

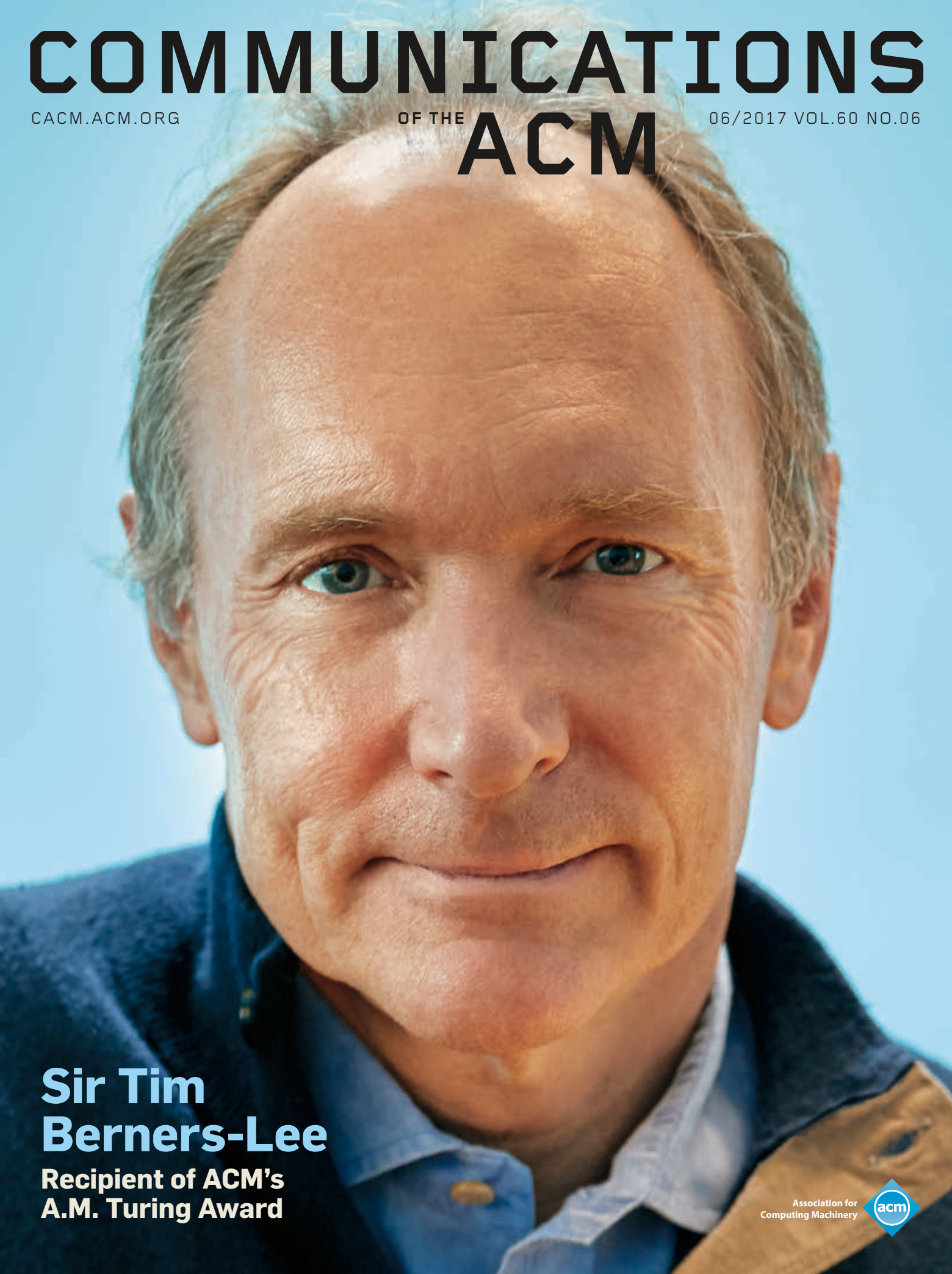
COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

06/2017 VOL.60 NO.06



**Sir Tim
Berners-Lee**

**Recipient of ACM's
A.M. Turing Award**

Association for
Computing Machinery

acm

CONFERENCE 27 – 30 November 2017
EXHIBITION 28 – 30 November 2017
BITEC, Bangkok, Thailand

THE CELEBRATION OF LIFE & TECHNOLOGY

CALL FOR SUBMISSIONS

Submit your works & be a presenter at SIGGRAPH Asia!

SIGGRAPH Asia 2017 invites you to submit your works and showcase your outstanding creative ideas and innovations at the 10th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia, taking place from **27 – 30 November**, in Bangkok, Thailand.

Log-on to sa2017.siggraph.org/submitters to submit your works.

CONFERENCE PROGRAMS' SUBMISSION DEADLINES*:

| DEADLINES | PROGRAMS |
|----------------|---|
| 27 April 2017 | Workshops' Proposals |
| 23 May 2017 | Technical Papers |
| 30 May 2017 | Emerging Technologies |
| 1 June 2017 | Art Gallery |
| 13 June 2017 | Symposium on Education |
| 21 June 2017 | Symposium on Mobile Graphics and Interactive Applications |
| 28 June 2017 | Courses |
| 29 June 2017 | Symposium on Visualization |
| 15 July 2017 | Student Volunteers - Team Leaders Application |
| 19 July 2017 | Computer Animation Festival |
| 30 July 2017 | VR Showcase |
| 12 August 2017 | Student Volunteers Application |
| 15 August 2017 | Posters Technical Briefs Workshops' Papers |

*The submission time for all dates is 23:59 UTC/GMT

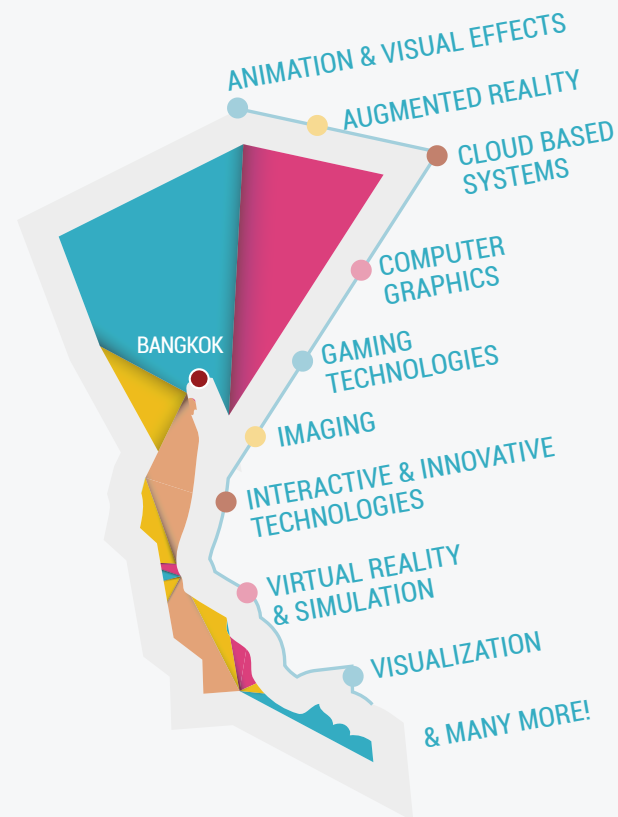
Visit sa2017.siggraph.org for more details.

CALL FOR EXHIBITORS & SPONSORS

Be a part of the SIGGRAPH Asia Exhibition – Asia's Digital Media Marketplace

Meet close to 7,000 technical and creative industry experts and individuals from over 60 countries and regions face-to-face to explore business opportunities, partnerships, and to strengthen existing relations – all in person at SIGGRAPH Asia 2017. Book your stand now to secure your preferred location.

Contact Clariss Chin at **+65 6500 6722** or clariss.chin@siggraph.org for more information on the exhibit space options and fees, as well as sponsorship packages.





CELEBRATING

50

YEARS OF
SOFTWARE ENGINEERING

GOTHENBURG
MAY 27 - JUNE 3, 2018

SWEDEN

SUBMISSION DATES

| | |
|------------------------------|--------------|
| Technical papers | Aug 25, 2017 |
| Workshop Proposals | Oct 10, 2017 |
| SE in Practice | Oct 23, 2017 |
| SE in Education and Training | Oct 23, 2017 |
| SE in Society | Oct 23, 2017 |
| NIER | Oct 23, 2017 |
| Technical Briefings | Nov 01, 2017 |
| Doctoral Symposium | Nov 20, 2017 |
| Demonstrations | Nov 20, 2017 |
| ACM SRC | Jan 08, 2018 |
| Student Volunteers | Jan 22, 2018 |
| Posters | Feb 05, 2018 |

SUPPORTED BY



UNIVERSITY OF
GOTHENBURG



City of
Gothenburg



INTERNATIONAL CONFERENCE
ON SOFTWARE ENGINEERING

Departments

- 5 **Editor's Letter**
Ten Years at the Helm of *Communications of the ACM*
By Moshe Y. Vardi
-
- 7 **Cerf's Up**
Open Access to Academic Research
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
Technologies *Do* Have Ethics
-
- 10 **BLOG@CACM**
Balancing Teaching CS Efficiently with Motivating Students
Mark Guzdial suggests a new balance is needed in computer science education between discovery learning and direct instruction.
-
- 39 **Calendar**
-
- 101 **Careers**

Last Byte

- 104 **Q&A**
This Is for Everyone
Sir Tim Berners-Lee on the formative years of the World Wide Web, and the challenges it now faces.
By Leah Hoffmann

News

- 12 **Deep Learning Takes on Translation**
Improvements in hardware, the availability of massive amounts of data, and algorithmic upgrades are among the factors supporting better machine translation.
By Don Monroe
-
- 15 **Optimization Search Finds a Heart of Glass**
Analog computing could provide greater efficiency, improved digital algorithms.
By Chris Edwards
-
- 17 **Potential and Peril**
The outlook for artificial intelligence-based autonomous weapons.
By Sarah Underwood
-
- 20 **Weaving the Web**
Sir Tim Berners-Lee created a paradigm shift that changed the world with his invention of the World Wide Web, Hypertext Transport Protocol, and Hypertext Markup Language.
By Neil Savage



Watch Sir Tim discuss his work in this exclusive *Communications* video. <https://cacm.acm.org/videos/turing-award-recipient-sir-tim-berners-lee>

- 24 **ACM Panels in Print: Big Data**
A discussion with David Blei, Daphne Koller, Vipin Kumar, and Michael Stonebraker

Viewpoints



- 26 **Inside Risks**
Trustworthiness and Truthfulness Are Essential
Their absence can introduce huge risks ...
By Peter G. Neumann
-
- 29 **Broadening Participation**
The Influence and Promise of Alliances
Evaluating the influence of broadening participation efforts on students, faculty, organizations, and the computing education infrastructure.
By Leslie Goodyear, Gary Silverstein, and Linda P. Thurston
-
- 31 **Kode Vicious**
Forced Exception Handling
You can never discount the human element in programming.
By George V. Neville-Neil
-
- 33 **Viewpoint**
Remaining Trouble Spots with Computational Thinking
Addressing unresolved questions concerning computational thinking.
By Peter J. Denning

Practice



40

40 The Debugging Mind-Set
Understanding the psychology of learning strategies leads to effective problem-solving skills.
By Devon H. O'Dell

46 Too Big NOT to Fail
Embrace failure so it does not embrace you.
By Pat Helland, Simon Weaver, and Ed Harris

51 Conversations with Technology Leaders: Erik Meijer
Great engineers are able to maximize their mental power.
By Kate Matsudaira

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

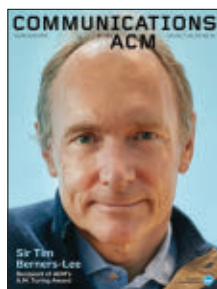
Contributed Articles



66

56 The SCION Internet Architecture
Adhering to the end-to-end principle even more than the current Internet yields highly available point-to-point communication.
By David Barrera, Laurent Chuat, Adrian Perrig, Raphael M. Reischuk, and Pawel Szalachowski

66 The Dynamics of Work-Family Conflict
Conflict is averted by separating work and family time and responsibility, as reflected in millions of tweets.
By Yili Liu and Lina Zhou



About the Cover:
Sir Tim Berners-Lee, recipient of ACM's 2016 A.M. Turing Award, photographed by Alexander Berg on the campus of the Massachusetts Institute of Technology.

Review Articles

72 Learnable Programming: Blocks and Beyond
New blocks frameworks open doors to greater experimentation for novices and professionals alike.
By David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, and Franklyn Turbak



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

Research Highlights

82 Technical Perspective
What Led Computer Vision to Deep Learning?
By Jitendra Malik

84 ImageNet Classification with Deep Convolutional Neural Networks
By Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton

91 Technical Perspective
Low-Depth Arithmetic Circuits
By Avi Wigderson

93 Unexpected Power of Low-Depth Arithmetic Circuits
Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi



Association for Computing Machinery
Advancing Computing as a Science & Profession



ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

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

ACM COUNCIL

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

BOARD CHAIRS

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

REGIONAL COUNCIL CHAIRS

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

PUBLICATIONS BOARD

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

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

Computer Science Teachers Association
Mark R. Nelson, Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

STAFF

DIRECTOR OF PUBLICATIONS
Scott E. Delman
cacm-publisher@cacm.acm.org

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

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

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

CONTACT POINTS

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

WEBSITE
<http://cacm.acm.org>

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

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

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

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

Media Kit acmm mediasales@acm.org

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

EDITORIAL BOARD

EDITOR-IN-CHIEF
Moshe Y. Vardi
eic@cacm.acm.org

NEWS

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

VIEWPOINTS

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

PRACTICE

Chair
Stephen Bourne
Board Members
Eric Allman; Samy Bahra; Peter Bailis; Terry
Coatta; Stuart Feldman; Camille Fournier;
Benjamin Fried; Pat Hanrahan; Tom Killalea;
Tom Limoncelli; Kate Matsudaira;
Marshall Kirk McKusick; Erik Meijer;
George Neville-Neil; Theo Schlossnagle;
Jim Waldo; Meredith Whittaker
The Practice section of the ACM
Editorial Board also serves as
the Editorial Board of queue.acm.org.

CONTRIBUTED ARTICLES

Co-Chairs
Andrew A. Chien and James Larus
Board Members
William Aiello; Robert Austin; Elisa Bertino;
Gilles Brassard; Kim Bruce; Alan Bundy;
Peter Buneman; Carlo Ghezzi; Carl Gutwin;
Yannis Ioannidis; Gal A. Kaminka;
Karl Levitt; Igor Markov; Gail C. Murphy;
Bernhard Nebel; Lionel M. Ni; Adrian Perrig;
Sriram Rajamani; Marie-Christine Rousset;
Krishan Sabnani; Ron Shamir;
Yoav Shoham; Michael Vitale;
Hannes Werthner; Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs
Azer Bestavros and Gregory Morrisett
Board Members
Martin Abadi; Amr El Abbadi; Sanjeev Arora;
Michael Backes; Nina Balcan; Andrei Broder;
Doug Burger; Stuart K. Card; Jeff Chase;
Jon Crowcroft; Alexei Efron; Alon Halevy;
Sven Koenig; Xavier Leroy; Steve Marschner;
Tim Roughgarden; Guy Steele, Jr.;
Margaret H. Wright; Nikolai Zeldovich;
Andreas Zeller

WEB

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

ACM Copyright Notice

Copyright © 2017 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@hq.acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$269.

ACM Media Advertising Policy

Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current advertising rates can be found by visiting <http://www.acm-media.org> or by contacting ACM Media Sales at (212) 626-0686.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhlp@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM*
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA

Printed in the U.S.A.



Association for
Computing Machinery





Moshe Y. Vardi

DOI:10.1145/3090801

Ten Years at the Helm of *Communications of the ACM*

IN JANUARY 2007, I undertook the task of revitalizing *Communications of the ACM*. In the 1970s, *Communications* was one of the premier publications in computing, publishing seminal articles such as Codd's "A Relational Model of Data for Large Shared Data Banks" (June 1970) and Rivest, Shamir, and Adleman's "A Method for Obtaining Digital Signature and Public-Key Cryptosystems" (February 1978). By the 1990s, however, *Communications'* content had evolved to be strongly slanted toward Management Information Systems. Over time, a significant segment of the ACM membership had lost interest in the publication. When David Patterson became ACM President in 2005, he argued that this state of affairs was unacceptable. The revitalized *Communications* was launched in July 2008, and by March 2009 I was able to conclude "Yes, it can be done" (see <https://goo.gl/0ySCMj>). Now, after 10 years at the helm of *Communications*, it is time for me to move on and for the magazine to get new leadership. Expect a formal announcement on this in the very near future!

My 10 years with *Communications* has been an incredible learning experience. I would like to share one of the most important lessons I learned. To turn *Communications* around, it was important to understand first what went wrong. How has *Communications* evolved from a premier publication in computing to one that was of interest only to a narrow segment of the computing community? The answer, in one word, is *emergence*, the phenomenon whereby a systemwide behavior arises through numerous interactions among system components. An example of

emergence is gentrification, which is a process of renovation of deteriorated urban neighborhoods by means of the influx of more affluent residents. The reverse trend is usually called urban decline. Gentrification and urban decline are the result of numerous individual decisions of residents moving into and moving out of the neighborhood. With no central direction, a neighborhood can completely change its character over a couple of decades.

The standard scholarly editorial model is that of *filtration*: authors submit articles, and editors filter them, with the help of reviewers, based on scope and quality. While the editorial process plays a critical role in shaping the face of a publication, the dominant factor is the nature of the submitted articles, which is determined solely by the submitting authors. Authors make submission decisions to a large measure based on articles already published. Thus, just as a neighborhood can change its character over a couple of decades, a publication can see its character change in just a few years. This is how *Communications* changed in the 1990s from a magazine representing all of computing to essentially "MIS Monthly."

The key to the turnaround of *Communications* was to change the editorial model from one mostly based on filtration to one mostly based on *curation*. The word "curation" comes from the verb "to care." It is typically used in the context of museums, where a curator selects items for exhibits. It also refers more broadly to the process of gathering and selecting content. While *Communications* is open to submissions, the lion's share of its published content is curated. Certain sections,

such as News, Viewpoint columns, and Practice, are purely curated, with an editorial-board section in charge of the curation. Other sections, combine curation and filtration. Take Contributed Articles, for example. During 2016, we received 202 submissions. Most of these articles were unsolicited, but a significant fraction of the submitted articles were solicited, which means that an editor encouraged a specific author to submit an article on a specific topic. Both solicited and unsolicited articles are subject to the same rigorous peer review, but the selection of topics and authors of solicited articles results in a higher probability of a positive editorial outcome.

Curation ensures *Communications* continues to be a publication that is broadly representative of computing. But curation requires an editorial board that is not merely reactive, but is strongly proactive, continually seeking topics and authors for high-quality articles. *Communications'* Editorial Board consists of approximately 100 committed volunteers. The quality and commitment of the Editorial Board is the real key to the success of the revitalized *Communications*. For all of *Communications'* success these past years, it will continue to remain a work in progress that will need the dedicated involvement of members of the community to keep it current and relevant for the fast-changing field of computing. Both ACM and the community will need to continue investing in *Communications* to keep it strong and vital.

Follow me on Facebook, Google+, and Twitter.

Moshe Y. Vardi, EDITOR-IN-CHIEF

Copyright held by author.

ACM Europe Conference

Barcelona, Spain | 6 – 8 September 2017

The ACM Europe Conference, hosted in Barcelona by the Barcelona Supercomputing Center, aims to bring together computer scientists and practitioners interested in exascale high performance computing and cybersecurity.

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

Keynote Talk by ACM 2012 Turing Award Laureate Silvio Micali, “ALGORAND: A New Distributed Ledger”

Co-located events:

- ACM Europe Celebration of Women in Computing: WomENCourage 2017 (Requires registration, <https://womencourage.acm.org/>)
- EXCDI, the European Extreme Data & Computing Initiative (<https://exdci.eu/>)
- Eurolab-4-HPC (<https://www.eurolab4hpc.eu/>)
- HiPEAC, the European Network on High Performance and Embedded Architecture and Compilation (<https://www.hipeac.net/>)

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

Registration to the ACM Europe Conference is free of charge for ACM members and attendees of the co-located events.



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



Vinton G. Cerf

DOI:10.1145/3084224

Open Access to Academic Research

It is June and as we do every year, ACM celebrates the extraordinary accomplishments and contributions of some of our colleagues in computer science and information

technology. This year, we honor Sir Tim Berners-Lee as the recipient of ACM's most prestigious 2016 A.M. Turing Award. The impact of the World Wide Web has steadily increased since it was first introduced in late 1991 and stands as a remarkable infrastructure, enhancing the value of the Internet, data-centers, smartphones, and all manner of other programmable systems.

