robustness and algorithms, but also it turns out to be highly practical in a variety of settings.**

## 1. INTRODUCTION

Machine learning is filled with examples of estimators that work well in idealized settings but fail when their assumptions are violated. Consider the following illustrative example: We are given samples  $X_1, X_2, \dots, X_N$  from a one-dimensional Gaussian

$$\mathcal{N}(\mu, \sigma^2, x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

and our goal is to estimate its mean  $\mu$  and its variance  $\sigma^2$ . It is well-known that the empirical mean  $\hat{\mu}$  and empirical variance  $\hat{\sigma}^2$  are effective, which are defined as

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^N X_i \quad \text{and} \quad \hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^N (X_i - \hat{\mu})^2$$

In fact, these are examples of a more general paradigm within statistics called *maximum likelihood estimation*: When we know the distribution comes from some parametric family, we choose the parameters that are the most likely to have generated the observed data. In 1922, Ronald Fisher<sup>12</sup> formulated the maximum likelihood principle. It has many wonderful properties (under various technical conditions), such as converging to the true parameters as the number of samples goes to infinity, a property called *consistency*. Moreover, it has asymptotically the smallest possible variance among all unbiased estimators, a property called *asymptotic consistency*.

In 1960, John Tukey<sup>24</sup> challenged the conventional wisdom in parametric estimation by asking a simple question: Are there provably robust methods to estimate the parameters of a one-dimensional Gaussian? He showed that various estimators that were not asymptotically consistent (and had thus fallen out of favor) outperformed the maximum likelihood estimator when the data is not exactly Gaussian, but instead comes from a distribution that is close to being Gaussian. His paper launched the field of *robust statistics*<sup>15,13</sup>

that seeks to design estimators that behave well in a neighborhood around the true model. In one dimension, robust statistics prescribes that it is better to use the empirical median than the empirical mean. Similarly, it is better to use the empirical median absolute deviation (or any number of other estimators based on quantiles) than the empirical standard deviation. See Section 3.1.

Although there is an urgent need for provably robust estimators in virtually every application of machine learning, there is a major obstacle to directly applying ideas from robust statistics. The difficulty is that virtually all provably robust estimators are hard to compute in high dimensions. In this work, we are interested in the following family of questions:

**QUESTION 1.1.** *Let  $\mathcal{D}$  be a family of distributions on  $\mathbb{R}^d$ . Suppose we are given samples generated from the following process: First,  $m$  samples are drawn from some unknown distribution  $P$  in  $\mathcal{D}$ . Then, an adversary is allowed to arbitrarily corrupt an  $\varepsilon$ -fraction of the samples. Can we efficiently find a distribution  $P'$  in  $\mathcal{D}$  that is  $f(\varepsilon, d)$ -close, in total variation distance, to  $P$ ?*

Our most important example is the direct generalization of John Tukey's challenge<sup>24</sup> to higher dimensions: Is there a provably robust *algorithm* for estimating the parameters of a high-dimensional Gaussian? Without algorithmic considerations, robust statistics already provides prescriptions such as the *Tukey median*<sup>25</sup> and the *minimum volume enclosing ellipsoid*<sup>23</sup> for estimating the high-dimensional mean and covariance, respectively. However, the best-known algorithms for computing these estimates run in time that is exponential in the dimension. In fact, we are not aware of any moderate-sized datasets with dimension larger than six where these estimates have been successfully computed!

In contrast, there are other techniques that one might try. For example, instead of computing the Tukey median, we could compute the coordinate-wise median. This can obviously be done in polynomial time but encounters a different sort of difficulty: It turns out that by adding corruptions along a direction that is not-axis aligned, an adversary can badly compromise the estimator. Quantitatively, even if an adversary is only allowed to corrupt only an  $\varepsilon$ -fraction of the samples, they can force the estimator to find a Gaussian that is as far as  $\varepsilon\sqrt{d}$  in  $\ell_2$ -distance, implying total variation distance is close to one.

The original version of this paper is entitled "Robust Estimators in High Dimensions without the Computational Intractability" and was published in *Proceedings of the 57<sup>th</sup> Annual IEEE Symp. on Foundations in Computer Science*. A version was also published in *SIAM J. on Computing*, 2019.

The main meta-question behind our work is: Is it possible to design estimators that are both provably robust in high dimensions (i.e., they do not lose dimension-dependent factors in their robustness guarantees) and also efficiently computable? We will give the first provably robust and computationally efficient methods for learning the parameters of a high-dimensional Gaussian, as well as various other related models. In concurrent and independent work, Lai et al.<sup>20</sup> gave alternative algorithms albeit with weaker guarantees. We discuss their work in Section 1.3.

The types of questions we study here also have roots in computational learning theory. They are related to the agnostic learning model of Kearns et al.<sup>17</sup> where the goal is to learn a labeling function whose agreement with some underlying target function is close to the best possible, among all functions in some given class. In contrast, we are interested in an unsupervised learning problem, but it is also agnostic in the sense that we want to find the approximately closest fit from among our family of distributions. Within machine learning, these types of problems are also called estimation under model misspecification. The usual prescription is to use the maximum likelihood estimator, which is unfortunately hard to compute in general. Even ignoring computational considerations, the maximum likelihood estimator is only guaranteed to converge to the distribution  $P'$  in  $\mathcal{D}$  that is closest (in Kullback-Leibler divergence) to the distribution from which the observations are generated. This is problematic because such a distribution is not necessarily close to  $P$  at all.

More broadly, in recent years, there has been considerable progress on a variety of problems in this domain, such as algorithms with provable guarantees for learning mixture models, phylogenetic trees, hidden Markov models, topic models and independent component analysis. These algorithms are based on the method of moments and crucially rely on the assumption that the observations were actually generated by a model in the family. However, this simplifying assumption is not meant to be exactly true, and it is an important direction to explore what happens when it holds only in an approximate sense. Our work can be thought of as a first step toward relaxing the distributional assumptions in these applications, and subsequent work in algorithmic robust statistics has given new methodologies for robustly estimating higher moments under weaker assumptions about the uncorrupted distribution.<sup>5, 10, 14, 19</sup>

### 1.1. Our techniques

All of our algorithms are based on a common recipe. The first step is to answer the following easier question: Even if we were given a candidate hypothesis  $P'$ , how could we tell if it is  $\varepsilon$ -close in total variation distance to  $P$ ? The usual way to certify closeness is to exhibit a coupling between  $P$  and  $P'$  that marginally samples from both distributions, with the property that the samples are the same with probability  $1 - \varepsilon$ . However, we have no control over the process by which samples are generated from  $P$ , in order to produce such a coupling. And even then, the way

that an adversary decides to corrupt samples can introduce complex statistical dependencies.

We circumvent this issue by working with an appropriate notion of parameter distance, which we use as a proxy for the total variation distance between two distributions in the class  $\mathcal{D}$ . See Section 2.2. Various notions of parameter distance underlie various efficient algorithms for distribution learning in the following sense. If  $\theta$  and  $\theta'$  are two sets of parameters that define distributions  $P_\theta$  and  $P_{\theta'}$  in a given class  $\mathcal{D}$ , a learning algorithm often relies on establishing the following type of relation between the total variation distance  $d_{\text{TV}}(P_\theta, P_{\theta'})$  and the parameter distance  $d_p(\theta, \theta')$ :

$$\text{poly}(d_p(\theta, \theta'), 1/d) \leq d_{\text{TV}}(P_\theta, P_{\theta'}) \leq \text{poly}(d_p(\theta, \theta'), d) \quad (1)$$

Unfortunately, in our agnostic setting, we cannot afford for (1) to have any dependence on the dimension  $d$  at all. Any such dependence would appear in the error guarantee of our algorithm. Instead, the starting point of our algorithms is a notion of parameter distance that satisfies

$$\text{poly}(d_p(\theta, \theta')) \leq d_{\text{TV}}(P_\theta, P_{\theta'}) \leq \text{poly}(d_p(\theta, \theta')) \quad (2)$$

that allows us to reformulate our goal of designing robust estimators, with distribution-independent error guarantees, as the goal of robustly estimating  $\theta$  according to  $d_p$ . In several settings, the choice of the parameter distance is rather straightforward. It is often the case that some variants of the  $\ell_2$ -distance between the parameters work.

Given our notion of parameter distance satisfying (2), our main ingredient is an efficient method for robustly estimating the parameters. We provide two algorithmic approaches that are based on similar principles. Our first approach is fast and practical, requiring only approximate eigenvalue computations. Our second approach relies on convex programming, which has the advantage that it is possible to mix in different types of constraints (such as those generated by the sum-of-squares hierarchy) to tackle more complicated settings. Notably, either approach can be used to give all of our concrete learning applications with nearly identical error guarantees. In what follows, we specialize to the problem of robustly learning the mean  $\mu$  of a Gaussian whose covariance is promised to be the identity, which we will use to illustrate how both approaches operate. We emphasize that learning the parameters in more general settings requires many additional ideas.

Our first algorithmic approach is an iterative greedy method that, in each iteration, filters out some of the corrupted samples. In particular, given a set of samples  $S'$  that contains a large set  $S$  of uncorrupted samples, an iteration of our algorithm either returns the sample mean of  $S'$  or finds a *filter* that allows us to efficiently compute a set  $S'' \subset S'$  that is much closer to  $S$ . Note the sample mean  $\hat{\mu} = \sum_{i=1}^N (1/N)X_i$  (even after we remove points that are obviously outliers) can be  $\Omega(\varepsilon\sqrt{d})$ -far from the true mean in  $\ell_2$ -distance. The filter approach shows that either the sample mean is already a good estimate for  $\mu$  or else there is an elementary spectral test that rejects some of the corrupted

points and almost none of the uncorrupted ones. The crucial observation is that if a small number of corrupted points are responsible for a large change in the sample mean, it must be the case that many of the corrupted points are very far from the mean in some particular direction.