This year also marks the first time the new ACM Prize in Computing is awarded—the inaugural recipient is Alexei A. Efros, for his groundbreaking, data-driven approaches to computer graphics and computer vision. Many other deserving recipients will be feted at the annual ACM Gala in San Francisco on June 24 and I hope to see many of you there to celebrate the achievements of our colleagues.

I recently spent a half-day with a group of government agency, publishing industry, and academic institutional representatives to discuss key considerations leading to improved access to academic research results, associated data, and analytic software. The sponsoring organization is the Open Scholarship Initiative (osinitiative.org) and its primary goal is accessibility of research and scholarly output. This was not necessarily about free access as much as making things easily discoverable and accessible.

The world of scientific and academic publishing has grown over time and this is not surprising. The number of narrowly focused academic publications is increasing, partly driven by business models of profit and non-

profit institutions and partly by the “publish or perish” dynamic in the academic world. Library budgets have not kept up with the cost of increasing numbers of publications. Changing this dynamic may necessitate revising the metrics of value: favoring quality over quantity. Academic tenure decisions often seem to turn on quantity and perhaps that must change.

Reproducibility of reported research is an important trend and is aided by funding-agency requirements for the preservation of research data, associated metadata, software and equipment documentation, as well as reported analytic results. This is the essence of the scientific method and is a laudable goal. The U.S. National Science Foundation sponsors the Research Data Alliance that undertakes to achieve this objective. Preservation of software and its execution environments is a topic about which I have written more than once in this column so I will simply reiterate here the challenge and importance of achieving this objective.

One of the challenges associated with data and publication preservation is the creation of business models that can sustain long-duration archives over decades, if not centuries. Present publication business models often sustain significant portions of the operating costs for academic organizations. For many research publishers, all of their operating costs must be covered this way. Ironically, researchers have often noted that academic colleagues at little direct cost

to publishers undertake the editing and review functions of most scientific journals. This has led some institutions to consider the formation of digital publication processes staffed by volunteer editors and reviewers. It should be noted, however, that considerable infrastructure must be maintained to ensure access to digital content over long periods of time and that, too, has an underlying cost.

In a recent National Academy of Engineering Spring Symposium, one of the speakers, Bret Victor, offered a working example of a modern-day digital publication that was far from static. Indeed, the tables and charts were fully interactive. Readers could alter parameters to see how the results might look under varied conditions. This kind of composite publication might indeed become the forerunner of a mechanism for reproducibility, especially if the researcher's tools might be accessed through the publication to test new hypotheses or to feed new data into the system.

In the ensuing discussions, it seemed clear the participants who might have been expected to be at odds on the economics of research publication were in fact open to exploring new ways to ensure increasingly open access to research results and data. I was reminded of the success of the Human Genome Project and the role that publishers played: if your papers were to be published, you had to agree to put your discovered genetic sequences into one of three international human genome databases. The resulting sharing of this key information accelerated our understanding of the human genome and its implications. □

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

Copyright held by author.

Technologies *Do* Have Ethics

THE CERF'S UP column "Social and Ethical Behavior in the Internet of Things" (Feb. 2017) by Francine Berman and Vinton G. Cerf was a welcome reminder of the importance of ethical issues involving sociotechnical systems in general and the Internet of Things in particular. Berman and Cerf did a great service giving them a high profile and thoughtful exposition. Here, we focus on their claim "Technologies have no ethics." Many computing professionals express this opinion, and we are confident many more believe it. But we think it is, as stated, a mistake, indeed a perilous mistake.

It is true that technologies do not "have ethics" in exactly the same way human beings have ethics. A human being is a carbon-based, biological entity, and any computer artifact (or other technological device) is fundamentally silicon-based and mechanical. Despite their differences, humans and technologies are interrelated and co-dependent. Society shapes technology, and technology shapes society. Technologies are the creations of humans; without humans, the technologies would not exist, and humans drive the creation of technology. Humans imbue their creations with moral significance, meaning their creations embody ethical decisions. Those ethics may be noble or they may be sketchy, but human ethics live inside every technology.

The 2009 book *Technology and Society: Building Our Sociotechnical Future* by Deborah G. Johnson and Jameson M. Wetmore, as well as the work of many other scholars of science and technology studies, addressed sociotechnical systems, including the Internet of Things. In that context, we explore some aspects of how ethics, technology, and computing professionals are related. Sociotechnical systems include people, devices, policies, and the connections among them. To properly understand any technology, the entire sociotechnical system of which they are a part must be understood. As technologies are developed,

they are already part of a sociotechnical system that develops concurrently with the technology.

At each stage of development of a sociotechnical system, people make decisions. Any system, including its technological components, embodies those decisions. Most decisions, regardless of how "technical" they may appear at face value, have an ethical component, because technical decisions and human values are intertwined. Sociotechnical systems matter to people, and those people are important stakeholders in the systems. Decisions that shape the systems and artifacts matter to people and thus have ethical significance.

Because any technology is best understood as part of a sociotechnical system, and because both the technological artifacts and the systems of which they are a part have ethically significant human decisions embedded in them, these artifacts and systems "have ethics." Such ethics are not identical to the ethics of a person, but they exist. Technologies have ethics; people put them there. And as technologies are developed, those ethics need to be considered by the people who put them there.

ACM Committee on Professional Ethics

Authors Respond:

Any blanket generalization is risky so we accept the argument that one may find an ethical element built into some technologies. Many technologies are, however, sufficiently neutral that they can be used and abused in accordance with human choices, regardless of the intent of the technology developer. The Internet is merely one of many examples. Perhaps the way we can end up in the same place as the Committee is to observe that programmers (and, more generally, technologists) should feel ethical responsibilities in the course of developing new technology to assure it resists accidental or deliberately induced malfunction.

Francine Berman, Troy, NY, and
Vinton G. Cerf, Mountain View, CA

Enough Already with Patent Profusion

I was seriously dismayed by the descriptions of three U.S. design patents in Pamela Samuelson's Legally Speaking column "Supreme Court on Design Patent Damages in *Samsung v. Apple*" (Mar. 2017): "a black rectangular round-cornered front face for the device"; "a rectangular round-cornered front face with a surrounding rim or bezel"; and "a colorful grid of 16 icons to be displayed on a screen." The profusion of such patents is intended primarily to stifle competition rather than protect truly innovative work and has a detrimental effect on computational scientists and the general public alike. These examples represent clear evidence that the U.S. patent system is in serious need of reform.

Nicholas Horton, Amherst, MA

Reengineer Peer Review to Eliminate Reviewer Bias

Elizabeth Varki's Viewpoint "Where Review Goes Wrong" (Mar. 2017) served computer science with its courageous and honest disclosure of struggles with the flawed scholarly peer-review system. As an ACM Fellow with more than 200 publications, I can attest to the problems she identified. I, too, have had papers rejected from venues on the basis of rants from the same reviewers. I was once able to make my case and solicit fresh reviews because carbon copies proved the same typewriter had been used. Digital documents and submission portals now make bias or abuse all but impossible to prove. Reviewing the same paper for more than one publication or conference is unethical, and reviewers should be required to recuse themselves on these grounds.

As someone who has seen the publication process from all sides—author, referee, conference organizer, and editor of multiple journals in multiple disciplines—I can say blind

peer review, the putative gold standard in science, is seriously flawed. Double-blind review is a sham. A reviewer who is current and competent in the subject matter will almost always know who are the authors of a submitted paper, from content, style, reputation, or cited references. Social science research has repeatedly proved double-blind reviewing is a myth. Identical papers submitted under female or ethnic names are more likely to be reviewed unfavorably and rejected. Single-blind review converts to certainty only the probability that authors are disadvantaged.

One issue Varki did not raise is the competence of referees to review a particular paper. As an editor and conference organizer, I know how difficult it is to secure enough capable referees. The more innovative and advanced the paper and author, the more likely reviewers will be less experienced and underqualified. History attests to cases of work that ultimately proved groundbreaking but was repeatedly rejected due to poor reviewing. My own work on coupling and cohesion, which spawned a rich research literature and eventually entered the canon of software engineering, was repeatedly rejected until a fluke opportunity brought it to the world in a journal then at the academic margins. I have seen solid papers by others rejected by reviewers who were self-evidently unqualified to evaluate the paper, even sometimes by their own admission.

It is time to consider re-engineering the entire peer-review process to reflect research evidence from the social sciences and the realities of contemporary academic publishing. Radical though it may seem, a fair process might be an open one without anonymity. So-called anonymous review that is only selectively anonymous leads to abuses and complications. In the deeply incestuous communities of scientific specialties and subspecialties, anonymous reviewing as now practiced is a hidebound fiction that fails in the ultimate purpose of peer review—ensure the quality of the cumulative literature and guarantee fair and open access to all qualified contributors.

Larry Constantine, Rowley, MA

To Inspire Future Engineers, Start at Home

Several fallacies stood out in Gregory Mone's news article "Bias in Technology" (Jan. 2017), which made the tacit but arguable assumption that working in tech has enough social value that getting more women and African-Americans into tech jobs is a laudable goal. Mone said African-Americans represent 1% of the work force at Google and Facebook and 4.6% of students awarded a bachelor's degree in computer science but wondered why only 1% or 2% at "some major companies are African American." Consider that 1% to 2% is actually an extremely high percentage. In his book *Work Rules! Insights from Inside Google That Will Transform How You Live and Lead*, Laszlo Bock, Google's Senior VP of People Operations (what other companies call "human resources") said, "We receive more than two million applications every year. [...] Of these, Google hires only several thousand per year, making Google 25 times more selective than Harvard, Yale, or Princeton. Approximately 0.1% to 0.3% of applicants get jobs at Google. I imagine the numbers are about the same for Facebook, Apple, and Microsoft.

Mone quoted Kaya Thomas, a second-year computer science student at Dartmouth, saying, "If you want to sell to everybody, you have to hire everybody." This statement is wrong in both theory and practice. The design principles that guide Facebook, Apple, and Microsoft to create great products used by billions of people around the world are universal and have nothing to do with affirmative action. Consider, the deeper reason for diversity. Diversity is important in an engineering organization not because of a social agenda or because you want to sell to everybody but because of the value of hiring smart people who think different from you and the opportunity to learn from them and because of the resilience diversity brings to building a great team.

As an engineer who has worked at Intel and as an entrepreneur, I am proud that my startup—Clear Clinica (cloud monitoring of clinical trial data)—has equal representation of men and women, native Israeli, American-born, Russian-born, religious, non-religious,

and ultra-Orthodox. We do this, not because of a social agenda but because diversity makes business sense, helping our team ship great products and survive challenges.

We can also learn an important lesson that goes beyond business sense. American tech companies employ diverse work forces, including every other ethnicity you can think of. Whether or not we agree with the learning ethic of families of any of them, we can agree that the drive to succeed in science and technology starts at home at a young age. I do not pretend to know how to change home values, but if I were looking to encourage people to go into engineering, I would start by encouraging parents to inspire their children to achieve in science, math, and computing.

Danny Lieberman, Modiin, Israel

In Constricting an Art Form, Digitization Can Open It As Well

Esther Shein's news story "Computing the Arts" (Apr. 2017) explored the relation between digitization and the arts. The history of European written music illustrates this development. The historic act of fixing Gregorian chant in a notation that used a seven-note octave during the European Middle Ages could be seen as a constriction of expression, as it eliminated the vitality of diverse vocal pitches in favor of just seven notes. But this particular form of digitization of music also opened the way for polyphony and the intense harmonies of later European music. Meanwhile, mathematics of a different type was behind the practice of perspective in Renaissance art. Digitization can derive insights by looking back at such historic precedents.

Andy Oram, Boston, MA

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

© 2017 ACM 0001-0782/17/06

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.

twitter

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

DOI:10.1145/3077227

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

Balancing Teaching CS Efficiently with Motivating Students

Mark Guzdial suggests a new balance is needed in computer science education between discovery learning and direct instruction.



**Mark Guzdial
'Figure It Out' Isn't
Enough: Striking a New
Balance Between
Understanding,
Problem-Solving, and
Design in Introductory CS Classes**

<http://bit.ly/2jvLDd0>
January 9, 2017

A computing educator has to balance teaching efficiently and motivating the student. Efficient teaching means teaching abstractly, emphasizing practice, and preferring direct instruction over having students “figure it out.” Motivating the student means giving the students authentic situations, real-world complexity, and reasons to practice.

I recently wrote an essay describing this tension (<http://bit.ly/2nFRuGZ>). Herbert Simon (one of the three authors of the *Science* article first answering the question “What is Computer Science?”; <http://bit.ly/2nFIzpf>) strongly believed in direct instruction, rather than problem-solving. Having students “figure out” the solution for themselves, to discover solutions to problems, was an in-

efficient way to learn for Simon, and his concern was for motivation:

When, for whatever reason, students cannot construct the knowledge for themselves, they need some instruction. The argument that knowledge must be constructed is very similar to the earlier arguments that discovery learning is superior to direct instruction. In point of fact, there is very little positive evidence for discovery learning and it is often inferior ... Because most of the learning in discovery learning only takes place after the construct has been found, when the search is lengthy or unsuccessful, motivation commonly flags.

A teacher of introductory computer science faces the tension between letting students figure out complex situations and telling students the answer. I'm going to describe the tension using a generalized, perhaps even stereotypical description of what students want. As Valerie Barr has pointed out (<http://bit.ly/2nFRSFx>), teachers need to understand the students who are in their classes, not generalizations. While the generalization I'm using doesn't accurately describe all students, the gen-

eralization matches how CS teachers think about their students (<http://bit.ly/2nYA3nE>), which does explain what we do in our classes.

A student who takes an introductory computer science course wants to *make* something. Even if the student doesn't want to become a professional software developer, they want to *create* software, to *design* something digital. We want to go from where they are to producing something interesting. The challenge (as Briana Morrison and I describe in the November 2016 *Communications*, <http://bit.ly/2iIFeEc>) is that students enter CS class with less background in the discipline than any other STEM field. It's difficult to design when you don't understand the medium that you're designing with.

We know students need to develop an understanding of what the computer does when it executes programs. Computing education researchers call that the *notional machine* (see the report from a Dagstuhl Seminar group on student learning about notional machines at <http://bit.ly/2oy1KoO>). To design and

debug programs, students need to develop a mental model of the notional machine. We have a poor track record in helping students be able to trace and predict program execution, as Raymond Lister has been exploring in his research (see <http://bit.ly/2n7A8Xr>).

Maybe our students don't develop enough understanding of the notional machine because our introductory courses don't make program understanding a key learning goal. Students *need* to understand, but they *want* to build. We can teach for understanding, but that's harder to do in authentic, complex learning situations, as Simon and others have pointed out (<http://bit.ly/2nFRuGZ>).

Perhaps in response, many introductory computer sciences take a middle ground and focus on *problem-solving* (see examples at <http://bit.ly/2oVyXXu> and <http://bit.ly/2n7yu80>). Some teachers even define computer science as “algorithmic problem-solving” (<http://bit.ly/2nYJhQP>, which doesn't appear at all in the Newell, Perlis, and Simon definition of the field at <http://bit.ly/2nFIzpf>). Teachers can get students to realize they have to solve problems to create software, so they emphasize how to solve problems with programs and algorithms.

A focus on problem-solving is a rational way to strike a balance between getting students to understand programs and their desire to build. We give students problem statements (describing things to build), and we teach them how to go from the problem statements to a working program. We teach them how to design with objects, and how to analyze problems for the data structures within them.

The problem is that teaching problem-solving is not the same thing as teaching for understanding, and empirical evidence suggests it isn't working. John Sweller showed years ago that more problem-solving doesn't lead to greater understanding. Problem-solving creates enormous cognitive load that interferes with learning to understand (<http://bit.ly/2nAufgV>). If we want students to understand more, we have to teach for understanding. In my book [Learner-Centered Design of Computing Education](http://bit.ly/1JYLeUz) (<http://bit.ly/1JYLeUz>), I describe some of the evidence that students are not developing an under-

We need to create learning situations where we ask students to practice program reading, to predict program execution, and to understand program idioms.

standing of programs and developing a mental model of the notional machine.

To teach for understanding, we would give students worked examples and ask them questions about the examples, ask students to predict outcomes or next steps in a visualization (<http://bit.ly/2nAs5xN>), or ask students to solve Parson's Problems. **We would do far less of giving students a problem they've never seen before, and asking them to generate a brand-new program to solve that problem.**

At ICER 2016, Briana Morrison, Lauren Margulieux, and Adrienne Decker presented a replication study showing that introductory students miss important details in problem statements, but they figure them out when the students reach their second CS course (<http://bit.ly/2n7BOjH>). Morrison thinks it takes students that long to develop their understanding so they are more effective at problem-solving. We could perhaps achieve better understanding earlier, but we'd have to teach for understanding. We computer science teachers tend to underemphasize program comprehension, because it's boring for us—and it's easy for computer science teachers. It falls in our expert blind spot. A focus on understanding can be boring for the students, too, because it's not about *making* stuff.

We need a new balance point. We need to do more to get students to understand. They need to build, too, because that's important for student motivation. We need to create learning situations where we ask students to practice program reading, to predict

program execution, and to understand program idioms. More problem-solving might need to wait until student understanding catches up.

Comments

I have to question the assumptions that teaching for understanding is the right goal. There are many aspects in the applications I build that I don't understand. But I don't need to.

“Understanding” is such a slippery goal, whether we talking about understanding computers or Shakespeare. It only becomes well defined when we talk about when you need it, for example, to find a bug, or adapt a solution to a new problem. But then we're back to skills in designing and problem solving as the primary goal.

I do love the notional machine concept though, at least as I “understand” it so far. Which means how I see applying it is thus: to help students when they are struggling in debugging or design, help them articulate and develop a better, partial, locally useful, notional machine.

—Christopher Riesbeck

Hi Chris,

I bet that we would agree that the depth of the understanding is the question. I use my computer all the time without thinking about transistors.

Think about it in terms of Bloom's taxonomy. The lowest level of learning is simply being able to repeat what was heard. Later levels include being able recognize the right thing and to predict. The highest levels are synthesis and problem-solving. Surely those lower levels matter in computer science. We can't ONLY expect students to perform at the highest levels. We also have to teach for those lower levels, too.

—Mark Guzdial

Dear Mr. Guzdial,

Many thanks for your article on CS for all. Being retired, I spare my time with a group “fighting against school dropping out.” We receive students after their school hours and help them doing their homework. It works fine! I'm planning, may be some times from now, to introduce them to CS. Your article inspired me. Again, many thanks!

—Mario Beland

Mark Guzdial is Director of Contextualized Support for Learning at the Georgia Tech College of Computing.

© 2017 ACM 0001-0782/17/06 \$15.00

Deep Learning Takes on Translation

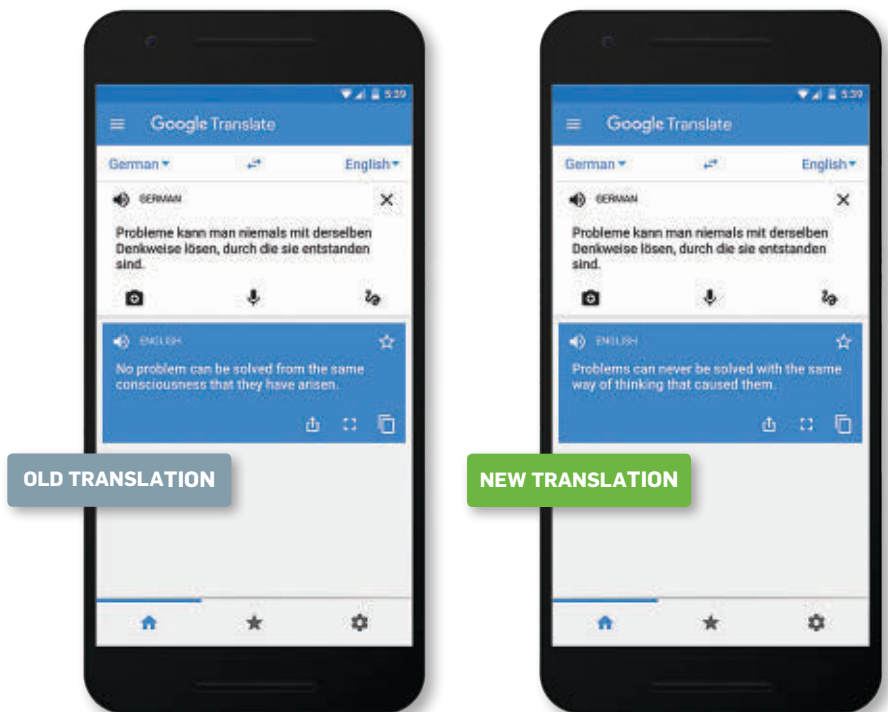
Improvements in hardware, the availability of massive amounts of data, and algorithmic upgrades are among the factors supporting better machine translation.