Our second algorithmic approach relies on convex programming. Here, instead of rejecting corrupted samples, we compute appropriate *weights*  $w_i$  for the samples  $X_i$ , so that the weighted empirical average  $\hat{\mu}(w) \triangleq \sum_{i=1}^N w_i X_i$  is close to  $\mu$ . We require the weights to be in the convex set  $\mathcal{C}_\tau$ , whose defining constraints are:

- (a)  $0 \leq w_i \leq \frac{1}{(1-\varepsilon)N}$  for all  $i$  and  $\sum_{i=1}^N w_i = 1$ , and
- (b)  $\left\| \sum_{i=1}^N w_i (X_i - \mu)(X_i - \mu)^T - I \right\|_2 \leq \tau$ .

We prove that *any* set of weights in  $\mathcal{C}_\tau$  yields a good estimate  $\hat{\mu}(w)$ . The catch is that the set  $\mathcal{C}_\tau$  is defined based on  $\mu$ , which is *unknown*. Nevertheless, it turns out that we can use the same types of spectral arguments that underlie the filtering approach to design an approximate separation oracle for  $\mathcal{C}_\tau$ . Combined with standard results in convex optimization, this yields our second algorithm for robustly estimating  $\mu$ .

The third and final ingredient is some new concentration bounds. In both of the approaches above, at best we hope that we can remove all of the corrupted points and be left with only the uncorrupted ones, and then use standard estimators (e.g., the empirical average) on them. However, an adversary could have removed an  $\varepsilon$ -fraction of the samples in a way that biases the empirical average of the remaining uncorrupted samples. What we need are concentration bounds that show for sufficiently large  $N$ , for samples  $X_1, X_2, \dots, X_N$  from a Gaussian with mean  $\mu$  and identity covariance, that every  $(1-\varepsilon)N$  set of samples produces a good estimate for  $\mu$ . In some cases, we can derive such concentration bounds by appealing to known concentration inequalities and taking a union bound. However, in other cases (e.g., concentration bounds for degree two polynomials of Gaussian random variables), the existing concentration bounds are not strong enough, and we need other arguments to prove what we need.

Finally, we briefly discuss how to adapt our techniques to robustly learn the covariance. Suppose the mean is zero and consider the following convex set  $\mathcal{C}_\tau$ , where  $\Sigma$  is the unknown covariance matrix:

- (a)  $0 \leq w_i \leq \frac{1}{(1-\varepsilon)N}$  for all  $i$  and  $\sum_{i=1}^N w_i = 1$ , and
- (b)  $\left\| \sum_{i=1}^N w_i X_i X_i^T - \Sigma \right\|_F \leq \tau$ .

Again, the constraints defining the convex set are based on the parameters of the distribution (this time, they use knowledge of  $\Sigma$ ). We design an approximate separation oracle for this unknown convex set by analyzing the spectral properties of the fourth moment tensor of a Gaussian. It turns out that our algorithms for robustly learning the mean when the covariance is the identity and robustly learning the covariance when the mean is zero can be combined to solve the general problem.

## 1.2. Our results

We give the first efficient algorithms for agnostically learning several important distribution classes with dimension-independent error guarantees. Our main result is an algorithm for robustly learning a high-dimensional Gaussian with an almost optimal error guarantee. Throughout this paper, we write  $N \geq \tilde{\Omega}(f(d, \varepsilon, \delta))$  when referring to our sample complexity, to signify that our algorithm works if  $N \geq C f(d, \varepsilon, \delta) \text{polylog}(f(d, \varepsilon, \delta))$  for a large enough universal constant  $C$ .

**THEOREM 1.2.** *Let  $\mu, \Sigma$  be arbitrary and unknown, and let  $\varepsilon > 0$  be given. There is a polynomial time algorithm that given an  $\varepsilon$ -corrupted set of  $N$  samples from  $\mathcal{N}(\mu, \Sigma)$  with  $N \geq \tilde{\Omega}\left(\frac{d}{\varepsilon}\right)$  produces  $\hat{\mu}$  and  $\hat{\Sigma}$  so that with probability 0.99, we have*

$$d_{\text{TV}}(\mathcal{N}(\mu, \Sigma), \mathcal{N}(\hat{\mu}, \hat{\Sigma})) \leq O(\varepsilon \log^{3/2}(1/\varepsilon)).$$

In later work,<sup>5</sup> we improved the sample complexity to  $N \geq \tilde{\Omega}\left(\frac{d}{\varepsilon}\right)$ , which is optimal up to constant factors even when there are no corruptions. Moreover, it was observed in Diakonikolas et al.<sup>6</sup> that the error guarantee of these algorithms can be improved to  $O(\varepsilon \log(1/\varepsilon))$ , which is the best possible for statistical query algorithms.<sup>9</sup>

Beyond robustly learning a high-dimensional Gaussian, we give the first efficient robust learning algorithms with dimension-independent error guarantees for various other statistical tasks, such as robust estimation of a binary product distribution, robust density estimation for mixtures of any constant number of spherical Gaussians, and mixtures of two binary product distributions (under some natural balanced-ness condition). We emphasize that obtaining these results requires additional conceptual and technical ingredients. We defer a description of these results to the full version of our paper.

## 1.3. Related work

In concurrent and independent work, Lai et al.<sup>20</sup> also study high-dimensional robust estimation. Their results hold more generally for distributions with bounded moments, but our guarantees are stronger (and optimal up to polylogarithmic factors) for the fundamental problem of robustly learning a Gaussian.

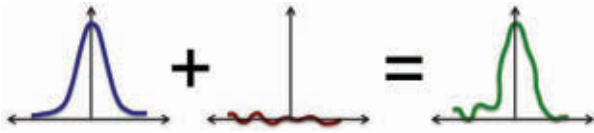
After both our and their work, there has been a flurry of activity in the area, such as algorithms for robust list learning when the fraction of corruptions is greater than a half,<sup>3</sup> algorithms for sparse mean estimation whose sample complexity is sublinear in the dimension,<sup>2</sup> lower bounds against statistical query algorithms,<sup>9</sup> and extensions to other generative models with weaker moment conditions<sup>14, 19</sup> and various supervised learning problems.<sup>22, 7</sup> An overview of recent developments in the area can be found in Diakonikolas and Kane.<sup>8</sup> We also note that spectral techniques for robust learning, which are relatives of our algorithms, appeared in earlier work.<sup>18, 1</sup> These works employed a “hard” filtering step (for a supervised learning problem), which only removes outliers and consequently leads to errors that scale logarithmically with the dimension.

## 2. PRELIMINARIES

### 2.1. Problem setup

Formally, we will work in the following corruption model:  
**DEFINITION 2.1.** For a given  $\varepsilon > 0$  and an unknown distribution  $P$ , we say that  $S$  is an  $\varepsilon$ -corrupted set of samples from  $P$  of size  $N$  if  $S = G \cup E \setminus S_r$ , where  $G$  is a set of  $N$  independent samples from  $P$ ,  $S_r \subset G$ , and  $E$  and  $S_r$  satisfy  $|E| = |S_r| \leq \varepsilon N$ .

In other words, a set of samples is  $\varepsilon$ -corrupted if an  $\varepsilon$ -fraction of the samples has been arbitrarily changed, which we can think of as a two-step process: first the adversary removes the samples in  $S_r$  and then adds in its own arbitrarily chosen data points  $E$ . Note that the  $\varepsilon$ -corruption model is a



strong model of corruption, and it gives it more power than other classical notions of corruption, such as Huber's contamination model. We can visualize how the adversary can change the probability density function of  $P$  as follows:

Here, the blue curve is the original density function, and the green curve is the new density function, which is merely approximately close to  $P$ . The regions where the blue curve lies above the green curve are places where the adversary has deleted samples, and the regions where the green curve is above the blue curve are places where it has injected samples. In fact, the true process is even more complicated because if an adversary first inspects the samples and then decides what to corrupt, even the samples it has not corrupted are no longer necessarily independent.

It turns out that this model has very close connections to a natural measure of distance between distributions, namely, total variation distance:

**DEFINITION 2.2.** Given two distributions  $P, Q$  over  $\mathbb{R}^d$  with probability density functions  $p, q$ , respectively, the total variation distance between  $P$  and  $Q$  is given by

$$d_{\text{TV}}(P, Q) = (1/2) \int_{\mathbb{R}^d} |p(x) - q(x)| dx.$$

The reason for this connection is as follows:

**FACT 2.3.** Let  $P, Q$  be two distributions with  $d_{\text{TV}}(P, Q) = \varepsilon$ . Let  $S$  be  $N$  i.i.d. samples from  $Q$ . Then, with probability at least  $1 - \exp(-\Omega(\varepsilon n))$ ,  $S$  can be viewed as a set of  $(1 + o(1))\varepsilon$ -corrupted samples from  $P$ .

This means that, up to subconstant factors in the fraction of corrupted points, learning from corrupted data is at least as hard as learning a distribution  $P$  from samples, if all we get are samples from some other distribution, which is  $\varepsilon$ -close to it in total variation distance. This fact immediately implies that if we are given  $\varepsilon$ -corrupted samples from  $P$ , the best we can generally hope for is to recover some  $\hat{P}$  so that  $d_{\text{TV}}(P, \hat{P}) = \Theta(\varepsilon)$ . As we shall see, it is often possible to match this lower bound (up to logarithmic factors).

### 2.2. Connections to parameter distance

In this paper, our focus is on *robust Gaussian estimation*, when  $P = \mathcal{N}(\mu, \Sigma)$  is a Gaussian distribution. That is, given a set of  $N$   $\varepsilon$ -corrupted samples from some unknown Gaussian distribution  $P$ , the goal is to output a Gaussian distribution  $\hat{P}$  such that  $d_{\text{TV}}(P, \hat{P})$  is small. As it turns out, learning a Gaussian in total variation distance is closely tied to learning the parameters of the distribution, in the natural affine-invariant measure. This is captured by the following two lemmata. Throughout this paper, we will say that  $f(x) \lesssim g(x)$  if  $f(x) \leq C g(x)$  for all  $x$  and some universal constant  $C$ . We also let  $\|A\|_F$  denote the Frobenius norm of a matrix  $A$ .