OVER THE LAST few years, data-intensive machine-learning techniques have made dramatic strides in speech recognition and image analysis. Now these methods are making significant advances on another long-standing challenge: translation of written text between languages.

Until a couple of years ago, the steady progress in machine translation had always been dominated by Google, with its well-supported phrase-based statistical analysis, said Kyunghyun Cho, an assistant professor of computer science and data science at New York University (NYU).

However, in 2015, Cho (then a post-doc in Yoshua Bengio's group at the University of Montreal) and others brought neural-network-based statistical approaches to the annual Workshop on Machine Translation (WMT 15), and for the first time, the "Google translation was not doing better than any of those academic systems."

Since then, "Google has been really quick in adapting this (neural network) technology" for translation, Cho observed. Based on its success,



A demonstration of the improvement in Google Translate thanks to the use of neural machine translation.

last fall Google began replacing the phrase-based system it had used for years, starting with some popular language pairs. The new system is "amaz-

ingly good, and it's all thanks to neural machine translation," Cho said.

Data posted by Google in November 2016 show that its new system is now

essentially as good as human translators, at least for translation between European languages. (Google declined to be interviewed for this story.)

Neural Networks

The architectures that enable these advances, as well as the methods for configuring them, are fundamentally similar to those developed in the late 1980s, says Yann LeCun, director of AI research at Facebook and a professor at NYU. These systems employ multiple layers of simple elements that are extensively interconnected like those in the brain, inspiring the name “neural networks.” Individual elements typically combine the outputs of many other elements with variable weights, and then use a non-linear threshold function to determine their own output.

“To solve a complicated problem, especially if we start with very low-level sensory information, we need to have a very high level of abstraction,” said Cho. “To get that kind of higher-level abstraction, it turns out we need to have many layers of processing.”

In analyzing an image, for example, initial layers might compare nearby pixels to identify lines or other primitive features, while deeper layers flag progressively more complex combinations of features, until the final object is classified as, say, a cat or a tank.

The specific analysis a particular network performs depends on the numerical weights assigned to each interconnection and on the details of how their combined output is computed. The challenge is to set the parameters for a particular task by exposing the system to a large series of inputs whose desired outputs are known, and sequentially adjusting the parameters to reduce any discrepancies.

After such “training” with known examples, the network can quickly extract high-level information from a novel low-level representation. With a large enough set of examples, this training can be done without even specifying which features are needed for classification.

Deep Learning

Neural networks are often described as “deep learning.” The phrase en-

Most translation implementations employ an encoder network and a decoder network that are trained as a pair for each choice of source and target language.

compasses systems that perform more complex functions than traditional “neurons,” but it also sidesteps the somewhat-checked reputation of neural networks.

Several ingredients help explain neural networks’ recent renaissance. First, hardware is vastly superior to that available 30 years ago, including commercial graphical processing units (GPUs) for rapid calculations. “That really brought a qualitative difference in what we can do,” LeCun said. Google has developed hardware it calls tensor processing units, which it says are needed for the rapid results that Web users expect.

A second enabler for deep learning is that researchers can now access enormous amounts of data. Such large data sets, gleaned from our increasingly digital lives, are critical for nailing down the many parameters of deep neural networks.

In addition, although “the basic principles were around 30 years ago, there are a few details in the way we do things now that are different from back then that allow us to train very large, very deep networks,” LeCun said. These algorithmic improvements include better nonlinear functions and methods for regularizing and normalizing input data to help the networks identify the data’s salient features.

Assessing Quality

An ongoing challenge is measuring the quality of a translation or a similar task. “As the problems that we solve get

more and more advanced,” Cho said, “how can we even tell that a model is doing well?”

Even when automated systems are excellent on average, they occasionally can make mistakes that humans find shocking. “We call them stupid errors,” said Salim Roukos, an IBM Fellow at IBM’s Thomas J. Watson Research Center in New York. Fifteen years ago, he and his colleagues developed a computer-based translation metric called bilingual evaluation understudy, or BLEU, which is still widely used and helpful for quality control and tool improvement.

However, BLEU is not sensitive enough to individual misfires, Roukos said. “When we invented BLEU, we thought that in three or five years we would have a better metric, because it has the significant shortcoming which is it’s not very good at the level of a sentence.” Researchers therefore still turn to human evaluators of translation for the ultimate validation, although even humans have weaknesses, such as favoring translations that sound natural even if they are inaccurate.

Turning to Translation

In the past few years, such deep-learning techniques have produced “step-function” improvements for image analysis and for recognition of spoken speech, said Roukos. The improvement for translation, while significant, “has not been as great” so far, he said. “The jury is still out.”

Most implementations of translation employ two neural networks. The first, called the encoder, processes input text from one language to create an evolving fixed-length vector representation of the evolving input. A second “decoder” network monitors this vector to produce text in a different language. Typically, the encoder and decoder are trained as a pair for each choice of source and target language.

An additional critical element is the use of “attention,” which Cho said was “motivated from human translation.” As translation proceeds, based on what has been translated so far, this attention mechanism selects the most useful part of the text to translate next.

Attention models “really made a big difference,” said LeCun. “That’s what everybody is using right now.”

A Universal Language?

The separation of the encoder for one language from the decoder for another language raises an intriguing question about the vector that passes information between the two. As Google put it in a November 2016 blog post, “Is the system learning a common representation in which sentences with the same meaning are represented in similar ways regardless of language—i.e., an ‘interlingua’?” This possibility is reminiscent of the universal language envisioned by 17th-century polymath Gottfried Leibniz for formally denoting philosophical, mathematical, and scientific concepts.

“I would be careful about interpreting things in this sense,” cautioned Yoav Goldberg, a senior lecturer in computer science at Bar Ilan University in Israel. “There definitely is some representation of the sentence that can be used for translating it into different languages,” he said, but “it is a very opaque representation which we cannot understand at all.”

Goldberg suspects the representation is only capturing words and short phrases, “so there is some kind of a shared structure, but I think it’s still kind of a shallow mapping between these languages.”

Cho even imagines a more general “plug-and-play” capability in the future, for example with an encoder for images driving a decoder for language to automatically create captions in any language. “We are a long way from that,” he admitted.

Beyond Europe

Even at a practical level, the separation of encoder and decoder may help solve the problem of inadequate training data for many language pairs. The more than 100 languages supported by Google Translate would require data sets for more than 50,000 language pairs. Training encoders and decoders separately would vastly ease the burden. Google reported “reasonable” translations for two languages that had never been used as a pair for training.

Nonetheless, translations are definitely more difficult between languages from different families. Arabic, for example, relies heavily on word endings to convey meaning, so “the con-

When dealing with languages from different linguistic families, preprocessing of the text can significantly improve translation.

cept of a word” is not the same as in English, Roukos said. Goldberg noted the challenge arising from free word order in Hebrew. In these cases, preprocessing of the text, for example dividing Arabic words into multiple segments to facilitate mapping between languages, can significantly improve translation.

Even at a practical level, the separation of encoder and decoder may help solve the problem of inadequate training data for many language pairs. The more than 100 languages supported by Google Translate would require data sets for more than 50,000 language pairs. Training encoders and decoders separately would vastly ease the burden. Google reported “reasonable” translations for two languages that had never been used as a pair for training.

In written Chinese, noted Victor Mair, a professor of Chinese language and literature at the University of Pennsylvania, there are no spaces separating words. In addition, he said that Chinese writing frequently includes words from Cantonese and other variants, as well as “classicisms” that may contain archaic vocabulary and grammar. “It confuses the computer no end,” Mair said, but he still finds the translation tools useful.

The new Google is “really pretty good” between Chinese and English, but Mair suspects that “there are some things they will never get right.”

Revolutions in Progress

In view of recent progress, however, “never” may not be so far away. Newer systems do not require spaces be-

tween words, Cho said, and “neural machine translation is making it easier for us to build a translation system” that makes use of the internal structure of complex Chinese characters, or the components of a compound German word.

A remaining question is how good neural systems can get at translation without exploiting traditional expert knowledge. “Knowing something about language in general, or properties of linguistic structure, definitely does help in the translation,” Goldberg said. “We shouldn’t be oblivious to the concepts of linguistics or expect them to be learned on their own.”

But the powerful new systems are continually challenging the importance of expertise.

Based on his experience with neural networks going back to Bell Labs in the 1980s, LeCun cited a fourth reason for the upswing in deep learning, besides fast computers, big data, and new ideas: “People started believing.” The reputation of neural nets as finicky, he said, began to break down with widespread sharing of code and methods, and a steady stream of solid results.

“First in speech recognition, then in image processing, and now in natural-language understanding and translation in particular, it really is a revolution,” LeCun said. “It’s an incredibly short time” for going from blue-sky research to industry standard. “All the large companies that have big language translation services are basically using neural nets.” ■

Further Reading

Byrne, M.

This DARPA Video Targeting AI Hype Is Necessary Viewing, *Motherboard*, https://motherboard.vice.com/en_us/article/this-darpa-video-targeting-ai-hype-is-necessary-viewing

A Neural Network for Machine Translation, at Production Scale, *Google Research Blog*, <https://research.googleblog.com/2016/09/a-neural-network-for-machine.html>

Zero-Shot Translation with Google’s Multilingual Neural Machine Translation System, *Google Research Blog*, <http://bit.ly/2nc469z>

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

© 2017 ACM 0001-0782/17/06 \$15.00

Optimization Search Finds a Heart of Glass

Analog computing could provide greater efficiency, improved digital algorithms.

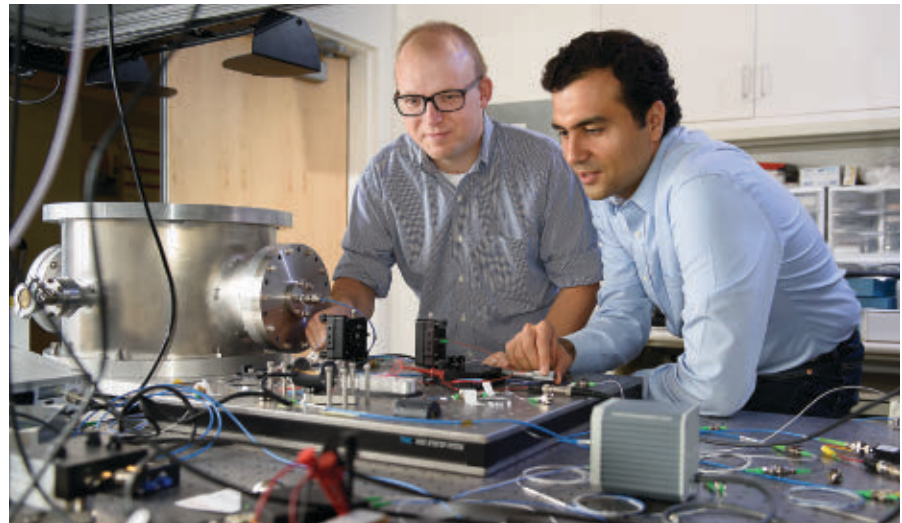
A 20TH-CENTURY THEORETICAL model of the way magnetism develops in cooling solids is driving the development of analog computers that could deliver results with much less electrical power than today's supercomputers. But the work may instead yield improved digital algorithms rather than a mainstream analog architecture.

Helmut Katzgraber, associate professor at Texas A&M in College Station, TX, argues, "There is a deep synergy between classical optimization, statistical physics, high-performance computing, and quantum computing. Those things really go hand in hand. Nature is the best optimizer out there. Lightning typically chooses the path of least resistance. A soap bubble will always give you the minimal surface."

"Maybe we should go back to what we did in the 1930s and 1940s: we built special-purpose computers," adds Katzgraber. The problem with such alternative architectures, he notes, is that the technology for such analog machines is at the level of digital computing in the 1940s. "Today we only have very limited machines and limited experience."

One physical process that lies at the heart of a number of experimental machines is the annealing of a spin glass. The focus of a model developed by physicist Ernst Ising in the 1920s, a spin glass represents the highly disordered state of a hot magnetic material. In the initial state, the spins are not aligned with each other, but as the material cools, an annealing process leads to the spins slowly becoming aligned as the spins of individual atoms flip up and down.

In Ising's model, a cost matrix that links each simulated atom to each of the others influences how the spins will align. The overall energy is captured in a Hamiltonian, a mathematical operator used in quantum me-



Stanford University visiting researcher Alireza Marandi (right) and post-doctoral scholar Peter McMahon inspect a prototype of a new light-based computer.

chanics. The operator represents the sum of individual energy states in the system. A prototype for one special-purpose computer that uses the Ising model is a long loop of fiber-optic cable in the E.L. Ginzton Laboratory at Stanford University in California.

The photonic machine at Stanford finds "approximate or exact solutions to the Ising problem," says Stanford University researcher Peter McMahon. "Phrased this way, it sounds very restrictive. It's more general than that one specific problem: it can be used for quadratic binary optimization problems, but it's not so general that it can solve any optimization problem."

The key is to develop a cost matrix that encapsulates the parameters to be optimized, and use that to drive the Hamiltonian to its lowest energy state. In principle, the physics-based simulation can find optimal solutions to certain problems in many fewer steps than classical digital techniques, but it is far from clear that the analog accelerators will prove to be faster than their digital counterparts.

The machine at Stanford is far from the first to attack optimization problems by emulating the behavior of Ising spin glasses. D-Wave Systems, Google, Microsoft, and a number of universities and computing companies are focusing on designs that use quantum effects to drive the annealing process.

Explains McMahon, "Quantum annealers are types of Ising machine. The leading quantum-annealing technologies are these superconducting circuits that use magnetic flux and circulating current rather than photons as the information in their systems. It's now a fairly advanced technology compared to ours: D-Wave has been around for almost 20 years now."

The major problems facing builders of computers that use the Ising model are scale and connectivity. Without scale, digital computers will easily outpace quantum annealers and other machines that exploit spin-glass behavior. At the beginning of 2017, D-Wave unveiled a machine that doubles the number of qubits to several thousand. Yet the D-Wave design is limited in terms of connectivity.

In the Ising model, the spin of each particle can couple to the spin of any of the others. The second-generation D-Wave can couple the spins of fewer than 10 qubits within each cluster. These clusters are aggregated into a tree-like structure called Chimera.

In 2008, Vicky Choi, then at D-Wave and now an assistant professor at Virginia Polytechnic Institute and State University, showed how problems could be split into groups that could be mapped onto the Chimera architecture. The cost was the use of more qubits to represent a problem than would be needed in a fully connected architecture.

McMahon says the photonic approach he works on has a fundamental advantage in terms of connectivity. “We’ve come up with a scheme that allows us to connect every spin to every other spin. We have built systems that are much larger than what people have been able to do before, in our case with up to 100 spins.”

In the photonic machines built by McMahon and colleagues at Stanford, and in a parallel experiment at NTT’s Basic Research Laboratories in Japan, photonic pulses pass a detector on each pass through the fiber. An electrical circuit tracks the state of each pulse and uses its stored cost matrix to apply a feedback signal that attempts to flip the phase of the photon, rather than its spin. “We only use the word ‘spin’ because the Ising Hamiltonian is related to spin,” McMahon says. “We don’t provide massive feedback to get an answer. We do it slowly over 100 or so round trips in a form of annealing that’s not quantum annealing. But the way in which we carry out the computation is very similar.”

McMahon says groups working with superconducting devices are now looking at similar architectures based on microwave-frequency photonics at chip rather than fiber scale, to build fully connected architectures.

There are other approaches to full connectivity. A group from The Institute of Photonic Sciences (ICFO) in Barcelona, Spain, developed a quantum annealer with full coupling. The machine uses trapped ions manipulated by lasers.

Says ICFO researcher Tobias Grass, “One can think of trapped ions as a lattice of spins. By shining light onto an

ion, it is possible to flip the spin and at the same time create or annihilate a lattice vibration.”

The lattice motion can be modeled using phonons, particles that transfer vibrations. Grass continues, “Once created, this phonon travels through the lattice and might at some point be absorbed together with another spin flip at a different position. Since these phonons represent a collective motion of the lattice, they travel through the whole system, and therefore can couple any pair of spins.”

The fully coupled architectures have their own scaling limits. Grass says the mutual repulsion between ions makes scaling to 100 qubits difficult using today’s techniques. McMahon says the photonic architecture can increase the size of the system to 10,000 elements—each one represented by a photon travelling around the loop of fiber. “Maybe you could really push it and get to 50,000 or 100,000, but 10,000 is where it looks to get prohibitively expensive,” he notes.

Even with highly connected architectures, the performance of Ising machines varies depending on the problem as well as the connectivity. Katzgraber points to tests in which the couplings defined in the cost matrix were altered to gauge their effect on the computability of the problem.

“As the numbers were changed, the problem became much easier. It seems that the values of the interactions matter more than the architecture,” Katzgraber says. Changing from a well-connected structure to one based on the Chimera architecture might make it harder to solve some problems, he adds. “You want to have as many connections as possible because that will allow you to work on many different problems. But you can get pathological graphs; they get stuck.”

One way researchers at Google believe it may still be possible to harness their work on quantum annealing in the face of problem-specific hurdles is to use digital computers to decide when it makes sense to hand off a computation to the analog accelerator. However, the multiple research projects may simply find that classical techniques are more likely to outperform annealing-based machines.

The U.S. Defense Department’s In-

telligence Advanced Research Projects Activity (IARPA) is funding some of the academic work to determine the most likely path of progress. Katzgraber says, “The goal of the IARPA program is not to build a working quantum computer but to prove beyond reasonable doubt if quantum annealing has a practical use or not. The goal is to either bury it or invest in it.”

The result of the work may use experience derived from annealing-based techniques to develop algorithms for classical computers, rather than new analog or quantum architectures. That has already happened in the field of satisfiability (SAT) solvers, Katzgraber says. Such a solver determines whether a Boolean formula made up of AND, OR, and NOT gates can be satisfied. It is possible to reframe the problem in a way that can be solved, on a small scale today, by quantum annealing.

“Satisfiability used to be an application for the IARPA program. We developed an approach to building SAT filters that’s so efficient that doing it in quantum devices will be a waste of time,” Katzgraber claims. “Every time quantum shows a success, classical tries to outperform it. You will see similar innovations driven by this.” **□**

Further Reading

Bainbridge, L.

Ironies of automation. New Technology and Human Error, J. Rasmussen, K. Duncan, J. Leplat (Eds.). Wiley, Chichester, U.K., 1987, 271–283.

Albash, T., and Lidar, D.

Adiabatic Quantum Computing, ArXiv Preprints. ArXiv:1611.04471 (2016)

McMahon, P., Marandi, A., et al.

A fully-programmable 100-spin coherent Ising machine with all-to-all connections. Science. 10.1126/science.aah5178 (2016)

Grass, T., Raventós, D., Juliá-Díaz, B., Gogolin, C., and Lewenstein, M.

Quantum annealing for the number-partitioning problem using a tunable spin glass of ions. Nature Communications. 10.1038/ncomms11524 (2016)

Katzgraber, H.G., Hamze, F., Zhu, Z., Ochoa, A.J., and Muñoz-Bauza, H.

Seeking Quantum Speedup Through Spin Glasses: The Good, the Bad, and the Ugly. Physical Review X. 5, 031026 (2015)

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

© 2017 ACM 0001-0782/17/06 \$15.00

Potential and Peril

The outlook for artificial intelligence-based autonomous weapons.

THE HISTORY OF battle knows no bounds, with weapons of destruction evolving from prehistoric clubs, axes, and spears to bombs, drones, missiles, landmines, and systems used in biological and nuclear warfare. More recently, lethal autonomous weapon systems (LAWS) powered by artificial intelligence (AI) have begun to surface, raising ethical issues about the use of AI and causing disagreement on whether such weapons should be banned in line with international humanitarian laws under the Geneva Convention.

Much of the disagreement around LAWS is based on where the line should be drawn between weapons with limited human control and autonomous weapons, and differences of opinion on whether more or less people will lose their lives as a result of the implementation of LAWS. There are also contrary views on whether autonomous weapons are already in play on the battlefield.

Ronald Arkin, Regents' Professor and Director of the Mobile Robot Laboratory in the College of Computing at Georgia Institute of Technology, says limited autonomy is already present in weapon systems such as the U.S. Navy's Phalanx Close-In Weapons System, which is designed to identify and fire at incoming missiles or threatening aircraft, and Israel's Harpy system, a fire-and-forget weapon designed to detect, attack, and destroy radar emitters.

The Campaign to Stop Killer Robots, which was founded in 2013 by a group of regional, national, and international non-governmental organizations (NGOs), agrees that no fully autonomous weapons are yet in use, but says existing systems could soon be extended to become fully autonomous and that the window to fulfill its ambition of achieving a preemptive ban on all such systems is closing.

The campaign's key tenet is that giving machines the power to decide



Participants in the first NGO Conference of the Campaign to Stop Killer Robots in London in 2013.

who lives and dies on the battlefield is an unacceptable application of technology and makes human control of any combat robot essential to ensuring humanitarian protection.