**LEMMA 2.4.** For any  $\mu, \mu' \in \mathbb{R}^d$ , we have

$$\min(\|\mu - \mu'\|_2, 1) \lesssim d_{\text{TV}}(\mathcal{N}(\mu, I), \mathcal{N}(\hat{\mu}, I)) \lesssim \|\mu - \mu'\|_2.$$

**LEMMA 2.5.** For any full-rank positive semidefinite matrices  $\Sigma, \Sigma'$ , we have

$$\min(\|\Sigma - \Sigma'\|_{\Sigma}, 1) \lesssim d_{\text{TV}}(\mathcal{N}(0, \Sigma), \mathcal{N}(0, \Sigma')) \lesssim \|\Sigma - \Sigma'\|_{\Sigma},$$

where  $\|A\|_{\Sigma} \triangleq \|\Sigma^{-1/2} A \Sigma^{-1/2}\|_F$ .

The first lemma states that if the covariances are both the identity, then the total variation distance between the two Gaussians is essentially the  $\ell_2$ -distance between the means of the Gaussians, except when the means are far apart. Note that total variation distance is always at most 1, and so when the means are very far apart, we only get a constant lower bound.

The second lemma is similar: It says that if both means are zero, then the total variation distance is captured by the Frobenius norm distance between the covariances, but “pre-conditioned” by one of the covariances. This is simply a high-dimensional analog of the fact that in one dimension, if we wish to get a meaningful approximation to the variance of a Gaussian, we need to learn it to multiplicative error.

## 3. ROBUST ESTIMATION

### 3.1. Univariate robust estimation

For the sake of exposition, we begin with robust univariate Gaussian estimation. A first observation is that the empirical mean is *not* robust: even changing a *single sample* can move our estimate by an arbitrarily large amount. To see this, let  $\tilde{\mu}$  be the empirical mean of the dataset before corruptions, and let  $\hat{\mu}$  be the empirical mean after increasing the value of the sample  $X_1$  by some amount  $t$ . Although standard concentration arguments imply that  $|\tilde{\mu} - \mu|$  is small, we have that  $|\hat{\mu} - \tilde{\mu}| = t/N$ , which we can make arbitrarily large with our choice of  $t$ . Fortunately, we describe a simple approach based on order statistics, which will allow us to estimate both the mean and the variance, even when a constant fraction of our dataset has been corrupted.

The most well-known robust estimator for the mean of a Gaussian is the *median*. More precisely, we let

$$\hat{\mu}_{\text{med}} = \text{median}(X_i).$$



Similarly, as an estimate for the standard deviation, we can consider a rescaling of the *median absolute deviation* (MAD), letting

$$\hat{\sigma}_{\text{mad}} = \frac{1}{\Phi^{-1}(3/4)} \cdot \text{median}(|X_i - \hat{\mu}|),$$

where  $\Phi^{-1}$  is the inverse of the Gaussian cumulative distribution function. The rescaling is required to make the MAD a consistent estimator for the standard deviation. The median and MAD allow us to robustly estimate the underlying Gaussian:

**THEOREM 3.1.** *Given a set of  $N \geq \Omega\left(\frac{\log 1/\epsilon}{\epsilon^2}\right)$   $\epsilon$ -corrupted samples from  $\mathcal{N}(\mu, \sigma^2)$ , with probability at least  $1 - \delta$ , we have*

$$d_{\text{TV}}\left(\mathcal{N}(\mu, \sigma^2), \mathcal{N}(\hat{\mu}_{\text{med}}, \hat{\sigma}_{\text{mad}}^2)\right) \leq C\epsilon$$

for a universal constant  $C$ .

This estimator is the best of all possible worlds. It is provably robust. It can be computed efficiently. In fact, it also achieves the information-theoretically optimal sample complexity. The median and MAD are examples of robust estimators based on *order statistics*. There are other provably robust estimators based on *winsorizing*.<sup>13</sup>

### 3.2. Natural multivariate approaches that fail

There are many natural approaches for generalizing what we have learned in the one-dimensional case to the high-dimensional case. But as we will see, there is a tension between being provably robust and computationally efficient. First, consider a coordinate-by-coordinate approach where we robustly estimate the mean along each coordinate direction, and concatenate the  $d$  univariate estimates into an estimate for the  $d$ -dimensional mean vector. Although this achieves error  $\Theta(\epsilon)$  in each direction's subproblem, combining the estimates results in an  $\ell_2$  error of  $\Omega(\epsilon\sqrt{d})$ . In high-dimensional settings, this gives vacuous bounds on the total variation distance except for vanishingly small values of  $\epsilon$ .

Alternatively, one could attempt to extend the median-based estimator to multivariate settings. Although the same definition of the median does not apply in more than one dimension, there are many ways to generalize it. One such generalization is the *Tukey median*,<sup>25</sup> proposed specifically for the problem of robust estimation. The Tukey median of a dataset is the point (not necessarily in the dataset) that maximizes the minimum number of points on one side of any half-space through the point. Although this achieves the desired  $O(\epsilon)$  accuracy, it is unfortunately NP-hard to approximate on worst-case datasets.<sup>16</sup> Another multivariate notion of median is the *geometric median*, the point that minimizes the sum of  $\ell_2$ -distances to points in the dataset. Although this is efficiently computable in polynomial time, it unfortunately also can be shown to incur  $\Omega(\epsilon\sqrt{d})$  error.<sup>20</sup>

All the approaches mentioned so far have one of the following drawbacks:

1. The optimization problem is NP-hard, making it intractable in settings of even moderate dimensionality.

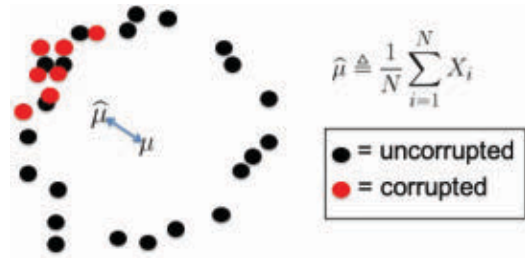
2. A large dimension-dependent factor appears in the error, resulting in very weak accuracy guarantees in high-dimensional settings.

At least one of these issues persists in all previously-known approaches, and either one would preclude realizable multivariate robust estimation. As few more examples, tournament-based hypothesis selection methods give accurate results, but are not computationally efficient. Alternatively, one could consider a pruning-based argument, which removes all points that are too far from the rest of the dataset. This is computationally efficient, but we again incur error of  $\Omega(\epsilon\sqrt{d})$ .

The primary contribution of our main result is a method that avoids both these issues simultaneously, providing an algorithm that is computationally efficient and does not lose a dimension-dependent factor in the accuracy.

### 3.3. Robust mean estimation

To offer some insight into why things go wrong in multivariate settings, we delve a bit deeper into the pruning-based approach. For the time being, we restrict our attention to Gaussians with identity covariance. It is well-known that,



given a dataset generated according to a  $d$ -dimensional spherical Gaussian distribution, all the data points will be tightly concentrated at a distance  $\Theta(\sqrt{d})$  from the mean. Thus, we can think about the distribution as being concentrated on a thin spherical shell, as depicted here:

A smart adversary can place all his corruptions within the shell too, in such a way that they move the empirical mean by as much as  $\epsilon\sqrt{d}$  in  $\ell_2$ -distance. This demonstrates an intrinsic limitation of any algorithm, which only looks *locally* for corruptions—as a result, any effective algorithm must remove points based on *global* properties of the dataset.

This is captured in the following geometric lemma:

**LEMMA 3.2.** *Let  $\epsilon \in (0, 1/2)$ . Let  $S$  be an  $\epsilon$ -corrupted set of points from  $\mathcal{N}(\mu, I)$  of size at least  $\Omega(d/\epsilon^2)$ . Let  $\hat{\mu}$ ,  $\hat{\Sigma}$  denote the empirical mean and covariance of  $S$ , that is,*

$$\hat{\mu} = \frac{1}{N} \sum_{X \in S} X, \quad \hat{\Sigma} = \frac{1}{N} \sum_{X \in S} (X - \hat{\mu})(X - \hat{\mu})^\top.$$

Then, with probability at least 0.99, we have:

$$\|\mu - \hat{\mu}\|_2 \lesssim \epsilon\sqrt{\log 1/\epsilon} + \sqrt{\epsilon\|\hat{\Sigma} - I\|_2}. \quad (3)$$

This lemma is a slight rephrasing of Lemma 4.15 in Diakonikolas et al.<sup>4</sup> At a high level, Lemma 3.2 states that if

the true mean and the empirical (and potentially corrupted) mean are far apart, then the empirical variance must be noticeably different along some direction. Thus, the spectral norm of  $\hat{\Sigma}$  can be used to certify that our estimate  $\hat{\mu}$  is close to the true mean in the sense that

$$\|\bar{\Sigma} - I\|_2 = O(\varepsilon \log 1/\varepsilon) \Rightarrow \|\mu - \hat{\mu}\|_2 = O(\varepsilon \sqrt{\log 1/\varepsilon}).$$

On the other hand, when the empirical mean has been corrupted, Lemma 3.2 gives us a way to algorithmically make progress. It isolates a specific direction—namely, the top eigenvector of  $\hat{\Sigma} - I$ —in which the corrupted points must contribute a lot. Both of the algorithms we describe use information gleaned about the corruptions from the empirical moments in somewhat different ways.

**Filtering approach.** The filtering approach works by removing points from the dataset, using the above intuition. It proceeds as follows:

- (a) Compute the top eigenvalue  $\lambda$  and eigenvector  $v$  of  $\hat{\Sigma}$ .
- (b) If  $\lambda$  is sufficiently small, terminate and output  $\hat{\mu}$ .
- (c) Otherwise, compute  $\tau_i = \langle X_i - \hat{\mu}, v \rangle^2$ , and for an adaptively chosen threshold  $T$ , remove all  $X_i$  so that  $\tau_i > T$ , and repeat.