Mary Wareham, global coordinator of the Campaign to Stop Killer Robots at Human Rights Watch in Washington D.C., explains the potential of precursor weapon systems that could be extended, exemplifying armed drones. Says Wareham, "Remotely piloted armed drones still have a human in the loop deciding on the selection of targets and force to be used, but new generations of arms could fly autonomously and complete missions with no human control. We don't want to see these systems in action."

From a robotic perspective, Arkin also defines autonomous machines as those that have no opportunity for human intervention, but says one of the concerns around LAWS is that there is no substantive agreement on what

they constitute. He does not advocate a total ban on LAWS, but suggests the need to look at specific instances of weapons and decide on a one-by-one basis whether they are viable or should be banned. He explains: "I am a proponent of a moratorium until there is more understanding of autonomous weapons. We all agree we don't want a scenario like *The Terminator*, but we do need to talk about what we do want."

Arkin argues that a better understanding of autonomous weapons could lead to the development of intelligent autonomous military systems that could be precise in hitting targets and, at the same time, reduce civilian casualties and property damage when compared to the performance of human fighters, whose behavior in the theatre of war can be inconsistent and waver between heroic and atrocious.

Says Arkin, "We need to assume more responsibility for non-combat-

ants, and not shoot first and ask questions later. In some circumstances, autonomous weapons could comply better with international humanitarian law than humans. But if weapons can't do as well as human fighters, they should not be put in place, hence my view on a moratorium."

With countries including the U.S., U.K., China, Russia, and South Korea developing autonomous weapons, and the U.K. Ministry of Defence estimating in 2011 that AI-based systems, as opposed to complex and clever automated systems, could be achieved in five to 15 years and that fully autonomous swarms of weapons such as drones could be available by 2025, the Campaign to Stop Killer Robots goes a step further than Arkin in its call for a pre-emptive ban on all autonomous weapons. It is pressing for the ban to be enacted through the implementation of international legislation or a new protocol under the Convention on Certain Conventional Weapons (CCCW), the key U.N. vehicle promoting disarmament, aiming to protect military troops from inhumane injuries, and seeking to prevent non-combatants from accidentally being wounded or killed by certain types of arms.

The most recent weapons to be excluded from warfare under the CCCW treaty are blinding lasers, which were banned in 1995.

The campaign defines three types of robotic weapons: human-in-the-loop weapons, robots that can select targets and deliver force only with a human command; human-on-the-loop weapons, robots that can select targets and deliver force under the oversight of a human operator who can override the robots' actions; and human-out-of-the-loop weapons, robots that are capable of selecting targets and delivering force without any human input or interaction. While these definitions are commonly used among developers of AI-powered weapons, their definitive meanings have yet to be agreed upon.

Reporting on a February 2016 roundtable discussion on autonomous weapons, civilian safety, and regulation versus prohibition among AI and robotics developers, Heather Roff, a research scientist in the Global Security Initiative at Arizona State University with research interests in the ethics of emerging military technologies, international hu-

manitarian law, humanitarian intervention, and the responsibility to protect, distinguishes automatic weapons from autonomous weapons. She describes sophisticated automatic weapons as incapable of learning, or of changing their goals, although their mobility and, in some cases, autonomous navigation capacities mean they could wreak havoc on civilian populations and are most likely to be used as anti-material, rather than anti-personnel, weapons.

Roff describes initial autonomous weapons as limited learning weapons that are capable both of learning and of changing their sub-goals while deployed, saying, "Where sophisticated automatic weapons are concerned, governments must think carefully about whether these weapons should be deployed in complex environments. States should institute regulations on how they can be used. But truly autonomous systems—limited learning or even more sophisticated weapons—ought to be banned. Their use would carry enormous risk for civilians, might escalate conflicts, and would likely provoke an arms race in AI."

Toby Walsh, professor of AI at the University of New South Wales, Australia, says, "There are many dangers here, not only malevolence, but also incompetence, systems designed by those with malicious intent, or systems that are badly made. Today, the military could develop, sell, and use stupid AI that hands responsibility to weapons that can't distinguish between civilians and combatants. The technology is brittle and we don't always know

"There are many dangers here, not only malevolence, but also incompetence, systems designed by those with malicious intent, or systems that are made badly."

how it will behave, so the last place to put AI systems that are trained on data in the environment is the battlefield, which is already a chaotic place.

"The real challenge is ensuring good outcomes of AI, but unexpected outcomes could be good or bad, and that is for us to decide."

Other perils identified by AI researchers in this space include unilateral use of autonomous weapons to support asymmetric warfare, the potential unpredictability of weapon behavior particularly where multiple systems interact as swarms, and the unimaginable human and material destruction that could result from terrorist use of such weapons.

Looking at the ethical issues of LAWS, Eric Schwitzgebel, professor of philosophy at University of California, Riverside with research interests in philosophy of mind and moral psychology, discusses AI-based systems as objects of moral concern and questions whether AI could become sophisticated enough to be conscious.

Schwitzgebel acknowledges that such a scenario is unlikely in the short term, but says it could be possible to create an autonomous system capable of experiencing joy and suffering at a similar level to a human. If such a system were to be sent to war and "die," he suggests this may not be morally different to the case of a human who is sent to war and dies, as the system was human enough that it would not want this to happen. Similarly, Schwitzgebel notes that if a system was sent to war against its will, this would be the moral equivalent of creating slaves and sending them to war.

Says Schwitzgebel, "We haven't thought through carefully what sorts of AI systems we need and don't need to be concerned about, and the differences between them and us that would make them morally different. Hypothetically, an artificial being could be created with moral rights and the capacities of a person. This sort of AI will not be developed any time soon, but development could go in this direction and should be stopped short of getting there."

Schwitzgebel cites more immediate dangers of deploying autonomous intelligences in combat as loss of responsibility and lack of predictability. The loss of responsibility for autonomous

systems in combat makes it difficult to apportion blame when something goes wrong. The question of whether the weapon designer, deployer, or indeed, any other entity should take the blame is far from answered. Schwitzgebel suggests the diffusion of blame could be consistent with governments collaborating on undesirable uses of autonomous systems.

Lack of predictability could also become a more serious threat as AI systems become more complex. “Human soldiers in warfare can be unpredictable, but within limits as military commanders have an understanding of what has happened in various conditions in the past,” says Schwitzgebel. “Autonomous systems could be more unpredictable than humans, which in warfare could lead to disastrous consequences. The ethics of autonomous weapons and issues of AI and philosophy are not as widely talked about as they should be.”


With many questions about the benefits and dangers of LAWS still up in the air, and no international agreements in place to provide answers, the Campaign to Stop Killer Robots and other research organizations keen to ensure a ban on their development, manufacture, and deployment, are exerting pressure on governments to adopt and implement their approach.

While the campaign is concerned that the window of time to reach agreement on a ban on LAWS is closing as autonomous weapons are being developed, progress in its favor is beginning to be made and autonomous weapons are moving up the agenda following a December 2016 United Nations review of the CCCW.

Making a small piece of history, the U.N. voted during the review to start a formal process that might lead to a ban on LAWS. Of course, there are no guarantees that the process will be successful, but as Walsh puts it: “If states hadn’t voted to start the process, there would have been no chance to finish.” Russia abstained from the vote.

Countries participating in the vote agreed to set up an open-ended Group of Governmental Experts that will discuss nations’ concerns about LAWS and the line between autonomous and non-autonomous weapons. The group will meet for two

“Autonomous systems could be more unpredictable than humans, which in warfare could lead to disastrous consequences.”

weeks in August this year, but the expectation is that it will take multiple years to reach consensus and add a protocol to the CCCW that will ban autonomous weapons operating beyond the boundaries of international humanitarian law. 

Further Reading

Losing Humanity: The Case against Killer Robots
November 2012, Human Rights Watch and International Human Rights Clinic at Harvard Law School
<https://www.hrw.org/report/2012/11/19/losing-humanity/case-against-killer-robots>

Views of the International Committee of the Red Cross (ICRC) on autonomous weapon systems
April 2016, ICRC
<https://www.icrc.org/en/document/views-icrc-autonomous-weapon-system>

Three in Ten Americans Support Using Autonomous Weapons
February 2017, Ipsos
<http://www.ipsos-na.com/news-polls/pressrelease.aspx?id=7555>

IEEE Ethically Aligned Design Document Elevates the Importance of Ethics in the Development of Artificial Intelligence (AI) and Autonomous Systems (AS)
December 2016, IEEE
http://standards.ieee.org/news/2016/ethically_aligned_design.html

Lethal Autonomous Systems and the Plight of the Non-combatant
July 2103, Ronald Arkin, Georgia Institute of Technology
<http://www.cc.gatech.edu/ai/robot-lab/online-publications/aisbq-137.pdf>

Campaign to Stop Killer Robots
<https://www.stopkillerrobots.org/>

Sarah Underwood is a technology writer based in Teddington, U.K.

© 2017 ACM 0001-0782/17/06 \$15.00

ACM Member News

DETERMINING NORMS FOR CYBER WARFARE



Patrick McDaniel, a Distinguished Professor in the School of Electrical Engineering

and Computer Science at Pennsylvania State University (Penn State), says that when he was 11 years old, his father brought home a TRS-80 portable computer from Radio Shack, and handed him the manual to BASIC. “Within 10 minutes I was addicted, and I have never looked back. I have bachelor’s, master’s, and Ph.D. degrees in computer science, and it has never even been a thought to do anything else.”

McDaniel obtained his undergraduate degree at Ohio University in 1989, and his master’s degree at Ball State University in 1991. He then worked to develop some of the first IP networking hardware as a project manager at Primary Access Corp. in San Diego, which was acquired by 3Com in 1995.

He later earned his Ph.D. in computer science and engineering at the University of Michigan, Ann Arbor. McDaniel spent several years as a senior research staff member at AT&T Labs in New Jersey, before joining the faculty at Penn State in 2004.

McDaniel is director of the Institute for Networking and Security Research at Penn State, and also university lead for the U.S. Army Cyber Security Research Alliance, a 10-year project to develop an understanding of how to make security-relevant decisions in cyberspace.

One area McDaniel is focused on concerns the norms for international cyber warfare. He works to help define and set standards for what is allowable; in effect, a Geneva Convention for cyber warfare. “Right now, because nothing is set up, it is really hard to go to the U.N. Security Council for sanctions when you haven’t set up any norms.”

—John Delaney



Weaving the Web

Sir Tim Berners-Lee created a paradigm shift that changed the world with his invention of the World Wide Web, Hypertext Transport Protocol, and Hypertext Markup Language.

WHEN HE FIRST came up with the idea for the World Wide Web in 1989, Sir Tim Berners-Lee had trouble getting people to grasp the concept. If he gave a lecture to a room of 100 people, demonstrating how his browser/editor could jump from one document to another when he clicked on a hypertext link, he recalls, the response would be a collective “So what?”

“Maybe two or three at the back would get it. Most people wouldn’t,” says Berners-Lee.

Hypertext was not new. CD-ROMs had links that allowed navigation from one page of, say, an encyclopedia to another, but “people didn’t understand the power of the link if it could link to everything conceivable,” he says. “That’s a paradigm shift, that if you click on it, it can go to anything on the planet.”

It is for creating that paradigm shift—by inventing the World Wide Web, the URL naming scheme, the HTTP protocol, and the HTML markup language—that Berners-Lee has been designated to receive the 2016 Turing Award in June, the 50th time the prize will have been bestowed.

The Web has become so fundamental, he says, that it has become as difficult to imagine the Web did not exist, as it once was to imagine that it should. “The paradigm shift is impenetrable both ways,” Berners-Lee says. “It was impossible to explain to people what the Web would be like then, and now when you talk to millennials they can’t understand what the problem was.”

Berners-Lee was working at CERN, the European Organization for Nuclear Research, in Geneva at the time. He had an undergraduate degree in physics from The Queen’s College, Oxford, but no formal training as a computer scientist, although

he had built his own computer and written software, and it was that combination of skills CERN needed.

He created the Web, in large part, to make his own life easier. There were perhaps 10,000 people working for CERN at the time, he says, but only about 3,000 on the actual campus; others were coming and going between there and other institutions. Berners-Lee thought it would be useful to have an online collaborative space where people could share ideas, and where people who came along later could follow the decision-making process by clicking through the links.

However, just bringing co-workers together did not seem like enough. The Web, he believed, should allow anybody anywhere to create information and link to it.

By 1989, the Internet was beginning to become generally connected, and Berners-Lee felt that linking everything to everything would spur users’ creativity. “I’d been harping on about joining all information together for ages,” he says. “What was critical at that point was that my boss finally let me just do it as a side-project.”

That boss, Mike Sendall, could not justify the project as having a direct relation to CERN’s goals. Instead, he decided it could be a good way

It has become as difficult to imagine the Web did not exist, as it once was to imagine that it should.

of testing the potential of the NeXT machine, a new computer architecture designed by Steve Jobs.

There were many types of computer systems in existence at the time—documentation systems, help systems, note-taking systems, paper-publishing systems—but each tended to focus on just one type of information. Berners-Lee wanted to break down the separation between those systems: “I realized the Web had to be universal. It had to be completely without attitude about what you were doing with it.” It also had to work no matter the type of computer, the programming language, or the language or culture of the user.

His one requirement was that everybody in the world label everything they had with what he originally called a UDI, or universal document identifier. That would later become known as a uniform resource locator (URL), and is now becoming a URI, uniform resource identifier. “That’s a very big ‘ask,’ so you can’t ask anything else,” he says.

Having made that “ask,” he then set out to make everything else about his system easy to swallow, which led to a set of fairly arbitrary design choices as he was creating Hypertext Markup Language, such as deciding whether to use round or square brackets and which type of slash to use.

“Whether those slashes were forward slashes or back slashes didn’t affect how the Web worked,” he says, “but it does affect how other developers react to it, so the trick was to use design languages that they already used.”

Berners-Lee made Hypertext Transfer Protocol (HTTP) look like other Internet protocols, such as Simple Mail Transfer Protocol (SMTP) and Network News Transfer Protocol (NNTP). He designed Hypertext Markup Language (HTML) to look like Standard Generalized Markup

Language (SGML). He felt it would be more logical if domain names were listed in descending order of hierarchy—org.acm.cacm, for instance—but the Domain Name System already existed, so he took it as it was.

In 1990, Paolo Palazzi, a physicist at CERN, tried to tempt Berners-Lee into joining his Programming Techniques Group. Instead, Berners-Lee showed him the Web project he was working on. Palazzi says he saw the problem Berners-Lee was trying to solve, but did not fully understand his proposed solution. Nonetheless, he thought the project was worthy of support. “Truly innovative ideas cannot really be grasped, so you have to trust the person proposing it,” he says.

Berners-Lee was someone he trusted. “He had a special way of going about solving problems,” Palazzi says. “Tim is one of this class of people who have peculiar abilities at inventing or discovering.”

It was the way he combined existing ideas—hypertext and Internet protocols—that was innovative, Palazzi says. HTML, HTTP, and URIs were inventive in themselves, but, Palazzi says, “The way they combine together is, I think, a stroke of genius.”

By 1994, the Web had progressed from a small research project to a global phenomenon, with companies such as IBM adopting it and new companies such as Netscape, the first browser company, being created. Berners-Lee moved to the computer science department at the Massachusetts Institute of Technology (MIT) and founded the World Wide Web Consortium, an international group that developed standards for the Web. In 2008, he was named the 3Com Founders Professor of Engineering at MIT. In 2016, he became a professor in the computer science department at Oxford University, although he is also still working at MIT.

The Turing Award is the latest in a long list of honors recognizing the work of Berners-Lee. In 2004, he was knighted by Queen Elizabeth, who also awarded him the Order of Merit in 2007. He is a fellow or member of many professional organizations, including the Royal Society and the National Academy of Sciences, and has been given medals by groups ranging from the

The World Wide Web Foundation “was about recognizing that there was a duty that the haves have to the have-nots, to try and get as many people to have as much access as possible.”

Institute of Physics (IOP) to the United Nations Educational, Scientific, and Cultural Organization (UNESCO).

The Turing Award includes a prize of \$1 million, with financial support provided by Google. Berners-Lee has not yet made plans for what he will do with that sum.

In 2008, Berners-Lee founded the World Wide Web Foundation, a nonprofit organization promoting access to the Web for all. “That was about recognizing that there was a duty that the haves have to the have-nots, to try and get as many people to have as much access as possible,” he says. The Alliance for Affordable Internet, a project of the Foundation, seeks to drive down the price of broadband access so people in developing countries can access the Web.

Open access continues to be an issue for the Web, Berners-Lee says. “Whenever people ask me the question, ‘What’s your biggest fear?’, it’s always been that some one entity, either commercial or political, should control the Web. That would be the death of it.” Net neutrality is important, he says; a service provider should not be able to control what content its customers see.

He also worries about governments either blocking access to certain Websites, or worse, using the Web to track which websites users visit, then punishing people based on that information.


“It’s also something which countries like America and the U.K. have to be very careful with, make sure they don’t slip into the fear of terrorism. The war on terrorism is being used as an excuse for many things, but one of them can be taking away people’s fundamental rights to communicate.” He cites recent calls by the U.K. government, in the wake of the terror attack in London in March, to be given back-door access to applications such as Whatsapp for fear terrorists are using them to coordinate attacks.

At MIT, Berners-Lee is co-director of the Decentralized Information Group, which is working, in his words, on “re-decentralizing” the Web to burst some of the “filter bubbles” people have created for themselves on social media.

“Because the Internet didn’t have countries as a thing, [some people] hoped that people would learn to just break down cultural barriers and it would lead to love and understanding across borders, and world peace. And it didn’t,” he says. The group is pursuing the notion that perhaps the right software could help realize that utopian vision.

Another project Berners-Lee currently is working on seeks to give people greater control over their data, such as where it is stored and what other people and applications have access to it.

For young computer scientists looking to have an impact on the world, Berners-Lee recommends ignoring conventional wisdom and following their own instincts.

“You should feel free to develop something for yourself, because it seems to appeal to you, scratches an itch that you have,” he says. “To a certain extent, one has to beware of asking the users what they want, because the things which they would find really exciting, they can’t imagine.” 

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

© 2017 ACM 0001-0782/17/06 \$15.00



Watch the Turing recipient discuss his work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/turing-award-recipient-sir-tim-berners-lee>

Blog Ubiquity

INFORMATION EVERYWHERE



The newest ACM forum.

Contributions that cover the vast information-rich world where computing is embedded everywhere.

ACM's *Ubiquity* is the online magazine oriented toward the future of computing and the people who are creating it.

We invite you to participate: leave comments, vote for your favorites, or submit your own contributions.

Captivating topics.

*Net Neutrality
and the Regulated
Internet*

*The End of Life
As We Know It*

*A Shortage of
Technicians*

*The Fractal
Software
Hypothesis*

*Your Grandfather's
Oldsmobile—NOT!*

*Superscalar
Smart Cities*



Association for
Computing Machinery

Visit us at
<http://ubiquity.acm.org/blog/>

Big Data

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

For our fourth and final Panel in Print, we invited 2014 ACM A.M. Turing Award recipient **MICHAEL STONEBRAKER**, 2013 ACM Prize recipient **DAVID BLEI**, 2007 ACM Prize recipient **DAPHNE KOLLER**, and ACM Fellow **VIPIN KUMAR** to discuss trends in big data.

Gartner estimates that there are currently about 4.9 billion connected devices (cars, homes, appliances, industrial equipment, among others) generating data. This is expected to reach 25 billion by 2020. What do you see as some of the primary challenges and opportunities this wave of data will create?

VIPIN KUMAR: One of the major challenges we are going to see is that the data being gathered from these connected devices and sensors is very different from other datasets that our big data community has had to deal with.

The biggest successes we have seen for big data are in applications such as

Internet search, e-commerce, placement of online ads, language translation, image processing, autonomous driving. These successes have been enabled, to a great extent, by the availability of large, relatively structured datasets that can be used to train a broad range of machine learning algorithms. But the data from multitudes of interconnected devices in its raw state, can be highly fragmented, disparate in space and time, and very heterogeneous. Analyzing such data will be a big and new technological challenge for the machine learning and data mining communities.

DAVID BLEI: The key idea here is that just the data from something as simple as Netflix watching habits doesn't provide the recommendation of a new movie; it's that data alongside all the data from everybody else that helps make recommendations.

It's an exciting world because we are personalizing our interaction with devices through the aggregate data of everybody using their devices. Of course, this all comes with a challenge around privacy and what we give up when we make our data available or the spectrum of how much we can give up against how much personalization power we get in return.

The other opportunity is in an unprecedented way to learn about the world through these huge collections of many individuals. This is a massive dataset, and patterns of communica-

tion, interaction, and movement—including all types of other macro-level descriptions of society and people and the world—are now available to us.

As more data is collected from a growing pool of devices, has the individual lost the right to information privacy?

MICHAEL STONEBRAKER: Imagine this simple example: you show up at your doctor's office and have an x-ray done and you want the doctor to run a query that shows who else has x-rays that look like yours, what was their diagnosis and what was the morbidity of the patients. That requires integrating essentially the country's entire online medical databases and presumably would extend to multiple countries as well. While that is a daunting data integration challenge, because every hospital chain stores its data with different formats, different encodings for common terms, etc., the social value gained from solving it is just huge. But that also creates an incredibly difficult privacy problem, one that is not a technical issue. Because if you're looking for an interesting medical query, you're not looking for common events; you're looking for rare events, and at least to my knowledge, there aren't any technical solutions that will allow access to rare events without indirectly disclosing who the events belong to.

I view the privacy problem to be basically a legal problem. We have to have legal remedies in this area.



David Blei



Daphne Koller



Vipin Kumar



Michael Stonebraker

There are tons of examples of data that can be assembled right now that will compromise privacy. Unfortunately, the social value to compromising privacy is pretty substantial. So, you can argue that technology has rendered privacy a moot question. Or you can argue that preserving privacy is a legislative issue.

As predictive models are increasingly used, how do we avoid biases when interpreting and using data?

DAPHNE KOLLER: Bias will always be a challenge, and there isn't a single, magic solution. The bigger question is: "How do we disentangle correlation from causation?" The gold standard in medicine is that of randomized case control. In the case of Web data, it's called AB testing. Although not perfect, randomized case control, or AB testing, is about as good a tool as we have been able to develop for addressing some of the confounders. Unfortunately, this type of control is not feasible in all cases. Then processes must be carefully scrutinized to check for different confounders and to look for any and all correlations that give rise to the phenomenon being viewed. It's a process that requires a lot of thought and a lot of care and cannot be overstated in its importance.

For example, sometimes there are biases that are reflected in the conclusions that are drawn from the data. In searches on certain sites for example, "Steph" auto-completes to "Stephen" rather than "Stephanie" because Stephen is a more common search term. Some would say this is a gender bias and should be eliminated. As a woman in tech, I can certainly relate to and understand that perspective. Some would also say that the data is what it is, and if Stephen is a more common search term than Stephanie—then do we really want to make the algorithm do something other than what is best for user efficiency? It's a real quandary, and one can make legitimate arguments either way.

MICHAEL STONEBRAKER: The trouble with predictive models is that they are built by humans, and humans by nature are prone to bias. If we look at the most recent presidential election, we see a spectacular failure of existing polling models. Twenty-twenty hindsight shows that nobody thought

"The trouble with predictive models is that they are built by humans, and humans are by nature prone to bias."

Trump could actually win, when in reality, it is far more likely the polling models were subtly biased against him.

So, the problem with predictive models is the models themselves. If they include fraud, bias, etc., they can yield very bad answers. One has to take predictive models with a grain of salt. We put way too much faith in predictive modeling.

What role can big data and machine learning play in helping scientists understand data (for example, in the Human Genome project) and bring forth some potential real-world opportunities in health and medicine?

DAPHNE KOLLER: One of the main reasons I came back to the healthcare field is because I think the opportunity here is so tremendous. As costs go down, our ability to sequence new genomes increases dramatically. And it's not just genomes; it's transcriptomes and proteomes and many other data modalities. When we combine that with wearable devices that allow you to see the effect of phenotypes, there is an amazing explosion of data that we could access. One reason this is beneficial is that it will improve our ability to determine the genetic factors that cause certain diseases. Yes, we could do that before, but when faced with tens of millions of variations in the genome and only a couple hundred examples to use, it's really difficult to extract much out of that except the very strongest signals.

Are there potential technological breakthroughs on the horizon that could transform this area again in the near future?

DAVID BLEI: I think we are in the middle of a transformative time for machine learning and statistics, and

it's fueled by a few ideas. Reinforcement learning is a big one. This is the idea that we can learn how to act in the face of an uncertain environment with uncertain consequences of our actions; it's fueling a lot of the amazing results that we're seeing in machine learning and AI. Deep learning is another idea—a very flexible class of learners that, when given massive datasets, can identify complex and compositional structure in high-dimensional data. Another idea is 60 years old, but it's optimization: I have some kind of function and I want the maximal value of that function, how do I do that? Well, it's called an optimization procedure. Optimization tells us how to do that very efficiently with massive datasets.

VIPIN KUMAR: New types of sensors and communication technologies can be quite transformational. The kinds of sensors that we see today, we could not even have been imagined just a few decades ago. Mobile health sensors such as Fitbit and Apple Watches that can record our physiological parameters at unprecedented detail have been around only for the past decade or so. New types of sensors based on advances in electronics, nanotechnology, and biomedical sciences are already enabling deployment of small and inexpensive satellites that can monitor the earth and its environment at spatial and temporal resolutions never possible before. Without technologies such as RFID, it would be very hard for someone to imagine that you could walk into a store and purchase something just by looking at it or by being close to it—something that is now possible at Amazon Go, a grocery store in Seattle that has no checkout counter. New sensors based on quantum technology may open up entirely new applications that we are not even considering today.

Final thoughts?

MICHAEL STONEBRAKER: All of the fancy social benefits we expect from big data depends on seamless data integration. Solving the problem of how to improve data integration is going to be key in getting the most benefit from all the data being created. □

Inside Risks

Trustworthiness and Truthfulness Are Essential

Their absence can introduce huge risks ...

TRUSTWORTHINESS IS AN attribute that is fundamental to our technologies and to our human relations. Overly trusting something that is not trustworthy often leads to bad results. Not trusting something that really is trustworthy can also be harmful.

In many of the past 240 Inside Risks columns, we have been concerned extensively with trustworthiness, which should be a basic requirement of all computer-related systems—particularly when used in mission-critical applications, but also in personal settings such as maintaining your own quality of life. Trustworthiness is absolutely essential to the proper behavior of computers and networks, and to the well-being of entire nations and industries that rely on proper behavior of their computer-based enterprises.

Computer-Based Systems and People

Trustworthy system behavior typically may depend on trustworthiness of

people—for example, system designers, hardware developers and programmers, operational staff, and high-level managers. Many systems that might have some assessment of trustworthiness can nevertheless be seriously compromised by malicious malware, external adversaries, and insider misuse, or otherwise disrupted by denial-of-service attacks. If such compromises arise unexpectedly, then those systems were most likely not so trustworthy as had been believed.

Thus, we need system designs and implementations that are tolerant of people who might usually be trustworthy but who make occasional errors, as well as systems that are resistant to and resilient following many other potential adversities. More importantly, we need measures of assurance—which assess how trustworthy a system might actually be in certain circumstances (albeit typically evaluated only against perceived threats). Unfortunately, some the assumptions made prior to the evaluation process may have been

wrong, or may change over time—for example, as new types of threats are detected and exploited.

In addition to trustworthiness or untrustworthiness of people relevant to their interactions with computers in the above sense, trustworthiness and specifically personal integrity are also meaningful attributes of people and governments in their daily existence. In particular, truthfulness and honesty are typically thought of as trustworthiness attributes of people. The question of whether a particular computer system is honest would generally not be considered, because such a system has no moral compass to guide it. However, truthfulness is another matter. A system might actually be considered dishonest or even untruthful if it consistently or even intermittently gives wrong answers just in certain cases—especially if it had been programmed explicitly to do exactly that. For example, such behavior has been associated with certain proprietary voting systems—see Douglas W. Jones and Bar-



bara Simons, *Broken Ballots*, University of Chicago Press, 2012.

Systems can be untrustworthy because of false assumptions by the programmers and designers. For example, sensors measure whatever they are designed to measure, which may not include the variables that should be of greatest concern. Thus, a system assessing the slipperiness of the road for a vehicle might rely upon a sensor that determines whether the road is wet. Sometimes that is done by checking whether the windshield wipers are on—which is a rather indirect measure of slipperiness and can lead to false or imprecise recommendations or actions. At least one commercial aviation accident resulted from an indirect and imprecise determination of runway slipperiness.

Risks of Believing in Computer Trustworthiness

Many people believe computers are infallible and cannot lie. However, computers are created by people who are not infallible. Therefore, logically we might conclude that computers can-

not be infallible. Indeed, they cannot always perform exactly as expected, given the presence of hardware errors, power outages, malware, hacking attacks, and other adversities.

Indeed, computers can be made to lie, cheat, or steal. In such cases, of course, the faults may originate with or be amplified by people who commission systems, or design them, or program them, or even just use them, but not with the computers themselves. However, even supposedly ‘neutral’ learning algorithms and statistics can be biased and untrustworthy if they are presented with a biased or untrustworthy learning set. Unfortunately, the complexity of systems makes such behavior difficult to detect. Worse, many statistical learning algorithms (for example, deep learning) and artificial intelligence cannot specify how they actually reached their decisions, making it difficult to assess their validity.

► People who believe that online gambling is “fair” are likely to be easy victims. So can those who know it is not fair, but are nevertheless addicted (see Francis X. Clines, “Threatened

with Ruin at the Virtual Casino,” *The New York Times*, Feb. 5, 2017).

► People who believe that elections based on Internet voting and proprietary unauditable voting machines are inherently “fair” can be easily misled. People who continue to believe that Russians had no influence on the November 2016 election in the U.S. or in the April preliminary elections in France are oblivious to real evidence in both cases. Furthermore, The Netherlands recently abandoned electronic voting systems, returning to paper ballots—wary of further ongoing Russian interference.

Risks of Believing in Human Truthfulness and Integrity

Human creativity can have its downsides. For example, opportunities for ransomware, cyberfraud, cybercrime, and even spam all seem to be not only increasing, but becoming much more sophisticated.

Social engineering is still a simple and effective way to break into otherwise secure facilities or computer systems. It takes advantage of normal human decency, helpfulness, politeness,

**Reimaging
the Avatar Dream**

How Is IT Important?

**Inference Auction Design
in Online Advertising**

**On Cryptographic
Backdoors and
Prosthetic Intelligence**

**Side Effects,
Front and Center**

The IDAR Graph

**Research for Practice:
Tracing and Debugging
Distributed Systems;
Programming
by Examples**

**When Does Law
Enforcement's Demand
to Read Your Data
Become a Demand
to Read Your Mind?**

**IronFleet:
Proving Safety and
Liveness of Practical
Distributed Systems**

**Fast and Powerful
Hashing Using
Tabulation**

Plus the latest news about
neuromorphic computing,
DNA data storage, and AI—
Is it more artificial
than intelligent?

and altruism. A knee-jerk attempt to rein in social engineering could involve eliminating these very desirable social attributes (which might also eliminate civility and decency from our society).

How Does This All Fit Together?

It should be fundamental to readers of Inside Risks articles that point solutions to local problems are generally insufficient, and that we have to consider trustworthiness in the total-system context that includes hardware, software, networking, people, environmental concerns, and more. On September 22, 1988, Bob Morris (then chief scientist of the National Computer Security Center at NSA) said in a session of the National Academies' Computer Science and Telecommunications [now Technology] Board on problems relating to security, "To a first approximation, every computer in the world is connected with every other computer." That quote is even more relevant today, almost 30 years later. Similarly, all of the issues considered in this column involving computers and people may be intimately intertwined.


Science is never perfect or immutable—it is often a work in progress. Hence, scientists can rarely if ever know they have the absolute final answer. However, scientific methods have evolved over time, and scientists generally welcome challenges and disagreements that can ultimately be resolved through better theories, experimental evidence, and rational debate. Occasionally, we even find fake science and untrustworthy scientists, although these aberrations tend to be refuted eventually via peer pressure. Where science has strong credible evidence, it deserves to be respected—because in the final analysis reality should be able to trump fantasies (although this in fact may not work).

Truth is perhaps even more in flux than science, and certainly relative, not absolute—with many caveats. However, truth matters. We might paraphrase the oft-cited Albert Einstein quote as "Everything should be stated as simply as possible, but not simpler." Oversimplifications, the lack of foresight, and a seriously non-objective perspective are often sources of serious misunderstandings, and can result in major catastrophes. On the other hand, untruthfulness must not

be confused with truth, even though that confusion appears remarkably common. People who believe everything they read on Facebook, Google, Amazon, Twitter, and other Internet sites are clearly delusional.

Conclusion

People who are less aware of technology-related risks tend to overendow computers as perfect, while computers have little respect for people. Neither computer behavior nor human behavior is always perfect, and should not be expected to be so. There are significant risks in blindly believing in computer trustworthiness and human truthfulness. We must not believe in computer infallibility, or in everything we read on the Internet in the absence of credible corroboration. But then we should also not believe people who pervasively dishonor truthfulness.

Unfortunately, the trends for the future seem relatively bleak. Computer system trustworthiness and the implications of its absence are increasingly being questioned. For example, a recent article by Bruce G. Blair (Hacking our Nuclear Weapons, *The New York Times*, Mar. 14, 2017) suggests "Loose security invites a cyberattack with possibly horrific consequences." Semi- and fully autonomous systems, the seemingly imminent Internet of Things, and artificial intelligence are providing further examples in which increasing complexity leads to obscure and unexplainable system behavior. The concept of trustworthiness seems to be becoming supplanted with people falsely placing their trust in systems and people that are simply not trustworthy—without any strong cases being made for safety, security, or indeed assurance that might otherwise be found in regulated critical industries such as aviation. However, the risks of false would-be "facts" may be the ultimate danger. An obvious consequence might be the extensive institutional loss of trust in what is neither trustworthy nor truthful. The truth and trustworthiness may be even more important now than ever before. 

Peter G. Neumann (neumann@csl.sri.com) moderates the ACM Risks Forum and is Senior Principal Scientist in SRI International's Computer Science Lab. He is grateful to Donald Norman for considerable useful feedback.

Copyright held by author.

Broadening Participation

The Influence and Promise of Alliances

Evaluating the influence of broadening participation efforts on students, faculty, organizations, and the computing education infrastructure.

IN EARLY 2016, the White House announced a government-wide investment in computing education—Computer Science for All—to be included in the President’s 2017 budget. Computer Science for All would give every P–12 student the chance to learn computer science and to be given the opportunities “that allow them to join the innovation economy, have the tools to solve our toughest challenges, and become active citizens in our increasingly technological world.”² A major component of the initiative is inclusion in computing by students from underrepresented groups, such as African Americans, Hispanics, Native Americans, and people with disabilities. The White House announcement stated that CS for All builds on computing education momentum at state and local levels.

Although the sources of this momentum were not named, we speculate that an important impetus for innovation and growth in computer science P–12 education comes from programs funded at the federal level, such as the NSF-funded CS 10K initiative, and community efforts, such as code.org. Predating these P–12 education efforts, the National Science Foundation launched the Broadening Participation in Computing (BPC) program in 2006 as a model to effectively address the issues of un-

derrepresentation, as well as respond to the need to increase participation in computing education and produce computing professionals. Chubin and Johnson¹ provided a description of the 11 alliances that constituted the core of BPC as of 2009. In total, 15 BPC Alliances (BPC-A) have been funded. These Alliances represent broad coalitions of academic postsecondary institutions, secondary and middle schools, government, industry, professional societies, and other not-for-profit organizations.

The goal of BPC-Alliances is to design and carry out comprehensive programs to reduce underrepresentation in the computing disciplines at various stages of the academic pathway from K–12 through the early faculty ranks. To this end, the BPC-Alliances have endeavored to create the best practices, educational resources, advocacy networks, and forums needed

Alliances use diverse strategies to expose students to computing concepts and careers.

to address issues of engagement and education. While specifically charged with addressing the long-standing underrepresentation of many groups within the computing community, many, if not most, of the Alliances’ activities also worked to increase awareness, access, engagement, and inclusion for all students. The Alliances’ strategies encompass not only activity design and implementation but also promotion of computing education through workshops and information dissemination.

Considered together as a whole, the BPC Alliances are expected to have significant impact on both the quality of opportunities afforded to participants and the number of participants. To assess this collective impact, in 2012, EDC, Inc., was awarded a contract to evaluate the accomplishments and impacts of the BPC-A program as a whole. They partnered with Westat, Inc., and the College of Education at Kansas State University in the multiple-year evaluation to investigate the BPC-A program’s creation of a national system of resources to support broadening participation in computing, focusing document review on the first five years (2006–2011) and empirical data collection on the second five years of program funding (2011–2016).

As of 2011, there were eight BPC Alliances:

- ▶ Access Computing (AC), University of Washington; <http://www.washington.edu/accesscomputing/>
- ▶ Computing Alliance of Hispanic-Serving Institutions (CAHSI), University of Texas at El Paso; <http://cahsi.cs.utep.edu/>
- ▶ Sustainable Diversity in the Computing Research Pipeline: Computing Research Association-Women/ The Coalition to Diversity Computing (CRA-W/CDC), the Computing Research Association in Washington D.C.; <http://cra.org/cra-w/>
- ▶ Expanding Computing Education Pathways (ECEP), University of Massa-

chusetts at Amherst; <http://expanding-computing.cs.umass.edu/>

- ▶ Institute for African-American Mentoring in Computing Sciences (IAAMCS), Clemson University; <http://www.iaamcs.org/>

- ▶ Into the Loop (ITL), University of California Los Angeles; <http://idea.gseis.ucla.edu/projects/into-the-loop>

- ▶ National Center for Women and Information Technology (NCWIT), University of Colorado at Boulder; <https://www.ncwit.org/>

- ▶ STARS Computing Corps (STARS), University of North Carolina at Charlotte; <http://starscomputingcorps.org/>

The programs and resources offered by these Alliances have leveraged public and private partnerships to target computing education in K–20 grade levels, preparing diverse students for careers in computing and computer science. These activities built a foundational wealth of knowledge and practices for the field of computing education research and diversity education research. The foci for these Alliances are seen in the figure here.

Documenting and assessing the associations, relationships, outcomes, and influence among Alliances and with their partners was a significant undertaking because of the reach and depth of the Alliances’ partnerships. In addition, the Alliances target different but overlapping audiences; implement a variety of strategies and activities; develop multiple types of resources and

products; and partner with diverse organizations to reach their outcomes and objectives. In consideration of these challenges, the evaluation team utilized a multiple-method case study approach to answer a variety of evaluation questions related to the Alliances’ influence on students, faculty, organizations and the infrastructure of computing education and efforts to broaden participation. This column describes a small part of the findings of the evaluation.

The evaluation team described the program accomplishments in two stages. The first five years of the Alliances were spent developing and testing models, such as courses, mentoring, service learning, workshops, summer programs, fellowships, webinars, and competitions. The second five years of the program focused more on leveraging knowledge and serving as national resources. Examples of activities during these years are influencing policies, serving as national resources, fostering research, facilitating networks of professionals and supporting a national infrastructure.

A cross-case analysis showed several findings related to the accomplishments and reach of the Alliances.

- ▶ Alliance partnerships and activities reached every state in the U.S.


- ▶ Alliances use diverse strategies to expose students to computing concepts and careers, influencing their interest and confidence in pursuing computing careers.

- ▶ Through capacity-building workshops, conferences, and professional development, Alliances inspire faculty, teachers, and other professionals to develop collaborations and change the climate in computer science education.

- ▶ Organizations across the computing education spectrum benefit from Alliances’ resources, finding new approaches to teaching and learning, recruiting and retaining students, and developing public/private partnerships.

- ▶ Alliances are well positioned to serve as national resources, disseminating promising practices, scaling tested models, and supporting national and regional efforts to broaden participation. In this capacity, they disseminate new knowledge and research about broadening participation in computing through publications, presentations, conferences, and Web-based dissemination strategies.

- ▶ Evaluation at both the Alliance and the program level is critical for documenting Alliances’ influence and outcomes. However, more and stronger data are needed to document the influence of the Alliances (individually and collectively) on broadening participation in computing.

The evaluation team discovered that BPC Alliances employ many different approaches, influencing people, organizations, infrastructure and ultimately the landscape of the field, demonstrating that there is no “one right way” to broaden participation in computing. Together, the Alliances for Broadening Participation in Computing are building a more equitable ecosystem for computer science education in the U.S. 

Foci of Eight Broadening Participation in Computing Alliances

The Alliances

AccessComputing

Supporting students with disabilities to pursue degrees in computing fields

Computing Alliance of Hispanic-Serving Institutions (CAHSI)

Leveraging member institutions to increase the number of Hispanic students who complete degrees in computing

Sustainable Diversity in the Computing Research Pipeline Increasing the participation of women and underrepresented minorities in computing research careers

Expanding Computing Education Pathways (ECEP)

Increasing the number and diversity of students completing computing degrees by supporting state-level computing education policy change

Into the Loop

Enhancing high school students’ computer science learning through implementation and dissemination of equity-focused curricula and professional development

Institute for African-American Mentoring in Computer Science

Addressing the shortage of African Americans pursuing Ph.D.’s and research careers in computing-related fields

National Center for Women & Information Technology (NCWIT)

Bringing together universities, non-profits, and for-profit organizations to advance women’s and girls’ participation in computer science

Students in Technology, Academia, Research, and Service Alliance (STARS)

Fostering a community of practice for students through service learning and building computing education capacity in member institutions

References

1. Chubin, D.E. and Johnson, R.Y. A program greater than the sum of its parts: The BPC Alliances. *Commun. ACM* 54, 3 (Mar. 2011), 35–37; DOI: 10.1145/1897852.1897866
2. White House Office of Science and Technology Policy (Feb. 2016); www.whitehouse.gov/ostp

Leslie Goodyear (lgoodyear@edc.org) is Principal Research Scientist at Education Development Center (EDC, Inc.). Her work includes multiple evaluations of NSF programs and projects focused on broadening participation in STEM.