If this is done carefully, then Lemma 3.2 guarantees that we always throw out many bad points compared to the number of good points we throw out. See Diakonikolas et al.<sup>4</sup> for a detailed description of how the threshold is chosen. To make this formal, for any two sets  $A, B$ , define  $\Gamma(A, B) \triangleq |A \Delta B|/|A|$ , which measures the relative size of the symmetric difference compared to the size of  $A$ . Then, we have the following guarantee for the filter:

**LEMMA 3.3 (INFORMAL).** *Let  $S = G \cup E \setminus S_r$  be an  $\varepsilon$ -corrupted set of points from  $\mathcal{N}(\mu, I)$  of size at least  $\tilde{\Omega}(d/\varepsilon^2)$ . Then, with probability at least 0.99 and after a simple preprocessing step, the filter satisfies the following property: Given any  $S' \subseteq S$  satisfying  $\Gamma(G, S') \leq 2\varepsilon$ , the filter either*

- (a) *outputs  $\hat{\mu}$  so that  $\|\mu - \hat{\mu}\|_2 \leq O(\varepsilon \sqrt{\log 1/\varepsilon})$ , or*
- (b) *outputs  $T$  so that  $\Gamma(G, T) \leq \Gamma(G, S') - \varepsilon/\alpha$ , where  $\alpha = d \log(d/\varepsilon) \log(d \log(d/\varepsilon))$ .*

Note that  $\Gamma(G, S) \leq 2\varepsilon$  initially. Now applying Lemma 3.3 we can guarantee that the procedure terminates after at most  $O(\alpha)$  iterations. And when we terminate, again by Lemma 3.3, we are guaranteed to output  $\hat{\mu}$ , which is close to the true mean. As described, each application of the filter would require computing the top eigenvector of a  $d \times d$  matrix, which would be prohibitively slow. However, it turns out that a rough approximation to the top eigenvector suffices for the correctness of the filter algorithm, and thus each iteration can be performed in nearly-linear time via an approximate power method.

**Convex programming approach.** The second approach uses the intuition behind Lemma 3.2 somewhat differently. Instead of trying to directly remove all of the

consequential outliers, we seek to iteratively decrease their influence. Let  $S = G \cup E \setminus S_r$  be an  $\varepsilon$ -corrupted set of points. For each point  $X_i \in S$ , we associate a nonnegative weight  $w_i$ . Ideally, we would want these weights to be uniform over  $S \setminus E$ , and zero otherwise. *A priori* the only thing we know about  $S \setminus E$  is that it has size at least  $(1 - \varepsilon)N$ . Consequently, a natural constraint to put on the weights  $w$  is that they must lie within the convex hull of the set of weights that are uniform on sets of size  $(1 - \varepsilon)N$ . This is the following set:

$$W_{n,\varepsilon} \triangleq \left\{ w \in \mathbb{R}^N : \sum_{i=1}^N w_i = 1, w_i \in \left[ 0, \frac{1}{(1-\varepsilon)N} \right] \right\}. \quad (4)$$

Given this, one can show that a slight extension of Lemma 3.2 implies that it suffices to find a set of weights  $w \in W_{n,\varepsilon}$  so that the empirical distribution over  $S$  with these weights is spectrally close to the identity after centering at  $\mu$ . To make this more formal, for any  $w \in W_{N,\varepsilon}$ , let

$$\mu(w) = \sum_{i=1}^N w_i X_i, \quad \Sigma(w) = \sum_{i=1}^N w_i (X_i - \mu(w))(X_i - \mu(w))^\top$$

be the mean and covariance of the empirical distribution over  $S$ , with weights  $w$ . Define also

$$M(w) = \sum_{i=1}^N w_i (X_i - \mu)(X_i - \mu)^\top$$

to be the empirical covariance, except we center at the true mean of the unknown distribution. Note that  $M(w)$  is a linear function of  $w$ , and so in particular, it is a convex function. Then, it suffices to solve the following convex problem:

$$\text{Find } w \in W_{N,\varepsilon} \text{ s.t. } \|M(w) - I\|_2 \leq C\varepsilon \log 1/\varepsilon, \quad (5)$$

where  $C > 0$  is some universal constant. If  $w \in W_{N,\varepsilon}$  satisfies (5), then one can show that  $\mu(w)$  is close to  $\mu$  with high probability, provided that  $N = \Omega(d/\varepsilon^2)$ .

There is an obvious difficulty in solving (5). Namely, the description of (5) requires knowledge of  $\mu$ , the parameter that we wish to estimate! Luckily, we can still construct a separation oracle for (5), which will suffice for computing a solution. In particular, given  $w \in W_{N,\varepsilon}$ , we want an algorithm that

- (a) if  $w$  satisfies (5), outputs YES and
- (b) otherwise, outputs a hyperplane  $\ell$  so that  $\ell(w) > 0$  but  $\ell(w') < 0$  for all  $w'$  satisfying (5).

First, note that if we knew  $\mu$ , then constructing such an oracle would be straightforward. We simply compute the largest eigenvalue  $\lambda$  of  $M(w) - I$  in magnitude, and its associated eigenvector  $v$ . If  $|\lambda| < C\varepsilon \log 1/\varepsilon$ , then output YES. If not, observe that the following is a separating hyperplane for  $w$ :

$$\sigma \cdot \left( \sum_{i=1}^N w_i \langle v, X_i - \mu \rangle^2 - 1 \right) > |\lambda|, \quad (6)$$

where  $\sigma$  is the sign of  $\lambda$ .

Now we need to remove the assumption that we know  $\mu$ . The key insight is that Lemma 3.2 allows us to substitute the top eigenvalue of  $\Sigma(w) - I$  in magnitude and its associated eigenvector for  $\lambda$  and  $v$ , respectively, and  $\mu(w)$  for  $\mu$  in (6). At a high level, this is because if  $\mu(w)$  is close to  $\mu$ , then  $\Sigma(w)$  is very

close to  $M(w)$ . On the other hand, if  $\mu(w)$  is far from  $\mu$ , then Lemma 3.2 guarantees that the shift caused by centering at  $\mu(w)$  rather than  $\mu$  is overshadowed by the large eigenvalues of  $M(w) - I$ .

### 3.4. Robust covariance estimation

The same geometric intuition that underlies our algorithms for robustly learning the mean also forms the basis for our algorithms for robustly learning the covariance. This time, we momentarily restrict our attention to Gaussians with zero mean. In the case of robust mean estimation, Lemma 3.2 states that a shift in the first moment caused by a small fraction of outliers causes a noticeable deviation in the second moment. It turns out that the same principles work for robustly learning the covariance, we just need to use higher moments. In particular, if we want to detect when the empirical second moment has been compromised by a small fraction of outliers, there must be some evidence in the fourth moment. However, making this rigorous is technically involved.

At a high level, the main difficulty is that in the case of robust mean estimation, we know the *structure* of the second moment, even if we do not know the mean. Namely, we assume that the covariance is the identity. However, the structure of the fourth moment depends heavily on the unknown covariance, and as a result, it is nontrivial to formulate the proper analog of Lemma 3.2 for this setting.

Fortunately, the relationship between the second moment and the fourth moment of a Gaussian follows a predictable formula, as a special case of *Isserlis' theorem*. For any vector  $v \in \mathbb{R}^d$ , let  $v \otimes v \in \mathbb{R}^{d^2}$  denote the tensor product of  $v$  with itself. Similarly, for any matrix  $M \in \mathbb{R}^{d \times d}$ , let  $M^{\otimes 2} \in \mathbb{R}^{d^2 \times d^2}$  be its tensor product with itself. Finally, let  $M^p \in \mathbb{R}^{d^2}$  be the  $d^2$ -dimensional vector that comes from flattening  $M$  into a vector. Then, the key identity is the following: for any covariance matrix  $\Sigma$ , we have

$$\mathbb{E}_{X \sim \mathcal{N}(0, \Sigma)} [(X \otimes X)(X \otimes X)^\top] = 2 \cdot \Sigma^{\otimes 2} + (\Sigma^b)(\Sigma^b)^\top. \quad (7)$$

Consider the case where the unknown covariance  $\Sigma$  is well-conditioned, so that it suffices to learn  $\Sigma$  to small error in Frobenius norm. Let  $\{X_1, \dots, X_N\}$  be an  $\varepsilon$ -corrupted dataset and set  $Y_i = X_i \otimes X_i$  for all  $i \in [N]$ . If  $Y_i$  is uncorrupted, then  $\mathbb{E}[Y_i] = \Sigma^b$ , so recovering  $\Sigma$  in Frobenius norm exactly corresponds to learning the mean of the uncorrupted  $Y_i$  to small error in  $\ell_2$  norm. Moreover, by (7), the covariance of the uncorrupted  $Y_i$  is  $2 \cdot \Sigma^{\otimes 2}$ .

Thus learning the covariance reduces to a complicated variant of the mean estimation problem, where the covariance depends on the unknown mean, but in a structured way. The relationship between them is sufficiently nice so that if the empirical mean of the  $Y_i$  is corrupted by outliers, then this still manifests as a large eigenvalue of the empirical covariance of the  $Y_i$ . This allows us to formulate a more sophisticated analog of Lemma 3.2 for this setting (see Claim 4.29 in Diakonikolas et al.<sup>4</sup>). By then, leveraging this geometric structure, we can then devise generalizations of the filtering and the convex programming approaches to robustly learn the covariance.

### 3.5. Assembling the general algorithm

At this point, we have designed efficient algorithms to solve two important subcases of our general problem. Specifically, we can

- (1) robustly estimate  $\mathcal{N}(\mu, I)$ , up to error  $O(\varepsilon \sqrt{\log(1/\varepsilon)})$  in total variation distance and
- (2) robustly estimate  $\mathcal{N}(0, \Sigma)$ , up to error  $O(\varepsilon \log(1/\varepsilon))$  in total variation distance.

In fact, we can combine these primitives into an algorithm that works in the general case when both  $\mu$  and  $\Sigma$  are unknown. The first observation is that we can use the doubling trick (even in the presence of noise) to zero out the mean. In particular, given two independent samples  $X_1$  and  $X_2$  from a distribution that is  $\varepsilon$ -close to a Gaussian  $\mathcal{N}(\mu, \Sigma)$ , their difference  $X_1 - X_2$  will be  $2\varepsilon$ -close in distribution to  $\mathcal{N}(0, 2\Sigma)$ .

The second observation is that, given an estimate  $\hat{\Sigma}$  of the covariance, we can approximately whiten our dataset. After applying the transformation  $(\hat{\Sigma})^{-1/2}$  to our data, we get noisy samples from

$$\mathcal{N}(\hat{\Sigma}^{-1/2} \mu, \hat{\Sigma}^{-1/2} \Sigma \hat{\Sigma}^{-1/2}).$$

This *almost* fits into the setting where just the mean is unknown, because the resulting covariance matrix is close to (but not exactly equal to) the identity matrix. Fortunately, we can exploit the robustness of our algorithms to handle this error, because the data is generated from a distribution that is  $O(\varepsilon \log(1/\varepsilon))$ -close to a Gaussian with identity covariance. Putting the pieces together, we get an error guarantee of  $O(\varepsilon \log^{3/2}(1/\varepsilon))$ . The overall algorithm is described in Algorithm 1. We will use  $\mathcal{X}$  and  $\mathcal{Y}$  to denote a set of samples that are fed into various subroutines.

#### Algorithm 1 Algorithm for robustly learning a Gaussian

- 1: **function** RECOVERROBUSTGAUSSIAN( $\varepsilon, X_1, \dots, X_{2N}$ )
- 2:     Let  $\hat{\Sigma} \leftarrow \text{LEARNCOVARIANCE}(4\varepsilon, \mathcal{X})$  for  
 $\mathcal{X} = (X_1 - X_2) / \sqrt{2}, \dots, (X_{N-1} - X_N) / \sqrt{2},$
- 3:     Let  $\hat{\mu} \leftarrow \text{LEARNMEAN}(O(\varepsilon \log(1/\varepsilon)), \mathcal{Y})$  for  
 $\mathcal{Y} = \hat{\Sigma}^{-1/2} X_{N+1}, \dots, \hat{\Sigma}^{-1/2} X_{2N}$
- 4:     **return**  $\mathcal{N}(\hat{\Sigma}^{1/2} \hat{\mu}, \hat{\Sigma})$

## 4. EXPERIMENTS

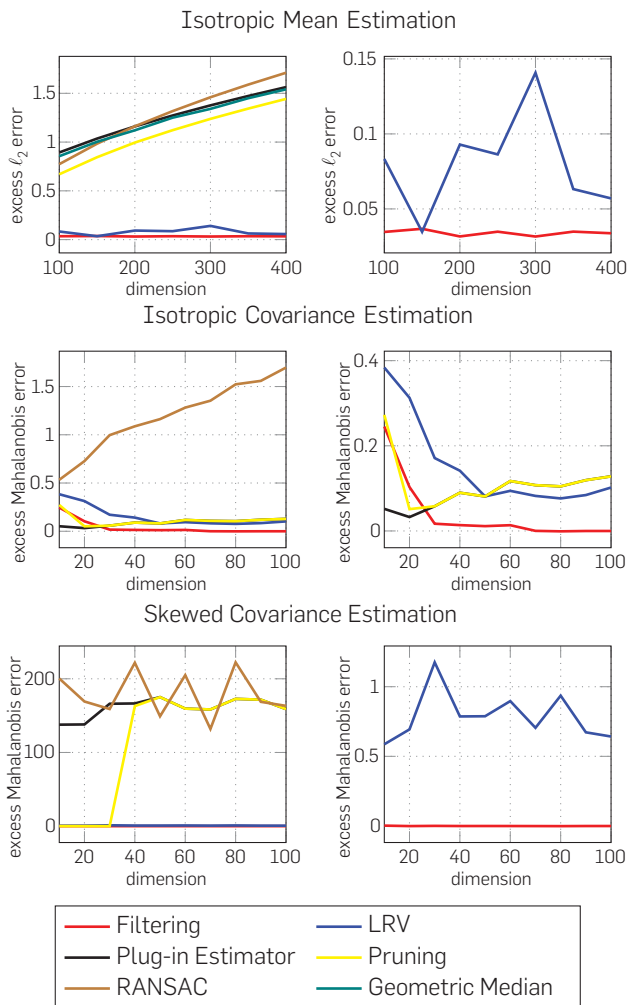
Our algorithms (or rather, natural variants of them) not only have provable guarantees in terms of their efficiency and robustness but also turn out to be highly practical. In Diakonikolas et al.,<sup>5</sup> we studied their performance on both synthetic and real-world data, and we discuss the results in this section.

In Figure 1, we demonstrate our results on synthetic data, for estimating the mean and covariance of a Gaussian. We compare our Filtering method with the algorithms of Lai et al.,<sup>20</sup> the empirical plug-in estimator, the empirical estimator in combination with pruning, random sample consensus (RANSAC),<sup>11</sup> and the geometric median (for mean estimation). The first row of plots displays mean

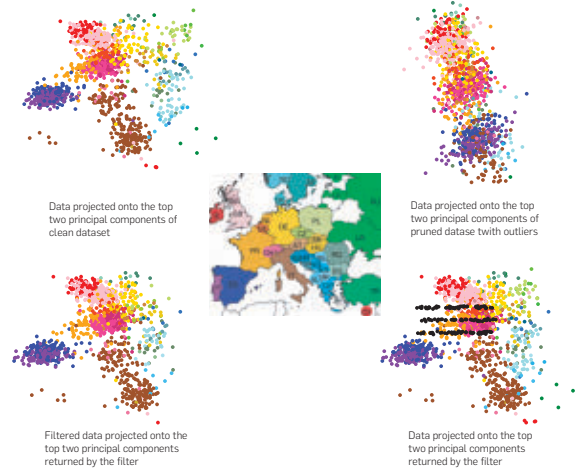
estimation for an isotropic Gaussian (with  $\varepsilon = 0.1$ ), the second row displays covariance estimation for an isotropic Gaussian, and the third row displays covariance estimation for a Gaussian with a highly-skewed covariance matrix (both with  $\varepsilon = 0.05$ ). The first column of plots compares all the methods, whereas the second column of plots omits the less accurate methods, to allow a more fine-grained comparison between our algorithm and competitive methods. The x-axis of each plot indicates the dimension of the problem, and the y-axis indicates the error incurred by the estimation method, where the baseline of 0 is the error of the plug-in estimator on the uncorrupted data. In Figure 1, for the mean estimation plots, this error is measured via  $\ell_2$ -distance, whereas for covariance estimation, this is measured in terms of Mahalanobis distance.

In all experiments, we found that our algorithm outperformed all other methods, often by substantial margins. As predicted by the theory, our error appears to remain constant as the dimension increases, but increases for all other methods (albeit minimally for LRV methods, which

**Figure 1. Robust parameter estimation on synthetic data. Our method (filtering) is shown to outperform all alternatives for both mean estimation (first row) and covariance estimation (last two rows).**



**Figure 2. Robust exploratory data analysis on semisynthetic data. The top-left figure shows the projection of a high-dimensional genomic dataset onto its top two principal components, which resembles the map of Europe (center). In the presence of synthetic outliers, this structure is lost (top-right). Our methods for robust covariance estimation allow us to preserve this structure (bottom).**



depend only logarithmically on the dimension). For mean estimation, our method performs better than LRV, which in turn performs much better than all alternatives. Similar trends are observed for covariance estimation, though the results are especially pronounced when estimating a skewed covariance, in which our methods outperform all others by orders of magnitude.

In our semisynthetic experiments, displayed in Figure 2, we revisit a classic study of Novembre et al.<sup>21</sup> In this study, the authors obtained a high-dimensional genomic dataset from the POPRES project. They annotated each data point with the individual’s country of origin and projected the dataset onto the top two principal components of the dataset. As displayed in the top-left plot in Figure 2, they found that the resulting projection closely resembles the map of Europe, thus leading to the adage that “genes mirror geography.” However, omitted from the description above is a crucial manual data curation process, in which immigrants were removed from the dataset, as they were considered to be genetic outliers. Our methods provide an automatic and principled way of removing outliers.

In our experiments, we worked with a projection of the original dataset onto the top 20 principal components. We injected synthetic noise points ( $\varepsilon = 0.1$ ) into the dataset and repeated the experimental procedure described above. Even with a pruning step, we found the empirical estimator was not able to preserve the structure of Europe (top-right of Figure 1). However, our method (based on our robust Gaussian covariance estimation algorithm) was able to relatively faithfully recreate the original map of Europe (bottom-left and bottom-right of Figure 1). Despite our filter being designed for Gaussian data, the method worked on the genomic data (which is not necessarily Gaussian) with minimal alterations.