Gary Silverstein (SILVERGI@westat.com) is an Associate Director at Westat, Inc. His work focuses on project monitoring and program evaluation of efforts to enhance educational opportunities at the K–20 levels.

Linda P. Thurston (lpt@k-state.edu) is professor of Special Education, Counseling, and Student Affairs in the College of Education at Kansas State University. She serves as Associate Dean for Research and Graduate Studies, is the Lydia E. Skeen Chair in Education, and is a co-principal investigator of the Kansas LSAMP program.

Copyright held by authors.



Kode Vicious

Forced Exception Handling

You can never discount the human element in programming.

Dear KV,

I subscribe to “The Morning Paper,” a daily summary prepared by one person, Adrian Colyer, who curates research papers and sends them out to interested readers (<https://blog.acolyer.org>). Last fall he reviewed “Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems” (<https://blog.acolyer.org/2016/10/06/simple-testing-can-prevent-most-critical-failures/>). It had some surprising results, including:

- ▶ Almost all catastrophic failures (48 in total, or 92%) are the result of incorrect handling of nonfatal errors explicitly signaled in software;

- ▶ Error handlers with TODO or FIX-ME in the comments. This example took down a 4,000-node production cluster; and

- ▶ Error handlers that catch an abstract exception type (for example, Exception or Throwable in Java) and then take drastic action such as aborting the system. This example brought down a whole Hadoop Distributed File System (HDFS) cluster.

And the list went on from there.

I have been reading your Kode Vicious columns for quite a while, and as I read the review and then the paper itself, it looked like something you would be interested in, so I have sent along the link.

Helpfully Not in Error

Dear Helpfully,

Yes, KV also reads “The Morning Paper,” although he has to admit that he



does not read everything that arrives in his inbox from that list. Of course, the paper you mention piqued my interest, and one of the things you did not point out is that it is actually a study of distributed systems failures. Now, how can we make programming harder? I know! Let’s take a problem on a single system and distribute it. Someday I would like to see a paper that tells us if problems in distributed systems increase along with the number of nodes, or the number of interconnections. Being an optimist, I can only imagine that it is $N(N + 1) / 2$, or worse.

I don’t think you pointed out this paper to KV just to hear me bang my head on my desk while thinking distributed systems, so let’s assume you’re asking the “Why?” question: “Why is it the case that 92% of the catastrophic failures in this paper are caused by a fail-

ure to handle nonfatal errors?”

Well, let’s see what else the paper had to say and then think about how software is actually implemented in the real world, rather than how we believe it ought to be implemented in the illusory world that management and marketing inhabit.

To get to the heart of why nonfatal errors might have led to fatal errors, we need look no further than this snippet from the paper: “This difference is likely because the Java compiler forces developers to catch all the checked exceptions; and a variety of errors are expected to occur in large distributed systems, and the developers program more defensively. However, we found they were often simply sloppy in handling these errors” (<https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-yuan.pdf>).

Hopefully anyone who has been a

COMMUNICATIONS APPS

Access the latest issue, past issues, BLOG@CACM, News, and more.



Available for iPad, iPhone, and Android



Available for iOS, Android, and Windows

<http://cacm.acm.org/about-communications/mobile-apps>



Association for Computing Machinery

professional programmer for more than a few days knows that many developers will always write the code they are most interested in, or pressured to deliver first, which is not the error- and exception-handling code, nor is it test code, nor documentation, the latter two of which I have already harangued readers about, ad nauseam. What management and the rest of the team want, is “the code,” and what most people see as “the code” is only the part of it that explicitly does the job you are expected to do. It’s not even the demands of others that cause this narrow focus; it’s often just that the error-handling parts are not as interesting to the person writing the code as getting a result. It would seem that many programmers just want to move those bits, munge that data, and show pictures of cats.

In point of fact we have a clear indication of the importance programmers put on the error-handling components of the code by this finding: “Error handlers with TODO or FIXME in the comment.” Personally, I prefer XXX, as it reminds me of my time in Amsterdam in the early 1990s, and unless you’re working in certain industries—industries that might also serve photos, and might still serve photos of cats—you’re unlikely to find XXX as a variable in the code.

We can look at the fact that the Java compiler forces programmers to catch all the unchecked exceptions in one of two ways. If we are charitable—and KV is the heart and soul of charity—we assume the Java language and compiler developers are simply helping programmers make fewer mistakes and make sure their code not only does what it is meant to do, but also acts appropriately when things go awry.

If we are less charitable, or perhaps more honest and realistic, we see this enforcement quite differently: as a naked attempt to control programmers and make them do what the language and compiler people thought was right at the time. “Programmers don’t do proper error handling. I know, we will MAKE them handle errors, or their programs won’t compile at all!” I believe this is said in the voice of an overbearing schoolteacher. “You *will* dot your i’s! You *will* catch all exceptions!” Except that unlike dotting an i, there are ways to skate around handling the exception that was meant to be

handled. In a rush? Well then, just add a TODO or FIXME or XXX in the comments and move on. You’ll come back to it later ... of course you will.

Both sides are a little bit wrong in this case. We can all point fingers at the person who leaves a trail of FIXMEs in the code, but who among us is without blame in that regard? We can also blame the pedants who thought that forcing every exception to be caught was doing us a favor. You can never discount the human element in programming. For everything you try to force on someone, there is something they will work to avoid if at all possible. Tool builders need to understand that the people who use their tools are often trying to get a very narrow job completed with a minimum amount of effort. Was it wrong to add the forced exception handling into the tool? Maybe and maybe not. In the hands of someone with the time and inclination to do the right thing, these errors are a welcome way of finding problems that they do have to handle.

Clearly, in the hands of a large percentage of programmers who work on some of the most complex systems yet devised, the feature is actually a nuisance, and it is likely time to rethink how this particular exception ought to be handled.

KV

Related articles on queue.acm.org

Productivity in Parallel Programming: A Decade of Progress

John T. Richards, et al.

Looking at the design and benefits of X10
<http://queue.acm.org/detail.cfm?id=2682913>

Going with the Flow

Peter de Jong

Workflow systems can provide value beyond automating business processes.
<http://queue.acm.org/detail.cfm?id=1122686>

The Challenge of Cross-language Interoperability

David Chisnall

Interfacing between languages is increasingly important
<http://queue.acm.org/detail.cfm?id=2543971>

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

Copyright held by author.

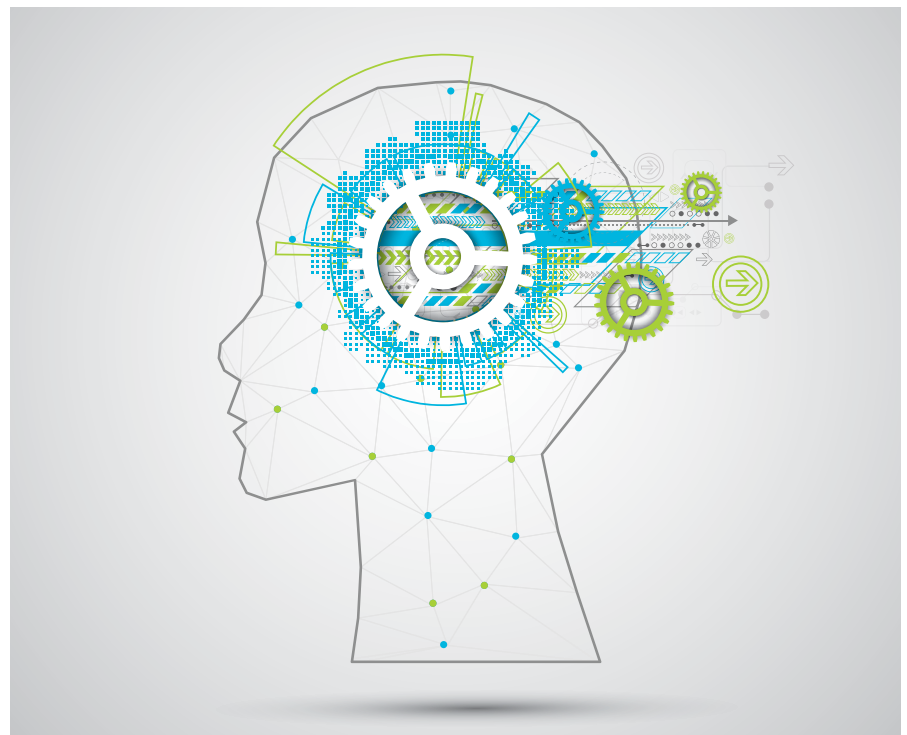
Viewpoint

Remaining Trouble Spots with Computational Thinking

Addressing unresolved questions concerning computational thinking.

COMPUTATIONAL THINKING HAS been a hallmark of computer science since the 1950s. So also was the notion that people in many fields could benefit from computing knowledge. Around 2006 the promoters of the CS-for-all K-12 education movement claimed all people could benefit from thinking like computer scientists. Unfortunately, in attempts to appeal to other fields besides CS, they offered vague and confusing definitions of computational thinking. As a result today's teachers and education researchers struggle with three main questions: What is computational thinking? How can it be assessed? Is it good for everyone? There is no need for vagueness: the meaning of computational thinking, evolved since the 1950s, is clear and supports measurement of student progress. The claims that it benefits everyone beyond computational designers are as yet unsubstantiated. This examination of computational thinking sharpens our definition of algorithm itself: an algorithm is not any sequence of steps, but a series of steps that control some abstract machine or computational model without requiring human judgment. Computational thinking includes designing the model, not just the steps to control it.

Computational thinking is loosely defined as the habits of mind developed



from designing programs, software packages, and computations performed by machines. The Computer Science for All education movement, which began around 2006, is motivated by two premises: that computational thinking will better prepare every child for living in an increasingly digitalized world, and that computational thinkers will be superior problem solvers in all fields.

Since 2006 hundreds of educators have participated in workshops, stud-

ies, committees, surveys, new courses, and public evaluations to define computational thinking for “CS for all” curricula. The Computer Science Teachers Association issued an operational definition in 2011 (see Box 1), the Computing at School subdivision of the British Computer Society followed in 2015 with a more detailed definition (see Box 2), and the International Society for Technology in Education followed in 2016 with a generalized technology

Box 1.

Computer Science Teachers Association's Concepts of Computational Thinking:⁴

Formulating problems for computational solution

Logically organizing and analyzing data

Abstractions including models and simulations

Algorithmic thinking

Evaluation for efficiency and correctness

Generalizing and transferring to other domains

Supported by: dispositions of confidence in dealing with complexity, persistence with difficult problems, tolerance for ambiguity, open-ended problems, communication and collaboration

Box 2.

Computing at School's Concepts of Computational Thinking:³

Logical reasoning

Algorithmic thinking

Decomposition

Generalization

Patterns

Abstraction

Representation

Evaluation

Supported by: techniques of reflecting, coding, designing, analyzing, and applying

Box 3.

ISTE's Standards for Students in Computational Thinking:¹⁵

Leverage the power of technological methods to develop and test solutions

Collect data

Analyze data

Represent data

Decomposition

Abstraction

Algorithms

Automation

Testing

Parallelization

Simulation

Supported by: empowered learner, digital citizen, knowledge constructor, designer, communicator, collaborator

definition (see Box 3). There are other frameworks as well.^{21,27}

Given all this work, I was surprised recently when some teachers and education researchers asked for my help answering three questions with which they continue to struggle:

1. What is computational thinking?
2. How do we measure students' computational abilities?
3. Is computational thinking good for everyone?

To support my answers, I reviewed many published articles. I learned that these three questions are of concern to teachers in many countries and that educators internationally continue to search for answers.²¹

It concerns me that teachers at the front lines of delivering computing education are still unsettled about these basic issues. How can they be effective if not sure about what they are teaching and how to assess it? In 2011, Elizabeth Jones, then a student at the University of South Carolina, warned that lack of answers to these questions could become a problem.¹⁶

I believe the root of the problem is that, in an effort to be inclusive of all

fields that might use computing, the recent definitions of computational thinking made fuzzy and overreaching claims. Is it really true that any sequence of steps is an algorithm? That procedures of daily life are algorithms? That people who use computational tools will need to be computational thinkers? That people who learn computational thinking will be better problem solvers in all fields? That computational thinking is superior to other modes of thought?

My critique is aimed not at the many accomplishments of the movements to get computer science into all schools, but at the vague definitions and unsubstantiated claims promoted by enthusiasts. Unsubstantiated claims undermine the effort by overselling computer science, raising expectations that cannot be met, and leaving teachers in the awkward position of not knowing exactly what they are supposed to teach or how to assess whether they are successful.

My purpose here is to examine these questions and in the process elucidate what computational thinking really is and who is it good for.

Question 1: What Is Computational Thinking?

A good place to look for an answer is in our history. Computational thinking has a rich pedigree from the beginning of the computing field in the 1940s. As early as 1945, George Polya wrote about mental disciplines and methods that enabled the solution of mathematics problems.²⁹ His book *How to Solve It* was a precursor to computational thinking.

In 1960, Alan Perlis claimed the concept of "algorithmizing" was already part of our culture.¹⁸ He argued that computers would automate and eventually transform processes in all fields and thus algorithmizing would eventually appear in all fields.

In the mid-1960s, at the time I was entering the field, the pioneers Allen Newell, Alan Perlis, and Herb Simon were defending the new field from critics who claimed there could be no computer science because computers are man-made artifacts and science is about natural phenomena.²³ The three pioneers argued that sciences form around phenomena that people want to harness; computers as information

transformers were a new focal phenomenon covered by no other field. They also argued that “algorithmic thinking”—a process of designing a series of machine instructions to drive a computational solution to a problem—distinguishes computer science from other fields.

In 1974, Donald Knuth said that expressing an algorithm is a form of teaching (to a dumb machine) that leads to a deep understanding of a problem; learning an algorithmic approach aids in understanding concepts of all kinds in many fields.

In 1979, Edsger Dijkstra wrote about the computational habits of mind he learned to help him program well:⁹ separation of concerns; effective use of abstraction; design and use of notations tailored to one’s manipulative needs; and avoiding combinatorially exploding case analyses.

Seymour Papert may have been the first to use the term computational thinking in 1980, when in his book *Mindstorms* he described a mental skill children develop from practicing programming.^{24,25}

In 1982, Ken Wilson received a Nobel prize in physics for developing computational models that produced startling new discoveries about phase changes in materials. He went on a campaign to win recognition and respect for computational science. He argued that all scientific disciplines had very tough problems—“grand challenges”—that would yield to massive computation.³³ He and other visionaries used the term “computational science” for the emerging branches of science that used computation as their primary method. They saw computation as a new paradigm of science, complementing the traditional paradigms of theory and experiment. Some of them used the term “computational thinking” for the thought processes in doing computational science—designing, testing, and using computational models to make discoveries and advance science. They launched a political movement to secure funding for computational science research, culminating in the High Performance Communication and Computing (HPCC) Act passed in 1991 by the U.S. Congress. Computer scientists were slow to join the movement,

The search for computational models pervades all of computational science.

which grew up independently of them. Easton noted that, as computational science matured, computational thinking successfully infiltrated the sciences and most sciences now study information processes in their domains.¹²

The current surge of interest in computational thinking began in 2006 under the leadership of Jeanette Wing.^{35–37} While an NSF assistant director for CISE, she catalyzed a discussion around computational thinking and mobilized resources to bring it into K–12 schools. Although I supported the goal of bringing computer science to more schools, I took issue with the claim of some enthusiasts that computational thinking was a new way to define computing.⁷ The formulations of computational thinking at the time emphasized extensions of object-oriented thinking to software development and simulation—a narrow view the field. Moreover, the term had been so widely used in science and mathematics that it no longer described something unique to the computing field.

In 2011, on the eve of Alan Turing’s 100th birthday, Al Aho wrote a significant essay on the meaning of computational thinking² for a symposium on computation in ACM Ubiquity.⁵ He said: “Abstractions called computational models are at the heart of computation and computational thinking. Computation is a process that is defined in terms of an underlying model of computation and computational thinking is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms.”^{2, a}

a Aho’s definition was noted by Wing in 2010³⁵ and is used as the definition of computational thinking by K12cs.org.

Aho emphasized at great length the importance of computational models. When we design an algorithm we are designing a way to control any machine that implements the model, in order that the machine produces a desired effect in the world. Early examples of models for computational machines were Turing machines, neural networks, and logic reduction machines, and, recently, deep learning neural networks for artificial intelligence and data analytics. However, computational models are found in all fields. The Wilson renormalization model is an example in physics, the Born-Oppenheimer approximation is an example in chemistry, and the CRISPR model is an example from biology. Aho says further, “[With new problems], we discover that we do not always have the appropriate models to devise solutions. In these cases, computational thinking becomes a research activity that includes inventing appropriate new models of computation.”²

As an example, Aho points out that in computational biology there is a big effort to find computational models for the behavior of cells and their DNA. The search for computational models pervades all of computational science. Aho’s insight that computational thinking relies on computational models is very important and has been missed by many proponents.

Aho’s term computational model is not insular to computer science—it refers to any model in any field that represents or simulates computation. I noted several examples above. Moreover, his definition captures the spirit of computational thinking expressed over 60 years of computer science and 30 years of computational science.^b It also captures the spirit of computational thinking in other fields such as humanities, law, and medicine.

This short survey of history reveals two major sources of ambiguity in the post-2006 definitions of computational thinking. One is the absence of any mention of computational models. This is a mistake: we engage with abstraction, decomposition, data repre-

b I was an active researcher in the computational sciences field during the 1980s and 1990s and can attest that his definition captures what the computational scientists of the day said they were doing.

INTERACTIONS



ACM's *Interactions* magazine explores critical relationships between people and technology, showcasing emerging innovations and industry leaders from around the world across important applications of design thinking and the broadening field of interaction design.

Our readers represent a growing community of practice that is of increasing and vital global importance.



To learn more about us, visit our award-winning website <http://interactions.acm.org>

Follow us on Facebook and Twitter



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

Association for Computing Machinery



sentation, and so forth, in order to get a model to accomplish certain work.

The other is the suggestion contained in the operational definitions that any sequence of steps constitutes an algorithm. True, an algorithm is a series of steps—but the steps are not arbitrary, they must control some computational model. A step that requires human judgment has never been considered to be an algorithmic step. Let us correct our computational thinking guidelines to accurately reflect the definition of an algorithm. Otherwise, we will mis-educate our children on this most basic idea.

Question 2: How Do We Measure Students' Computational Abilities?

Most teachers and education researchers have the intuition that computational thinking is a skill rather than a particular set of applicable knowledge. The British Computer Society CAS description quoted earlier seems to recognize this when discussing what “behaviors” signal when a student is thinking computationally.³ But we have no consensus on what constitutes the skill and our current assessment methods are unreliable indicators.

A skill is an ability acquired over time with practice—not knowledge of facts or information. Most recommended approaches to assessing computational thinking assume that the body of knowledge—as outlined in Boxes 1–3—is the key driver of the skill's development. Consequently, we test students' knowledge, but not their competence or their sensibilities. Thus it is possible that a student who scores well on tests to explain and illustrate abstraction and decomposition can still be an incompetent or insensitive algorithm designer. Teachers sense this and wonder what they can do. The answer is, in a nutshell, to directly test for competencies.^c

The realization that mastering a domain's body of knowledge need not confer skill at performing well in the

domain is not new. As early as 1958, philosopher Michael Polanyi discussed the difference between “explicit knowledge” (descriptions written down) and “tacit knowledge” (skillful actions).²⁸ He famously said: “We know more than we can say.” He gave many examples of skilled performers being unable to say how they do what they do, and of aspirants being unable to learn a skill simply by being told about it or reading a description. Familiar examples of tacit knowledge are riding a bike, recognizing a face, or diagnosing an illness. Many mental skills fall into this category too, such programming or learning a foreign language. Every skill is a manifestation of tacit knowledge. People learn a skill only by engaging with it and practicing it.