## References

1. Awasthi, P., Balcan, M.F., Long, P.M. The power of localization for efficiently learning linear separators with noise. In *Proceedings of the 46th Annual ACM Symposium on the Theory of Computing*, STOC '14 (New York, NY, USA, 2014), ACM, 449–458.
2. Balakrishnan, S., Du, S.S., Li, J., Singh, A. Computationally efficient robust sparse estimation in high dimensions. In *Proceedings of the 30th Annual Conference on Learning Theory*, COLT '17 (2017), 169–212.
3. Charikar, M., Steinhart, J., Valiant, G. Learning from untrusted data. In *Proceedings of the 49th Annual ACM Symposium on the Theory of Computing*, STOC '17 (New York, NY, USA, 2017), ACM, 47–60.
4. Diakonikolas, I., Kamath, G., Kane, D.M., Li, J., Moitra, A., Stewart, A. Robust estimators in high dimensions without the computational intractability. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '16 (Washington, DC, USA, 2016), IEEE Computer Society, 655–664.
5. Diakonikolas, I., Kamath, G., Kane, D.M., Li, J., Moitra, A., Stewart, A. Being robust (in high dimensions) can be practical. In *Proceedings of the 34th International Conference on Machine Learning*, ICML '17 (2017), JMLR, Inc., 999–1008.
6. Diakonikolas, I., Kamath, G., Kane, D.M., Li, J., Moitra, A., Stewart, A. Robustly learning a Gaussian: Getting optimal error, efficiently. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18 (Philadelphia, PA, USA, 2018), SIAM.
7. Diakonikolas, I., Kamath, G., Kane, D.M., Li, J., Steinhart, J., Stewart, A. Sever: A robust meta-algorithm for stochastic optimization. In *Proceedings of the 36th International Conference on Machine Learning*, ICML '19 (2019), JMLR, Inc., 1596–1606.
8. Diakonikolas, I., Kane, D.M. Recent advances in algorithmic high-dimensional robust statistics. *CoRR*, abs/1911.05911, 2019.
9. Diakonikolas, I., Kane, D.M., Stewart, A. Statistical query lower bounds for robust estimation of high-dimensional Gaussians and Gaussian mixtures. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '17 (Washington, DC, USA, 2017), IEEE Computer Society, 73–84.
10. Diakonikolas, I., Kane, D.M., Stewart, A. List-decodable robust mean estimation and learning mixtures of spherical Gaussians. In *Proceedings of the 50th Annual ACM Symposium on the Theory of Computing*, STOC '18 (New York, NY, USA, 2018), ACM, 1047–1060.
11. Fischler, M.A., Bolles, R.C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 6, 24 (1981), 381–395.
12. Fisher, R.A. On the mathematical foundations of theoretical statistics. *Phil. Trans. R. Soc. Lond. Ser. A* 594–604, 222 (1922), 309–368.
13. Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., Stahel, W.A. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, Hoboken, New Jersey, 2011.
14. Hopkins, S.B., Li, J. Mixture models, robustness, and sum of squares proofs. In *Proceedings of the 50th Annual ACM Symposium on the Theory of Computing*, STOC '18 (New York, NY, USA, 2018), ACM, Hoboken, New Jersey, 1021–1034.
15. Huber, P.J., Ronchetti, E.M. *Robust Statistics*. Wiley, 2009.
16. Johnson, D.S., Preparata, F.P. The densest hemisphere problem. *Theor. Comp. Sci. I*, 6 (1978), 93–107.
17. Kearns, M.J., Schapire, R.E., Sellie, L.M. Towards efficient agnostic learning. *Mach. Learn.* 2–3, 17 (1994), 115–141.
18. Klivans, A.R., Long, P.M., Servedio, R.A. Learning halfspaces with malicious noise. *J. Mach. Learn. Res.*, 10 (2009), 2715–2740.
19. Kothari, P., Steinhart, J., Steurer, D. Robust moment estimation and improved clustering via sum of squares. In *Proceedings of the 50th Annual ACM Symposium on the Theory of Computing*, STOC '18 (New York, NY, USA, 2018), ACM, 1035–1046.
20. Lai, K.A., Rao, A.B., Vempala, S. Agnostic estimation of mean and covariance. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '16 (Washington, DC, USA, 2016), IEEE Computer Society, 665–674.
21. Novembre, J., Johnson, T., Bryc, K., Kutalik, Z., Boyko, A.R., Auton, A., Indap, A., King, K.S., Bergmann, S., Nelson, M.R., Stephens, M., Bustamante, C.D. Genes mirror geography within Europe. *Nature* 7218, 456 (2008), 98–101.
22. Prasad, A., Suggala, A.S., Balakrishnan, S., Ravikumar, P. Robust estimation via robust gradient estimation. *arXiv preprint arXiv:1802.06485* (2018).
23. Rousseeuw, P. Multivariate estimation with high breakdown point. *Math. Statist. Appl.*, 8 (1985), 283–297.
24. Tukey, J.W. A survey of sampling from contaminated distributions. In *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*, Stanford University Press, Stanford, California, 1960, 448–485.
25. Tukey, J.W. Mathematics and the picturing of data. In *Proceedings of the International Congress of Mathematicians* (1975), American Mathematical Society, 523–531.

**Ilias Diakonikolas** (iliad@cs.wisc.edu), University of Wisconsin, Madison, WI, USA.

**Gautam Kamath** (g@csail.mit.edu), University of Waterloo, Canada.

**Daniel M. Kane** (dakane@cs.ucsd.edu), University of California, San Diego, CA, USA.

**Jerry Li** (jerrl@microsoft.com), Microsoft Research AI, Redmond, WA, USA.

**Ankur Moitra** (moitra@mit.edu), Massachusetts Institute of Technology, Cambridge, MA, USA.

**Alistair Stewart** (stewart.al@gmail.com), Web3 Foundation, Zug, Switzerland.



This work is licensed under a Creative Commons Attribution International 4.0 license.

# Semantic Web for the Working Ontologist

*Effective Modeling for Linked Data, RDFS, and OWL*

**Dean Allemang  
James Hendler  
Fabien Gandon**

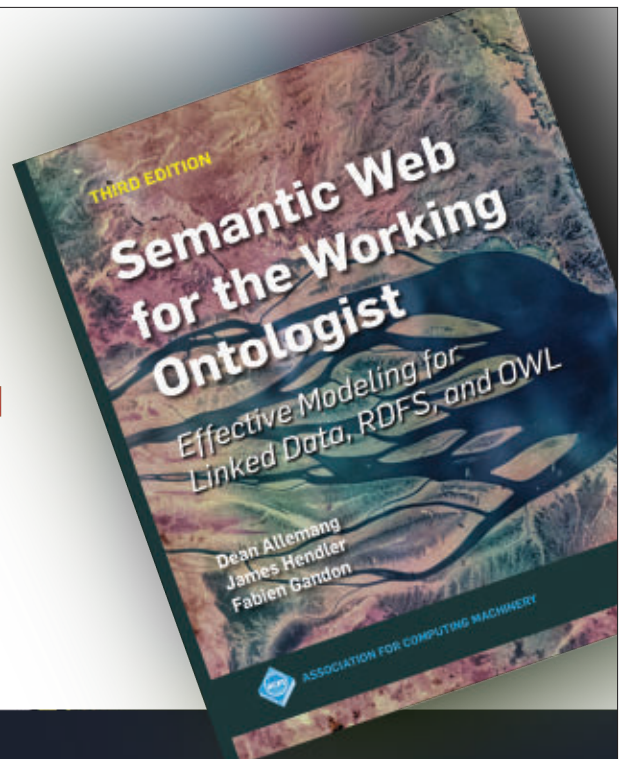
**THIRD EDITION**

ISBN: 978-1-4503-7617-4

DOI: 10.1145/3382097

<http://books.acm.org>

<http://store.morganclaypool.com/acm>



 **ACM BOOKS**  
Collection II

# CAREERS

## Ahmedabad University School of Engineering and Applied Science Assistant/Associate/Full Professor

The School of Engineering and Applied Science at Ahmedabad University invites applications for several tenure track faculty positions at all ranks in all areas of Computer Science and Engineering. Successful candidates will be expected to contribute to the advancement of their discipline and the School by developing a strong research program and to have a strong commitment to high-quality teaching. Interdisciplinary research and collaborations with industry are encouraged.

Ahmedabad University is a private, non-profit university. It was founded in 2009 by Ahmedabad Education Society (AES), a non-profit educational trust that is more than 80 years old. AES has been instrumental in setting up several institutions, amongst them are Indian Institute of Management Ahmedabad, National Institute of Design, Physical Research Laboratory, and CEPT University. The university is located in the heart of Ahmedabad, a UNESCO heritage city. The university is part of a neighbourhood that hosts several educational institutions and is surrounded by

a variety of industries and research establishments. A young university, it aims to be an excellent teaching and research institution, emphasizing inter-disciplinary research and innovative teaching which is premised on learning by doing. The School of Engineering and Applied Science brings together faculty from Computer Science, Electronics and Communications Engineering, Chemical Engineering, Mechanical Engineering and science disciplines. It offers programmes in Computer Science at bachelors, masters and doctoral levels.

The university offers highly competitive compensation and benefits. The university also supports professional growth of the faculty in various ways, including start-up research funding, seed grants, challenge grants for interdisciplinary research, and an annual faculty development grant. The university also provides funding for attending national and international conferences annually.

Applicants must have PhD degree in Computer Science or a closely related discipline. Applications are reviewed periodically throughout the year. Applications for Fall 2021 appointments should be submitted by June 7, 2021 for full con-

sideration. To apply: send CV, research statement and teaching statement to [faculty.search.seas@ahduni.edu.in](mailto:faculty.search.seas@ahduni.edu.in).

Please visit <https://ahduni.edu.in/careers> for more details.

## University of Macau Full Professor and Assistant / Associate Professor in Computer and Information Science

(Ref. No.: FST/CIS/AAP/03/2021; FST/CIS/FP/03/2021)

The Department of Computer and Information Science (CIS) of the University of Macau (UM) invites applications for the position of Full Professor in Computer Science and Assistant / Associate Professor in areas of Financial Technology and Distributed Computing, Software Engineering and Service Computing, Visualization and AR/VR. We are seeking candidate with a proven record of accomplishment in research and education.

CIS was founded in 1990 and currently has 25 academic staff. The Department offers Bachelor's, Master's and PhD degree programmes, which cover the main aspects of modern computer science and related technologies. UM is among the top 1% in ESI rankings in both Engineering and Computer Science. In the THE World University Rankings, the Computer Science programme is ranked among the top 200.

The candidates must have an earned PhD degree in related areas. Preference will be given to candidates with extensive research and teaching and corresponding academic leadership experience at the tertiary education level for Full Professor.

Applicants should visit <https://career.admo.um.edu.mo/> for more details and apply ONLINE.

## Viterbi School of Engineering Chair of the Department of Computer Science

USC Viterbi School of Engineering at the University of Southern California invites nominations and applications for Chair of the Department of Computer Science

The USC Viterbi School of Engineering at the University of Southern California invites nominations and applications for Chair of the Department of Computer Science. We are seeking an individual who can provide strong, dynamic and innovative leadership for advancing excellence in research, teaching, and service to the professional community. The position is at the tenured Professor rank, starting either Fall 2021 or Spring 2022, and with an initial Chair term of 3 years. In addition to a proven record of scholarly achievement, the candidate must also possess visionary technical leadership, a firm commitment to computer science education, as well as strong management and interpersonal skills. The candidate should have an earned doctorate in computer sci-

*The National  
Academies of*

SCIENCES  
ENGINEERING  
MEDICINE

### ARL Distinguished Postdoctoral Fellowships

The Army Research Laboratory (ARL) Distinguished Postdoctoral Fellowships provide opportunities to pursue independent research in ARL laboratories. Fellows benefit by working alongside some of the nation's best scientists and engineers, while enhancing the mission and capabilities of the U.S. Army and the warfighter in times of both peace and war.

Fellows must display extraordinary abilities in scientific research and show clear promise of becoming future leaders. Candidates are expected to have already successfully tackled a major scientific or engineering problem or to have provided a new approach or insight, evidenced by a recognized impact in their field.

Fellowships are one-year appointments, renewable for up to three based on performance. The award includes a \$100,000 annual stipend, health insurance, paid relocation, and a professional travel allowance. Applicants must have completed all requirements for a Ph.D. or Sc.D. degree by October 1, 2021, and may not be more than five years beyond their doctoral degree as of the application deadline. For more information and to apply visit [www.nas.edu/arl](http://www.nas.edu/arl).

Online applications must be submitted by  
June 15, 2021 at 5 PM Eastern Time.





## ADVERTISING IN CAREER OPPORTUNITIES

**How to Submit a Classified Line Ad: Send an e-mail to [acmm mediasales@acm.org](mailto: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](mailto: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](mailto:acmm mediasales@acm.org)

ence or in a closely allied field and be qualified for a tenured full professor appointment.

Established in 1976, the computer science department at the USC Viterbi School has experienced unprecedented growth in recent years. The department now boasts more than 80 faculty members (tenured/tenure-track, research and teaching), and has vibrant undergraduate and graduate programs. The department has strong research groups in AI and machine learning, graphics, robotics, software engineering, systems, and theory. Our faculty include a Turing award winner, 7 members of national academies, 8 ACM fellows, 15 IEEE Fellows, and 39 AAAI and AAAS fellows. The department will soon have a new 98,000-square-foot state-of-the-art home ([announced](#) in 2020).

Founded in 1905 and located near downtown Los Angeles, a global center for arts, technology and international business in the heart of the Pacific Rim, the USC Viterbi School is a hub for entrepreneurship that connects students from 64 countries. It is consistently ranked in the top 10 graduate programs (U.S. News and World Report), counting 189 full-time, tenure-track faculty members. It is home to the Information Sciences Institute, a Department of Energy EFRC (Energy Frontiers Research Center), and the Department of Homeland Security's first University Center of Excellence, CREATE. The School is affiliated with the Alfred E. Mann Institute for Biomedical Engineering, the Institute for Creative Technologies, and the USC Stevens Center for Innovation. Research expenditures typically exceed \$210 million annually.

To receive full consideration, candidates should apply on-line at <https://usccareers.usc.edu/>.

Applicants must submit a cover letter describing their leadership and administrative experiences, a statement describing their vision and goals as Chair, and a statement about their approach to diversity, equity and inclusion in higher education. A complete curriculum vitae, as well as research and teaching statements, must also be submitted. References will be requested at a later time.

Applications received by April 2nd, 2021, will be given full consideration; those received after this deadline may also be considered.

*USC is an equal opportunity, affirmative action employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, sexual orientation, gender identity, national origin, protected veteran status, disability, or any other characteristic protected by law or USC policy. USC will consider for employment all qualified applicants with criminal histories in a manner consistent with the requirements of the Los Angeles Fair Chance Initiative for Hiring ordinance.*

# The Handbook of Multimodal-Multisensor Interfaces, Volume 3

*Language Processing, Software,  
Commercialization, and Emerging Directions*

**Edited by Sharon Oviatt et al**

ISBN: 978-1-970001-72-3

DOI: 10.1145/3233795

<http://books.acm.org>

<http://store.morganclaypool.com/acm>



**ACM BOOKS**  
Collection 1



## Digital Government: Research and Practice

*Digital Government: Research and Practice (DGOV)* is an Open Access journal on the potential and impact of technology on governance innovations and its transformation of public institutions. It promotes applied and empirical research from academics, practitioners, designers, and technologists, using political, policy, social, computer, and data sciences methodologies.



For further information  
and to submit your  
manuscript,  
visit [dgov.acm.org](http://dgov.acm.org)

[CONTINUED FROM P. 120] MusiCorp by this evening. If she was late, the company would just take the song in whatever state it was in (locking her out of further modifying it), run it through some AIs, and release it. She didn't want that to happen, because much of her income flows from tiny per-millisecond-listen royalties. So her interest in the song succeeding was much greater than MusiCorp's. The company would be releasing a few dozen songs worldwide leading up to the weekend, and they only needed for a couple to hit. Skiz wanted to make sure that her song would be one of those hits.

Leaning into the console, she rested both hands palms down, leaning her forearms into the surface. Tiny movements in the console fur gave her arms and hands a warm tingle. Speaking into the space between her hands, "[Two measures before final chorus... Play...]" The song began playing, with the sound coming directly from the fibers of the console surface, surrounding her quite completely. She commanded, "[EQ... graphic... thirty six critical band...]," causing the furry surface to reshape beneath her hands and arms. Some fur fibers laid flat, others curved and receded, still others stood up, revealing a new surface that looked like a row of three dozen vertical 120mm slide potentiometers. "Knob" bumps in the "sliders" took on positions to match the EQ she had last applied at that point in the song, poised to be adjusted by hand or voice command. Using her fingers, she made a couple of quick fur-slider adjustments, replayed the segment, then commanded:

"[Stop.. New segment.. Bridge.. Guitar solo.. Sort of like Roger May's, from Love of My Life.]"

"{ALERT:}" piped up HiG, "{Use of that solo could be cost-prohibitive. Consider alternatives...}"

"[I know, I said LIKE that solo, not that exact solo. HC...]" she said into the now-flat fur surface. "{Yes...}" replied the AI. "[Maybe combine the Roger May solo with the Prince riff from the beginning of Doves Cry?]"

"{WARNING!!}" HiG yelled, "{Any use of Prince or Prince-like material will most certainly...}"

"[OK Fine.]" she interrupted HiG,

clarifying, "[Combine the May solo with the Prince solo turned backward. Use Genetic Algorithm #7 to merge and mutate them, arriving at three different solos that are clear of any potential copyright issues. Play me those three]."

Seven seconds later, she was listening to the first solo. Not bad. The second one was much better, but the third one was terrible. "[OK... Use the 2nd solo, but change the five highest notes to E F E G F. Final five note durations: dotted quarter, eighth, quarter, quarter, and hold the last note. Transpose last two notes up one octave. Put final note in distortion feedback, and fade it into reverb over eight seconds. Play...]"

She listened to the result. Yep, that would do. Almost finished. Now for the vocals.

Ch33t@h (the artist) hadn't come out with a "new" song for a while. She had been dead for well over five years, and everyone knew it, but that didn't matter very much. Since MusiCorp owned the whole Ch33t@h catalog and the rights to all likenesses of her, the company could release "new" songs by the artist at any time, usually with back-channel press "leaks" that some previously unreleased gem had been "discovered" in an attic, or basement, or dusty book/record store, or on an old hard drive. Fans would vigorously argue over whether the new work(s) were her best work, or what period of her life they must have come from, but hardly anyone argued as to their authenticity. This was mainly because: 1) the fans wanted new (actually old, but new) music, really badly, and 2) the AIs, combined with engineer/producers like Skiz, were really good at this.

Selecting one of the musical sketch books of the dead artist, Skiz found a page with some lyrics scrawled on it, enough and interesting enough to justify a whole new song. She laid the book face-down on the console surface. The fur flattened and turned translucent, emitting a faint blue glow. Two seconds later, she heard Ch33t@h's voice begin speaking the words.

Commanding through the console: "[HC: Infer melody from verse 1 backing track...]" The speaking turned to



singing. “[Sadder please...]” caused a mode change, and distinctive voice quality shift. “[Not that sad...]”.

HiG piped up, “{This melody is too similar to seven historical songs, four of which MusiCorp has no primary, intrinsic, or derivative rights to...}”.

HarryBlox chimed in. “{Searching... Located... Calculating estimated royalties...}”.

Sighing, Skiz commanded the AIs, “[HC: Use Genetic Algorithm #11 to modify the melody, iterate with HiG and HarryBlox until royalty free ...]”. The console fur pulsed and stood up a bit, oscillating in waves like a wheat field in the wind.

The melody got worse, and worse, until it was pretty much just two alternating notes. That would not do at all. “[Stop... push... bang bang]” “[HC: Randomize melody over octave range]” “[pop... execute]”. Skiz had learned archaic shell commands in her 3rd grade computing history course. They made her work go a little faster, but mostly they appealed to her nerdy side. HC ran GA#11 again on the new randomized melody, iterating with HiG and HarryBlox. The resulting melody was rapidly shaping up... to be equally horrible.

Skiz groaned and decided to take a new tack. Speaking to the console, “[Delete melodies. Speech mode for lyrics, spoken-word prosaic style. Let’s have some fun]”

“{Yep!}” purred the console.

“[Gimme a microphone... RCA 77DX.]” The flat fuzzy surface rippled, and a 2” silo-like tower rose from the center. “[UA 610 Tube Preamp.... DBX 160 Compressor, female backing vocal settings. Lexicon 224 Reverb, very wet. .... Record... Start...]”.

As the backing track played, and the resynthesized voice of the long dead Ch33t@h spoke the words to her poem, Skiz began to sing a high, haunting descant to the underlying musical themes from the backing tracks. She closed her eyes, letting the composition and the poetry guide her. Singing higher, flowing like a hawk soaring over the musical landscape, the result was sounding quite good, and, best of all, the AI nagbots were silent. Until...

“{ALERT...}, a human voice (from MusiCorp) broke into the session:

## Much of her income flows from tiny per-millisecond-listen royalties, so her interest in the song succeeding was much greater than MusiCorp's.

“{What is that singing?}”

“[Uh... It’s me?]”, Skiz said tentatively. “[Sorry. I’ll return to the Ch33t@h voice synthesis.]”

...pausing... “{No. Complete project in that style, with your own voice.}” commanded MusiCorp.

Uh, OK, whatever, Skiz thought. Within two hours, she had completed the song, and eSigned it over to the mastering, transcoding, marketing, and release bots at MusicCorp. The fur console burped out her TDrive, purred, and slumped silent. The project was out of her hands now, which felt good, but it felt really odd to have released something with her own voice on it.

.....

After submitting the song on Thursday, she had turned off all of her devices, including her CCalls and alarms. Saturday morning she was sleeping in, recovering from celebrating the completed project, but something really weird was going on. Face down on the pillow, she could feel something relentlessly poking her in her side. At first she had dreamed that it was her childhood cat Bowie (now long gone), but as she woke up, realizing it couldn’t be Bowie, and feeling another stabbing prod, she sat up with a start. Her TDrive had crawled out of her bag, up onto the bed, and was urgently jabbing her in the ribs!! OK, very strange, and had definitely never happened before.

She spoke aloud, “[News...]”