To certify skills you need a model for skill development. One of the most famous and useful models is the framework created by Stuart and Hubert Dreyfus in the 1970s.¹⁰ They said that practitioners in any domain progress through six stages: beginner, advanced beginner, competent, proficient, expert, and master. A person's progress takes time, practice, and experience. The person moves from rule-based behaviors as a beginner to fully embodied, intuitive, and game-changing behaviors as a master. Hubert Dreyfus gives complete descriptions of these levels in his book on the Internet.¹¹ We need guidelines for different skill levels of computational thinking to support competency tests.

The CAS and K12CS organizations have developed frameworks for defining computational thinking that feature progressions of increasingly sophisticated learning objectives in various tracks including algorithms, programming, data, hardware, communication, and technology.^d These knowledge progressions are not the same as skill acquisition progression in the Dreyfus model. The CAS framework does not discuss abilities to be acquired during the progression. The K12CS framework gets closer by proposing seven practices^e—only three of which are directly

c In 1992, Ted Sizer of Brown University started a national movement for competency-based assessment in schools.³¹ He used the term “exhibitions” for assessment events. I gave examples for engineering schools.⁸ According to the Christensen Institute, competency-based learning is a growing movement in schools.³⁴ In 2016, Purdue became the first public university to fully embrace competency-based learning in an academic program in its Polytechnic Institute.

d CAS: <https://community.computingatschool.org.uk/resources/2324>; K12CS: <https://k12cs.org>

e Fostering an inclusive and diverse computing culture, collaborating, recognizing and defining computational problems, developing and using abstractions, creating computational artifacts, testing and refining, communicating.

related to competence at designing computations. Their notion of practice is “way of doing things” rather than an ability accompanied by sensibilities. Teachers who use these frameworks are likely to find that the associated assessment methods do not test for the abilities they are after.

Employers are turning to competency-based assessment faster than educational institutions. Many employers no longer trust transcripts and diplomas. Instead they organize interviews as rigorous problem-solving sessions with different groups in the company. An applicant will not be hired without demonstrating competence in solving the kinds of problems of interest to the employer. The idea of assessing skill by performance is actually quite common in education. In sports, music, theater, and language departments, for example, students audition for spots on the team, places in the orchestra, roles in the play, or competency certificates at a language. Although code-a-thons are becoming more prevalent and many computing courses include projects that assess skill by performance, computing education would benefit from a deep look at competency-based assessment.

Given that so much education is formulated around students acquiring knowledge, looking carefully at skill development in computational thinking is a new and challenging idea. We will benefit our students by learning to approach and assess computational thinking as a skill.

Question 3: Is Computational Thinking Good for Everyone?

The third question addresses a bundle of claims about benefits of computational thinking. Let us unpack them and see which claims are substantiated and which are not.

Wing’s vision for the computational thinking movement was that “everyone, not just those who major in computer science, can benefit from thinking like a computer scientist”^{36,37} At a high level it is hard to question this claim—more tools in the mental toolbox seems like a worthy goal. However, on a closer look not everyone benefits and some claims do not seem to benefit anyone. Consider the following.

There is little doubt that people who design and produce computa-

Traditional versus New Computational Thinking

The companion article traces the history of computational thinking from its origins in the 1950s until the present time. It is a story of the mental habits and disciplines for designing useful and reliable programs. It began with Alan Perlis in the 1950s, was well characterized for CS by Donald Knuth and Edsger Dijkstra in the 1970s, and expanded as the third way of science in the computational science movement of the 1980s. We call this Traditional CT.

After 2006 a new version emerged, seeded by an article by Jeannette Wing and then propelled when the U.S. National Science Foundation put a lot of resources into using CT as a conceptual lever to get computing into all K–12 schools. This massive effort defined its own version of CT independent of the past history. It is a story of how problems might be solved by expressing their solutions as computational steps. We call this New CT.

The Traditional CT and the New CT are not the same. One of the important differences is that in Traditional CT programming ability produces CT, and in New CT learning certain concepts produces programming ability. The direction of causality is reversed. The table here may help readers understand the origins of the trouble spots discussed in this Viewpoint.

| Traditional CT | New CT |
|--|---|
| Mental habits and disciplines for designing useful software | Formulating problems so that their solutions can be expressed as computational steps |
| Extensively practicing programming cultivates CT as a skill set | CT is a conceptual framework that enables programming |
| Skills of design and software crafting—for example separation of concerns, effective use of abstraction, devising notations tailored to one’s needs, and avoiding combinatorically exploding case analyses | Set of problem solving concepts such as representation, divide-and-conquer, abstraction, information hiding, verification, and logical reasoning |
| A new way of conducting science, alongside theory and experiment—a revolution in science | Useful in sciences and most other fields |
| Algorithms are directions to control a computational model (abstract machine) to perform a task | Algorithms are expressions of recipes for carrying out tasks; no awareness of computational models is needed |
| Programs are tightly coupled with algorithms; programs are algorithms expressed in a computer language; algorithms derive their precision from a computational model | Programs are loosely coupled with algorithms; algorithms are for all kinds of information processors including humans—it is completely optional whether an algorithm will ever be translated into a program |
| Designing computations in a domain requires extensive domain knowledge | Someone schooled in the principles of CT can find computational solutions to problems in any domain |
| End users can follow algorithms and get the result without any understanding of the mechanism | People engaging in any step-by-step procedure are performing algorithms and are (perhaps unconsciously) thinking computationally |
| Engaging in a computational task without awareness is not computational thinking | People who are engaging in any task that could be performed computationally are engaging in subconscious computational thinking |

tional models and software in many fields—let’s call them computational designers—develop strong skills of computational thinking. Experienced computational designers believe they are sharper and more precise in their thinking and are better problem solvers.

Recognizing this early on, Alan Perlis was one of the first to generalize (1960); he claimed that everyone can benefit from learning computational

thinking. Other luminaries have followed suit.^{9,18,19,24,33,35} However, this general claim has never been substantiated with empirical research.

For example, it is reasonable to question whether computational thinking is of immediate use for professionals who do not design computations—for example, physicians, surgeons, psychologists, architects, artists, lawyers, ethicists, realtors, and more. Some of

these professionals may become computational designers when they modify tools, for example by adding scripts to document searchers—but not everybody. It would be useful to see some studies of how essential computational thinking is in those professions.

Another claim suggested in the operational definitions is that users of computational tools will develop computational thinking. An architect who uses a CAD (computer aided design) tool to draw blueprints of a new building and a VR (virtual reality) tool to allow users to take simulated tours in the new building can set up the CAD and VR tools without engaging in computational thinking. The architect is judged not for skill in computational thinking but for design, esthetics, reliability, safety, and usability.^f Similar conclusions hold for doctors using diagnostic systems, artists drawing programs, lawyers document searchers, police virtual reality trainers, and realtors house-price maps. Have you noticed that our youthful “digital natives” are all expert users of mobile devices, apps, online commerce, and social media but yet are not computational thinkers? As far as I can tell, few people accept this claim. It would be well to amend the operational definitions to remove the suggestion.

Another claim suggested in the operational definitions is that computational thinking will help people perform everyday procedural tasks better—for example, packing a knapsack, caching needed items close by, or sorting a list of customers. There is no evidence to support this claim. Being a skilled performer of actions that could be computational does not necessarily make you a computational thinker and vice versa.^{13,14} This claim is related to the idea I criticized earlier, that any sequence of steps is an algorithm.

The boldest claim of all is that computational thinking enhances general cognitive skills that will transfer to other domains where they will manifest as superior problem-solving skills.^{3,37} Many education researchers have searched for supporting evi-

dence but have not found any. One of the most notable studies, by Pea and Kurland in 1984, found little evidence that learning programming in Logo helped students’ math or general cognitive abilities. In 1997, Koschmann weighed in with more of the same doubts and debunked a new claim that learning programming is good for children just as learning Latin once was.²⁰ (There was never any evidence that learning Latin helped children improve life skills.) Mark Guzdial reviewed all the evidence available by 2015 and reaffirmed there is no evidence to support the claim.¹⁴

Guzdial does note that teachers can design education programs that help students in other domains learn a small core of programming that will teach enough computational thinking to help them design tools in their own domains. They do not need to learn the competencies of software developers to be useful.

Finally, it is worth noting that educators have long promoted a large number of different kinds of thinking: engineering thinking, science thinking, economics thinking, systems thinking, logical thinking, rational thinking, network thinking, ethical thinking, design thinking, critical thinking, and more. Each academic field claims its own way of thinking. What makes computational thinking better than the multitude of other kinds of thinking? I do not have an answer.

My conclusion is that computational thinking primarily benefits people who design computations and that the claims of benefit to non-designers are not substantiated.

Underlying all the claims is an assumption that the goal of computational thinking is to solve problems.

Conclusion

Promoters of computer science have long believed computational thinking is good for everyone. The definition of computational thinking evolved over 60 years applies mainly to those involved in designing computations whether in computer science or in other fields. The promoters of computer-science-for-all, believing that “designing computations” is an insular computer science activity, sought a broader, more encompassing definition to fit their aspiration. The result was a vague definition that targeted not only designers but all users of computational tools, anyone engaging in step-by-step procedures, and anyone engaging in a practice that could potentially be automated. Teachers who find the vagueness confusing have asked for a more precise definition that also clarifies how to assess student learning of computational thinking.

My advice to teachers and education researchers is: use Aho’s historically well-grounded definition and use competency-based skill assessments to measure student progress. Be wary of the claim of universal value, for it has little empirical support and draws you back to the vague definitions. Focus on helping students learn to design useful and reliable computations in various domains of interest to them. Leave the more advanced levels of computational design for education in the fields that rely heavily on computing.

In the late 1990s, we in computer science (including me) believed everyone should learn object-oriented programming. We persuaded the Educational Testing Service to change the Advanced Placement curriculum to an object-oriented curriculum. It was a disaster. I am now wary of believing that what looks good to me as a computer scientist is good for everyone. The proposed curriculum for computational thinking looks a lot like an extended object-oriented curriculum. This is not a good start for a movement aiming to define something greater than programming. Early warnings that the object-oriented vision was not working came from the front-line teachers who did not understand it, did not know how to assess it, and could not articulate the

^f If the architect were to specify how to erect the building by assembling 3D printed parts in a precise sequence, we could say the architect thought computationally for the manufacturing aspect but not for the whole design.

benefit for their students. We are now hearing similar early concerns from our teachers. This concerns me.

Underlying all the claims is an assumption that the goal of computational thinking is to solve problems. Is everything we approach with computational thinking a problem? No. We respond to opportunities, threats, conflicts, concerns, desires, etc by designing computational methods and tools—but we do not call these responses problem-solutions. It seems overly narrow to claim that computational thinking, which supports the ultimate goal of computational design, is simply a problem-solving method.

I have investigated three remaining trouble spots with computational thinking—the definition, the assessment methods, and the claims of universal benefit. It would do all of us good to tone down the rhetoric about the universal value of computational thinking. Advocates should conduct experiments that will show the rest of us why we should accept their claims. Adopting computational thinking will happen, not from political mandates, but from making educational offers that help people learn to be more effective in their own domains through computation. ■

References

- ACM. Computer Science Curriculum 2013; <https://www.acm.org/education/CS2013-final-report.pdf>
- Aho, A. Computation and Computational Thinking, 2011; <http://ubiquity.acm.org/article.cfm?id=1922682>
- Computing at School, a subdivision of the British Computer Society (BCS). 2015. Computational Thinking: A Guide for Teachers; <http://community.computingatschool.org.uk/files/6695/original.pdf>
- CSTA. Operational Definition of Computational Thinking, 2011; <http://www.csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf>
- Denning, P., Ed. Ubiquity symposium: What is computation? (Oct. 2011); <http://ubiquity.acm.org/symposia2011.cfm?volume=2011>
- Denning, P. and Martell, C. *Great Principles of Computing*. MIT Press, 2015.
- Denning, P. Beyond computational thinking. *Commun. ACM* 52, 6 (June 2009), 28–30; DOI: 10.1145/1516046.1516054
- Denning, P. Educating a new engineer. *Commun. ACM* 35, 12 (Dec. 1992), 82–97; 10.1145/138859.138870
- Dijkstra, E. My hopes for computing science. 1979; <https://www.cs.utexas.edu/users/EWD/transcriptions/EWD07xx/EWD709.html>
- Dreyfus, S. and Dreyfus, H. A five-stage model of the mental activities involved in directed skill acquisition. *Storming Media*, 1980; <http://www.dtic.mil/cgibin/GetTRD?oc?AD=ADA084551&Location=J2&doc=GetTRDoc.pdf>
- Dreyfus, H. *On the Internet*. Routledge 2003 (2d ed. 2008).
- Easton, T. Beyond the algorithmization of the sciences. *Commun. ACM* 49, 5 (May 2006), 31–33.
- Guzdial, M. HCI and computational thinking are ideological foes? *Computing Education Blog* 2011, (2/23/11); <https://computinged.wordpress.com/2011/02/23/hci-and-computational-thinking-are-ideological-foes/>
- Guzdial, M. *Learner-Centered Design of Computing Education: Research on Computing for Everyone*. Morgan-Claypool, 2015.

- International Society for Technology in Education. ISTE Standards for Students, 2016; <http://www.iste.org/standards/standards/for-students-2016>
- Jones, E. The trouble with computational thinking, 2011; <http://www.csta.acm.org/Curriculum/sub/CurrFiles/JonesCTOnePager.pdf>
- Kafai, Y. From computational thinking to computational participation in K–12 education. *Commun. ACM* 59, 8 (Aug. 2016), 26–27.
- Katz, D. The use of computers in engineering classroom instruction. *Commun. ACM* 3, 1 (Oct. 1960), 522–527.
- Knuth, D. Computer science and its relation to mathematics. *American Mathematical Monthly* 81, 4 (Apr. 1974), 323–343.
- Koschmann, T. Logo-as-Latin Redux. *J. Learning Sciences* 6, 4 Lawrence Erlbaum Associates, 1997.
- Mannila, L. et al. Computational thinking in K–9 education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference, ITICSE-WGR '14* ACM, NY, 2014, 1–29.
- National Research Council, Computer Science and Telecommunications Board. *Being Fluent with Information Technology*. National Academies Press, 1999.
- Newell, A., Perlis, A.J., and Simon. *Computer Science*, [letter] *Science* 157 (3795): (Sept. 1967), 1373–1374.
- Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, 1980.
- Papert, S. An exploration in the space of mathematics educations. *Int'l Journal of Computers for Mathematical Learning* 1, 1 (1996), 95–123; <http://www.papert.org/articles/AnExplorationintheSpaceofMathematicsEducations.html>
- Pea, R. and Kurtland, M. On the cognitive effects of learning computer programming. *New Ideas in Psychology* 2, 2 (1984), 137–168.
- Perkovic, L. et al. A framework for computational thinking across the curriculum. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education, ITICSE '10*, (2010), ACM, NY, 123–127.
- Polanyi, M. *The Tacit Dimension*. University of Chicago Press, 1966.
- Polya, G. *How to Solve it* (2nd ed.). Princeton University Press, 1957; <https://math.berkeley.edu/~gmlvin/polya.pdf>
- Simon, H. *The Sciences of the Artificial*, 3rd ed. MIT Press, 1969.
- Sizer, T.R. *Horace's School*. Houghton-Mifflin, 1992.
- Snyder, L. *Fluency with Information Technology*. Pearson, 2003 (6th edition 2014).
- Wilson, K. Grand challenges to computational science. In *Future Generation Computer Systems*. Elsevier, 1989, 33–35.
- Weise, M. and Christensen, C. Hire Education. Christensen Institute for Disruptive Innovation, 2014; <http://www.christenseninstitute.org/wpcontent/uploads/2014/07/Hire-Education.pdf>
- Wing, J. Computational thinking. *Commun. ACM* 49, 3 (Mar. 2006), 33–35; DOI: 10.1145/1118178.1118215
- Wing, J. Computational thinking—What and why? Carnegie-Mellon School of Computer Science Research Notebook (Mar. 2011). <https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why>.
- Wing, J. Computational thinking, 10 years later. Microsoft Research Blog (March 23, 2016); https://blogs.msdn.microsoft.com/msr_er/2016/03/23/computational-thinking-10-years-later/

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA, is Editor of ACM Ubiquity, and is a past president of ACM. The author's views expressed here are not necessarily those of his employer or the U.S. federal government.

I extend personal thanks to Douglas Bissonette, Mark Guzdial, Roxana Hadad, Sue Higgins, Selim Premji, Peter Neumann, Matti Tedre, Rick Snodgrass, and Chris Wiesinger for comments on previous drafts of this Viewpoint.

Copyright held by author.

Calendar of Events

June 5–7

Web3D '17: The 22nd International Conference on Web3D Technology
Brisbane, QLD, Australia,
Sponsored: ACM/SIG,
Contact: Matt Adcock,
Email: matt.adcock@csiro.au

June 5–8

ICMR '17: International Conference on Multimedia Retrieval
Bucharest, Romania,
Sponsored: ACM/SIG,
Contact: Niculae Sebe,
Email: sebe@disi.unitn.it

June 5–9

SIGMETRICS '17: ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems
Urbana-Champaign, IL,
Sponsored: ACM/SIG,
Contact: Bruce Hajek,
Email: b-hajek@uiuc.edu

June 6–10

SACMAT'17: The 22nd ACM Symposium on Access Control Models and Technologies (SACMAT)
Indianapolis, IN,
Sponsored: ACM/SIG,
Contact: Elisa Bertino,
Email: bertino@cerias.purdue.edu

June 7–9

PerDis '17: The International Symposium on Pervasive Displays
Lugano, Switzerland,
Sponsored: ACM/SIG,
Contact: Marc Langheinrich,
Email: marc.langheinrich@usi.ch

June 10–14

DIS '17: Designing Interactive Systems Conference 2017
Edinburgh, U.K.,
Sponsored: ACM/SIG,
Contact: Oli Mival,
Email: o.mival@napier.ac.uk

June 12–13

SCF '17: ACM Symposium on Computational Fabrication
Cambridge, MA,
Sponsored: ACM/SIG,
Contact: Stefanie Mueller,
Email: stefanie.mueller@student.hpi.uni-potsdam.de

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

Understanding the psychology of learning strategies leads to effective problem-solving skills.

BY DEVON H. O'DELL

The Debugging Mind-Set

SOFTWARE DEVELOPERS SPEND 35%–50% of their time validating and debugging software.¹ The cost of debugging, testing, and verification is estimated to account for 50%–75% of the total budget of software development projects, amounting to more than \$100 billion annually.¹¹ While tools, languages, and environments have reduced the time spent on individual debugging tasks, they have not significantly reduced the total time spent debugging, nor the cost of doing so. Therefore, a hyperfocus on elimination of bugs during development is counterproductive; programmers should instead embrace debugging as an exercise in problem solving.

It is more appropriate for programmers to focus their efforts on acquiring and encouraging effective learning

strategies that reduce the time spent debugging, as well as changing the way they perceive the challenge. This article describes how effective problem-solving skills can be learned, taught, and mentored through applying research on the psychology of problem solving, conducted by Stanford's Carol Dweck and others.

In the 1974 classic, *The Elements of Programming Style*, Kernighan and Plauger wrote, "Everyone knows that debugging is twice as hard as writing a program in the first place. So if you are as clever as you can be when you write it, how will you ever debug it?"⁶ Is debugging really twice as hard as writing a program? If so, why should it be?

To answer these questions, let's first consider why bugs occur. In a 2005 paper, Andrew J. Ko and Brad A. Meyers suggest that bugs occur as a result of "chains of cognitive breakdowns... formed over the course of programming activity."⁷

At a fundamental level, all software describes changes in the state of a system over time. Because the number states and state transitions in software can have combinatorial complexity, programmers necessarily rely on approximations of system behavior (called *mental models*) during development. The intent of a mental model is to allow programmers to reason accurately about the behavior of a system.

Since mental models are approximations, they are sometimes incorrect, leading to aberrant behavior in software when it is developed on top of faulty assumptions. Neophyte programmers experience this problem to a greater degree: their models of the language and development environment are necessarily incomplete, and even a simple syntax error can be a major blocker for someone learning to program. This is not limited to novice programmers. The C11 language specification is just north of 700 pages, for example. How many systems programmers understand the language completely?

The most sinister bugs occur when programmers falsely believe their men-



tal models to be complete. This is the crux of the problem: they have assumed correctness in their implementations and by definition do not know where they went wrong. The only way programmers can hope to solve such bugs is through knowledge acquisition. While it remains unclear whether Kernighan is correct about the quantitative difference in difficulty between programming and debugging, it does seem clear that debugging is the more difficult task. Since solving bugs requires learning, the debugging process can be made easier by better understanding effective learning and teaching strategies.