The walls lit up with news feeds and

images. Her lower neck started buzzing immediately. The feeds that first caught her eye were the music industry rags, where the fan forums were on fire about the newest Ch33t@h release. Most threads centered around the startling news that apparently Ch33t@h had a previously unknown daughter, and that mystery child’s voice was the one floating high above the new song, as her mom poured her heart out in spoken-word.

Skiz chuckled at the nefarious creativity of music marketing people. She started laughing, almost hysterically. How gullible fans are.

“[Comms on... Sort messages by importance...]”

Trying to compose herself, she settled back, preparing to deal with the message queue, still laughing out loud. Her laughing didn’t last long, as the realization came that her that life was about to change in every way. A new star isn’t born every week (monthly, maybe). The world would be expecting more songs, and a concert tour featuring Ch!Ld (the artist name the adfolk had made up for the angel-voiced baby girl of Ch33t@h). The fans would demand it. The record company would REALLY demand it.

“{Messages: URGENT Incoming Comms from MusiCorp...”

URGENT: Please eSign and return this new artist contract immediately...

URGENT: Please eSign and return these name and legal identity change forms...

URGENT: A plastic surgery appointment has been made for Ch!Ld. This Tuesday at 11:AM.

Fasting recommended prior to this appointment...

URGENT: Studio is booked for Ch!Ld tomorrow, 9:AM. New backing tracks already on TDrive.

URGENT: Ch!Ld has a voice-only interview scheduled with MVTVZ tonight from 7-7:15PM.

URGENT: ... ..

URGENT: ... ..

URGENT: ... ..}”

**P-Ray** is the creative/artistic moniker of **Perry R. Cook**, who is professor emeritus of computer science (also music) at Princeton University. Cook is advisor and IP Strategist to social music company Smule, and co-founder of online arts education company Kadenze.

© 2021 ACM 0001-0782/21/5 \$15.00

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/3453712

P-Ray

## Future Tense Behold the Ch!Ld

*Opportunity can come calling when you least expect it.*

RRRRRRRRRRRRRRRRZZZ... SKIZ AWOKE to a buzzing inside the lower back of her skull. It wasn't the jarring buzz of an alarm, nor the gentler bio-alert, telling her to take her meds. It wasn't the urgent "Item On Sale! Buy It NOW!!!" buzz, either. Rather, it was an incoming CommCall that was pulsing her neuro/vibrotactile implant. Who would be calling at this ungodly...? Oh no... Checking <date:time> in her retinal projection, she quickly realized that she had overslept, and was REALLY late for her session. That buzzing would be the studio Ccalling to ask why she wasn't there using her (expensive) booked time.

Responding to the CCall without picking up:

(in a soft, low voice to the left)

"[reply.]"

"[Coming Now.]"

(to the right, in a normal voice)

(soft low voice to the left)

"[send... dismiss.]"

rrrrzzzzp ... the pulsing stopped. OK. Up, throw on jeans, t-shirt, hoodie, out the door and into a TRaxi to the studio. She could do a little work during the 15-minute trip there. "[WAIT! Hold on]" voice commanding the TRaxi. Back into the apartment to grab her important TDrive (Thumb Drive) which held all of her current projects. Back in the TRaxi, "[Go! Fast. I'll pay extra.]"

Fourteen minutes later, she was running into the studio building, the security AI opening each door as she approached. Down the last hallway, peeling off her hoodie, she entered the main studio, and tossed the TDrive onto the production con-

sole. The console woke up, purring, appearing to stretch itself. The furry console surface rippled, absorbing the TDrive, giving the appearance of a huge rectangular grey cat devouring a bulb of flesh.

The studio wasn't completely necessary, of course. Producers, composers, DJs, and artists had been making proper music in bedrooms, on airplanes, in coffee shops, or wherever for decades now, but Skiz liked the studio, partly because of the furry console. She also loved the feeling of just being there, since it was quite a bit larger and nicer than her living cube. Most of all, she had gotten used to the higher bandwidth links to the agents she used in her production work: HarryBlox (block chain music licensing), GettySamplz (repository of raw audio materials), CSpot (Cliche Spotter, an

aesthetic evaluator/recommender), HC (Hit Composer), and HiG (Hook infringement Guardian).

Of course her TDrive had versions of all of these AI tools, and subsets of the databases they accessed. But a TD can only hold a zettabyte or so, and reading/writing the data coded into the DNA of that lab-grown clone of her own thumb takes some time. Working in the studio is significantly faster, with everything quantOptically linked to the massive servers on the BPB (Big Phat Backbone).

By the time she sat down, the console had already scanned the directories and diff files on her TDrive, and had synced itself to the current state of the latest project. She was actually quite far along on this song, which was good, because it was due to be delivered to [CONTINUED ON P. 118]



# Systor2021

June 14-16

The 14th ACM International  
System and Storage Conference

*Hybrid*

Sponsored by



In cooperation with



Platinum supporter



Gold supporters



## Keynote Speakers:

**Alexandra (Sasha) Fedorova**, UBC

**Kim Keeton**, HP labs

## Program Chairs:

**Vijay Chidambaram**, University of Texas at Austin / VMWare Research, USA

**Danny Raz**, Technion - Israel Institute of Technology, Israel

## Presentation on topics including:

Distributed, parallel, and cloud systems;

Security, privacy, and trust;

File and storage systems;

Fault tolerance, reliability, and availability



*scan me*



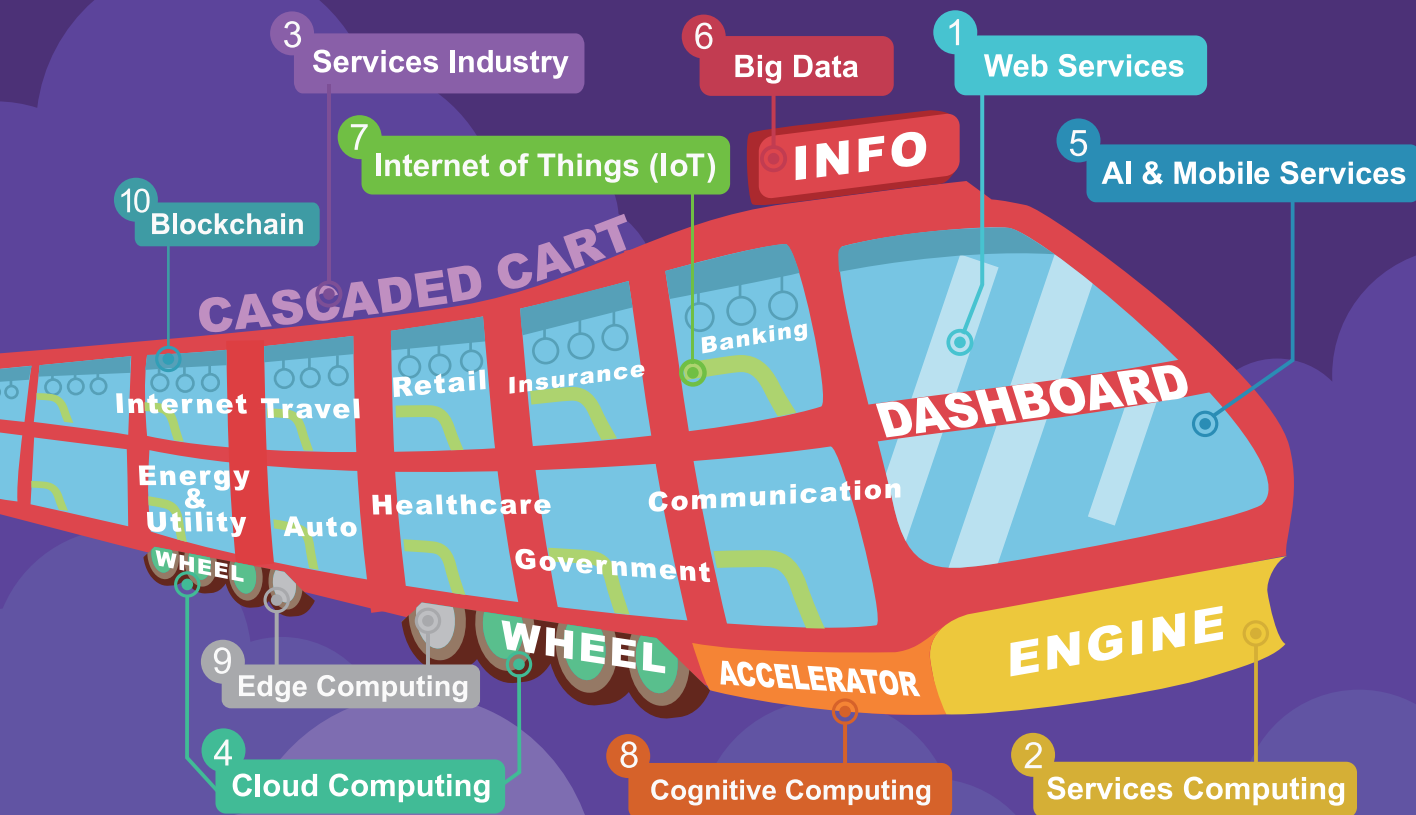
# 2021 Digital Transformation

Sept. 18-22, 2021, Online & Satellite Sessions

••• CELEBRATING THE 19<sup>th</sup> GATHERING OF ICWS •••



- 1 2021 International Conference on Web Services (ICWS 2021)
- 2 2021 International Conference on Services Computing (SCC 2021)
- 3 2021 World Congress on Services (SERVICES 2021)
- 4 2021 International Conference on Cloud Computing (CLOUD 2021)
- 5 2021 International Conference on AI & Mobile Services (AIMS 2021)
- 6 2021 International Conference on Big Data (BigData 2021)
- 7 2021 International Conference on Internet of Things (ICIOT 2021)
- 8 2021 International Conference on Cognitive Computing (ICCC 2021)
- 9 2021 International Conference on Edge Computing (EDGE 2021)
- 10 2021 International Conference on Blockchain (ICBC 2021)



## Submission Deadlines:

Early-Birds Submission: 4/9/2021  
Regular Submission: 5/14/2021

## Contact:

confs@servicessociety.org  
www.icws.org



Largest not-for-profits organization (501(c)(3))  
dedicated for serving 30,000+ worldwide  
services computing professionals



ICWS.ORG