Teaching Debugging Skills

Debugging is simply a domain-specific term for *problem solving*. Bugs are described as word problems: for exam-

ple, “An out-of-bounds write to object O causes corruption of adjacent memory when conditions A and B are both true.” Unfortunately, word problems are found to be some of the most difficult to teach, and in software nearly all bugs can effectively be characterized as word problems.¹³

Perhaps to compensate for this difficulty, much existing research on the pedagogy of debugging focuses first on discriminating “experts” from “novices” and assessing the techniques each use in debugging tasks. It then attempts to improve the novice’s ability through teaching expert techniques. A review of the literature, however, finds that even experts differ greatly in debugging skill.⁸

Those most effective at debugging draw from extensive experience, as well

as refined problem-solving skills. They also employ generalized strategies for problem solving¹³ instead of treating every individual bug as a new, specific case. While teaching the techniques of expert programmers can reduce the time novices spend on programming activities, it has not been shown to be effective in reducing the time spent debugging. Furthermore, testing students from groups that have and have not received expert technique intervention does not yield statistically significant differences in test scores.²

Experience and accurate models are precisely the tools the novice programmer lacks. Although computer science education devotes a lot of time to teaching algorithms and fundamentals, it appears that not much of this time is spent applying them to general

problems. Debugging is not taught as a specific course in universities. Despite decades of literature suggesting such courses be taught, no strong models exist for teaching debugging.

The problems are what to teach and how to teach it. Carefully considering research in the field of psychology can be helpful in both understanding students' needs and designing appropriate curricula.¹³ Instructors, mentors, and educators should therefore be aware of the relevant research in this area to guide what they are teaching and how they are teaching it. Individuals must be cognizant of how to approach problems, and whether they are perceived as limits of ability or as part of the learning process.

A Note on Tools and Environments

Significant effort has been expended on developing tools, languages, and programming environments aimed at reducing or eliminating bugs. Tools (once their use is understood) undoubtedly save time in the debugging process, but they can't solve all problems. What if no tool exists for a specific problem? What if available tools don't scale to the load required to reproduce an issue? What if you are unaware of the tools, or unable to afford them? Tools are not a panacea; when they don't exist (or are otherwise unavailable for use), you need to be willing and able either to write them or to forego their use.

Virtual machine-based languages, interpreted languages, and languages with runtime environments encourage users to view the execution environment as a black box. The goal here is to make programming easier by reducing the scope of the mental model the programmer must maintain. When bugs occur in these execution environments, you are left with a complete gap in understanding. You might solve this by understanding more about the execution environments, but then what is the point of that abstraction? Such runtime environments are not a panacea; you still need to understand how they behave.

Finally, extensive research exists in the area of formally verifiable languages. These environments allow programmers to prove the correctness of their code while solving some prob-

lem, but they do not help us understand the problems they are solving in the first place. If you do not fully understand a problem and then implement a "provably correct" solution based on incomplete understanding, debugging will still be required. Such languages are not a panacea; you still need a complete understanding of the problems to be solved before writing your software.

Since bugs occur because of an incomplete mental model, the philosophy of relying entirely on tools, runtime environments, and languages to catch bugs seems self-defeating. It's almost as if we are saying that we cannot possibly be smart enough to understand these bugs, so why even bother? Such a view falls squarely to the side of the "entity theorist" as described by Carol Dweck's research on "self-theories of intelligence."⁵

Self-Theories of Intelligence

Dweck, a leading researcher in the field of motivation, is responsible for four decades of research that attempts to identify and characterize behaviors of high-achieving individuals. Her work has been repeatedly validated in many studies across individuals of varying cultures, genders, ages, and socioeconomic backgrounds.

Dweck proposes that individuals fall somewhere on a spectrum of self-theories. This spectrum varies from the *entity theorist* on one end to the *incremental theorist* on the other. An entity theorist tends to view intelligence as innate and fixed, and fundamentally believes that not much can be done to increase intelligence. An incremental theorist fundamentally believes that challenging problems are a core part of the learning process and that intelligence is malleable: it can be gained through hard work. When confronted with challenges, entity theorists interpret them as limits of their abilities and do not try hard to solve them. In literature, an entity theory is frequently referred to as a *fixed mindset* and an incremental theory as a *growth mindset*—owing largely to the underlying motivations of the entity and incremental theorists.

Though Dweck's work initially focused on theories of intelligence, these theories of self can be applied to domain-specific beliefs. For example,

individuals can hold an incremental theory toward intelligence, while still having entity-theory views on specific skills such as programming or debugging,¹² so this research does not imply global views.

Individuals with an entity theory of intelligence are more likely to be motivated by appearance than performance, are more likely to engage in maladaptive behaviors (such as giving up or cheating) in the face of problems, and are less likely to collaborate with others. In the software development field, these behaviors manifest through maladaptive strategies including hero worship, impostor syndrome, "cargo culting," and lackadaisical work ethic.

Incremental theorists by contrast are motivated by performance-oriented goals. Whereas the entity theorist takes shortcuts to appear smart (such as tacking a large number of objectively simple tasks), the incremental theorist tackles hard problems head-on, seeing hard work as a necessity to gain knowledge and deliver value. Furthermore, incremental theorists tend to work effectively in groups, value helping educate and promote the success of others, and engage in other behaviors generally positive for professional programming environments.

This incremental theory of intelligence is clearly beneficial to programmers and characteristic of the best in the field, whether professionally or academically oriented. It would be useful to figure out how to apply Dweck's research effectively, recognizing that the practice of programming (like many other fields) gains from lifelong learning. Educators, managers, and mentors would benefit from incorporating this research into courseware, management, and teaching strategies.

The key to applying Dweck's research to our field is that it is possible to "shift" on the spectrum of self-theories. By changing the way you react to, respond to, and praise folks, you can help move others (and yourself) from a "fixed" to a "growth" view.

Self-Theories in Computer Science

How capable are you at programming compared with when you started? At debugging? Whether you have been at it for one year or 50, you are almost cer-

tainly solving problems that seemed intractable when you began your learning journey. This isn't a result of innate ability: before we started programming, none of us had this ability. We don't have to look very far back in history to understand that ability in this area is nascent among humanity.

Still, we must carefully consider psychological research before applying it to our work¹³ and decide if it is relevant. In a 2008 paper, Laurie Murphy and Lynda Thomas enumerate ways in which embracing such research can help in the field of computer science:⁹

- ▶ Applying this research to introductory computer science courses may improve student learning and course retention rate.

- ▶ The gender gap in computer science education and practice may be directly attributable to cultural influences on self-theories of intelligence in women.


- ▶ Collaborative work, such as pair programming, is found in some studies to be wildly effective—and wildly ineffective in others. Conflicting goals of entity versus incremental theorists could explain these apparent contradictions.

- ▶ Defensive classroom climates, characterized by students who ask “pseudo-questions” to demonstrate knowledge and professors who afford special status to those students, could be explained and ameliorated by adopting Dweck's work.


As bugs present themselves as problems, and because problems tend to be perceived by the entity theorist as fundamental limits on ability, the focus here should be on moving students, peers, and co-workers toward a more malleable view of intelligence.

Organizational Adoption of Malleable Self-Theories

Whether operating as a manager, mentor, or educator, adopting and promoting malleable self-theories is important in creating successful students and co-workers. Debugging must not be an afterthought in educating; industry must stop insisting that bugs be interpreted as failures of individual programmers (especially since individual programmers are rarely responsible for the design and functioning of an entire system). Programmers should instead



Whether operating as a manager, mentor, or educator, adopting and promoting malleable self-theories is important in creating successful students and co-workers.



be praised for their efforts in solving bugs. In all cases, solving bugs is part of the learning process. How can they be presented as such?

Several studies have attempted to move individuals' views from a fixed to a malleable perspective. A familiarity with the literature is important here, especially for educators and managers. In a 2004 paper, Mantz Yorke and Peter Knight suggest that educators should “appreciate the significance of self-theories for student learning; be able to infer whether students are inclined toward fixedness or malleability; and possess strategies for encouraging ‘fixed’ students to move toward malleability.”¹⁴

To do this, Murphy and Thomas suggest looking at psychologist Lev Vygotsky's work from the early 20th century. They state, “students learn best when pushed slightly beyond their independent capabilities.” This requires an instructor or mentor capable of assisting students past their current ability; Yorke and Knight observe that this system works best when both the teacher and student possess a malleable view of intelligence.

Moving an individual from an entity to an incremental view can be as simple as framing information in a particular way. In multiple studies, Dweck notes that information presented as praise of *ability* promotes formation of an entity theory, whereas praise for *effort* promotes an incremental theory. This appears to be true whether or not the individual receiving the information is the individual being praised.

Dweck states that the temporal effect of praise-based shifts is unclear. In one set of individuals in their study, Cutts et al. consistently reinforced an incremental mindset every time feedback was given on graded work.³ Only students receiving this intervention (coupled with two other intervention methods) saw movement toward incremental self-theories *and* statistically significant improvement in test scores (a feat that no other study I have found has been able to show). It may be that consistent environmental feedback promoting an incremental theory is sufficient for an individual's long-term adoption. Given that it is shown to work in short-term situations, con-

sistent feedback may simply be a functional equivalent.

Most examples of such feedback fall into framing information in a way that promotes growth-oriented goals. When a colleague solves a particularly nasty bug, people tend to say, “You’re brilliant!” Instead they should say, “Great job on the hard work!” If you task engineers with work that is very easy for them to solve, you can be apologetic: “I’m sorry for assigning you a task you couldn’t learn much from.” Try to direct more challenging work to those colleagues in the future. Students and individuals will face frustration when solving bugs. If you, your students, or your colleagues voice frustration, try to frame it in terms of what you or they can expect to learn upon completing the project.

Unfortunately, students and job candidates are generally not able to choose teachers or managers who hold such views. Because learning is best achieved when both students and teachers hold incremental theories,¹⁴ it is crucial that organizations hire individuals possessing such views and train existing staff on the material.

Instructors and tutors possessing an entity theory are more likely to focus on helping only those they perceive to be the brightest, writing off the struggling students as lost causes. This is particularly unfortunate since in several studies, Dweck shows that underachieving students with a malleable mindset moving into harder problems (as happens in transitions from elementary school to middle school, or from high school to college) tend to outperform high-achieving students with an entity theory. Instructors with an entity theory are less likely to help the underperforming students succeed.

Reinforcing Incremental Theories for Individuals

Education is a lifelong process. As individuals, we should adopt malleable views of intelligence in daily life. There are specific ways of approaching problems that develop and reinforce a malleable view of intelligence.

Active recall. One way people may force themselves into entity-framework thinking is an overreliance on reference manuals and documentation. Are the arguments to `memcpy` in the

order of source, destination, length; or are they destination, source, length? Is `strstr` haystack, needle; or is it needle, haystack? Many individuals develop a habit of consulting references (such as system manuals) as soon as the question comes up.

Active recall is a study method in which you first make a guess before looking up the solution. (This is the basis upon which study tools like flash cards are built, but it must be employed properly to be effective.) In the case of interfaces such as `memcpy` and `strstr`, write the code first: take your best guess at the argument ordering. Once you have done this, look at the reference manual to confirm whether you have done it correctly.

Segmented study. Think of the last time you spent an entire day on a marathon debugging session. Did you solve the problem? Or did you need to take a break and work on something else for a bit, perhaps going home, sleeping, then solving the problem the next day?

Engineers frequently engage in sunk-cost fallacies while debugging, expending additional effort solving a problem despite diminishing returns for the time spent. Our brains are not tuned to focus on specific tasks for hours on end. Segmented study is the idea of having one or two additional and unrelated tasks to switch to over the course of an activity. Switching gears and taking breaks are (perhaps counterintuitively) effective methods for making progress when you are stuck.

For managers, mentors, and educators, this is also true. Explicitly allow colleagues and students time to work on other problems. Students and employees will not usually ask to work on something else, so it is crucial to grant folks the time they need to process information.

Persevere. What *segmented study* doesn’t mean is giving up when things get difficult. You must persevere: tenacity and passion are at least as important as intelligence to both success and skill development. The most successful individuals in a field practice their craft for years at the edge of their ability. This “grit” is highly correlated with individual success factors.⁴

Be curious. With sufficient experience, it becomes appropriate to mentor others. When teaching and men-

toring others, it is vital to be curious. Many paths exist to arrive at some knowledge, and your own intrinsic views are not always the best for everybody. When learning, people want to relate new knowledge to existing knowledge; this gives them a strong foundation upon which to build. Curiosity is about being open to ideas, solutions, and methods of thinking that do not necessarily reflect your own views. You must teach from the perspective of those you wish to educate.

A General Approach to Debugging

Through continued learning, malleable views of problems, and effective use of tools, you can become successful in debugging. Still, some insist that debugging is more of an art than a science. I think we can dispatch this idea entirely. It is clear that debugging requires learning, and the scientific method is specifically designed to yield new knowledge. The method, summarized: (1) Develop a general theory of the problem. (2) Ask questions leading to a hypothesis. (3) Form a hypothesis. (4) Gather and test data against the hypothesis. (5) Repeat.

In my experience, very little attention is paid to the formation of a hypothesis, resulting in a wasted effort, testing without any theory pertaining to the cause of the bug. Forming a good hypothesis is rather more difficult than it would seem. In practice, hypotheses are poorly formed, and many rely more on intuition than information gathering. Intuition can be an effective strategy for debugging but requires extensive experience and, when used as the only strategy, leaves the programmer unprepared to handle new, unfamiliar bugs. Lacking a framework for solving these bugs is particularly damaging for entity theorists.

A good hypothesis describes a problem and is both testable and falsifiable. In fact, forming a proper hypothesis almost always implies that a bug is fully understood. Consider the following three statements:

- ▶ A bug exists in the logging module.
- ▶ A race condition exists in the logging module when concurrent log producers enqueue the same item.
- ▶ A race condition between concurrent log producers consuming pooled work objects is caused by an improper-

ly handled return code when the pool is empty, and later results in a double-free when the consumer attempts to reenqueue the same object multiple times into a pool but fails because the pool is then full.

These statements were part of a thought process around solving a real bug. The first hypothesis is indicative of very little planning or research, and is the result of the “hunch” programmers have about what could constitute a bug. This is a testable hypothesis, but it is poor: if this hypothesis is confirmed through testing, the testing is unable to provide any more data on how to solve the problem.

The second statement is marginally better. It's clear that it is operating on more information, and so it seems like the bug has been reproduced at this point. This hypothesis is still incomplete, because it does not make any predictions as to why concurrent log producers would produce the same item. Furthermore, though it sounds like it describes what the failure is (a race condition), this is not actually the terminal flaw, as described in the third hypothesis.

This third hypothesis is clearly the best. It describes both why the bug happens and what the failure is. Importantly, it identifies that the cause of failure occurs separately from where and when the program actually fails. This hypothesis is great because it can be very specifically tested. If regression tests are part of your development framework, only this hypothesis provides a description of how such a test should behave.

Falsifiability is an important and crucial property of a real hypothesis. If a hypothesis cannot be proven false, any test will confirm it. This cannot possibly give you confidence that you understand the issue.

Forming a sound hypothesis is important for other reasons as well. Mental models can be used to intuit the causes of some bugs, but for the more difficult problems, relying on the mental model to describe the problem is exactly the wrong thing to do: the mental model is incorrect, which is why the bug happened in the first place. Throwing away the mental model is crucial to forming a sound hypothesis.

This may be harder than it seems. For example, comments in code suspected to contain bugs may reinforce existing mental models. This may cause you to paper over buggy code, thinking it is obviously correct. Consider, for example:

```
/* Flush all log entries */
for (i = 0; i <= n_entries; i++)
{ flush_entry(&entry[i]); }
```

This code (maybe obviously) illustrates an example of an off-by-one error. The comment above it is correct but incomplete. This code *will* flush all entries. It will also flush one more. When debugging, treat comments as merely informative, not normative.

Conclusion

Debugging is one of the most difficult aspects of applied computer science. Individuals' views and motivations in the area of problem solving are becoming better understood through far-reaching research conducted by Carol Dweck and others. This research provides a means to promote continued growth in students, colleagues, and yourself.

Debugging is a science, not an art. To that end, it should be embraced as such in institutions of higher learning. It is time for these institutions to introduce entire courses devoted to debugging. This need was suggested as far back as 1989.¹⁰ In 2004, Ryan Chmiel and Michael C. Loui observed that “[t]he computing curriculum proposed by the Association for Computing Machinery and the IEEE Computer Society makes little reference to the importance of debugging.”² This appears to still be true.

Learning solely through experience (suggested by Oman et al. as the primary way debugging skills are learned¹⁰) is frustrating and expensive. At a time when the software engineering industry is understaffed, it appears that individuals with certain self-theories resulting from social and cultural influences are left behind. Understanding Dweck's work and changing the way we approach education, mentorship, and individual study habits can have a profound long-term effect on the progress of the software development industry. While research into tools to ease the

task of debugging continues to be important, we must also embrace and continue research asking and showing how better to help students, colleagues, and peers toward success in computer science. □

Related articles on queue.acm.org

Undergraduate Software Engineering

Michael J. Lutz, J. Fernando Naveda, and James R. Vallino

<http://queue.acm.org/detail.cfm?id=2653382>

Coding Smart: People vs. Tools

Donn M. Seeley

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

Interviewing Techniques

Kode Vicious

<http://queue.acm.org/detail.cfm?id=1998475>

References

1. Britton, T., Jeng, L., Carver, G., Cheak, P. and Katzenellenbogen, T. Reversible debugging software. Cambridge Judge Business School, 2013; <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.444.9094&rep=rep1&type=pdf>.
2. Chmiel, R., Loui, M.C. 2004. Debugging: from novice to expert. *SIGCSE Bulletin* 36, 1 (2004), 17–21.
3. Cutts, Q., Cutts, E., Draper, S., O'Donnell, P. and Saffrey, P. Manipulating mindset to positively influence introductory programming performance. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 2010, 431–435.
4. Duckworth, A.L., Peterson, C., Matthews, M.D., Kelly, D.R. Grit: Perseverance and passion for long-term goals. *J. Personality and Social Psychology* 92, 6 (2007), 1087–1101.
5. Dweck, C. *Self-theories: Their Role in Motivation, Personality, and Development*. Psychology Press, 1999.
6. Kernighan, B.W. and Plauger, P.J. *The Elements of Programming Style*. McGraw-Hill, 1974.
7. Ko, A.J. and Meyers, B.A. A framework and methodology for studying the causes of software errors in programming systems. *J. Visual Languages and Computing* 16, 1-2 (2005), 41–84.
8. McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L. and Zander, C. Debugging: a review of the literature from an educational perspective. *Computer Science Education* 18, 2 (2008), 67–92.
9. Murphy, L., Thomas, L. 2008. Dangers of a fixed mindset: implications of self-theories research for computer science education. *SIGCSE Bulletin* 40, 3 (2008), 271–275.
10. Oman, P.W., Cook, C.R. and Nanja, M. Effects of programming experience in debugging semantic errors. *J. Systems and Software* 9, 3 (1989), 197–207.
11. RTI. The economic impacts of inadequate infrastructure for software testing, 2002: <http://www.nist.gov/director/planning/upload/report02-3.pdf>.
12. Scott, M. and Ghinea, G. On the domain-specificity of mindsets: the relationship between aptitude beliefs and programming practice. *IEEE Transactions on Education* 57, 3 (2014), 169–174.
13. Winslow, L. Programming pedagogy—A psychological overview. *SIGCSE Bulletin* 28, 3 (1996), 17–22.
14. Yorke, M. and Knight, P. Self-theories: some implications for teaching and learning in higher education. *Studies in Higher Education* 29, 1 (2004), 25–37.

Devon H. O'Dell is a tech lead at Fastly, where his primary focus includes mentorship of team members and the scalability, functionality, and stability of Fastly's core caching infrastructure. Previously, he was lead software architect at Message Systems and contributed heavily across the Momentum high-performance messaging platform.

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

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

Embrace failure so it does not embrace you.

BY PAT HELLAND, SIMON WEAVER, AND ED HARRIS

Too Big NOT to Fail

WEB-SCALE COMPUTING MEANS running hundreds of thousands of servers. It requires a fundamentally different approach from smaller environments. Consistent server hardware and datacenter plans, as well as consistent and simple configurations, are essential. Everything is designed to expect and embrace failure without human intervention. Operations must be largely autonomous. Software must assume crashes. Development, test, and deployment must have integrated and automated solutions.

This article offers a brief survey of many of the techniques so well known to those working at Web scale and yet frequently surprising to others.

Web-scale infrastructure implies *lots* of servers working together—often tens or hundreds of thousands of servers all working toward the same goal. How can the complexity of these environments be managed? How can commonality and simplicity be introduced?

A lot of effort goes in to achieving uniform and fungible hardware resources in Web-scale datacenters. If you can have just one kind of server with the same CPU, same DRAM, same storage, and the same network capacity, then any server is as good as the next server. When all the servers are the same, there is a single pool of spares and a single resource to allocate.

You want to treat your hardware





resources like a bag of nails. Any single nail doesn't matter very much. It's the aggregate number of useful nails that gets the house built. Typically, you buy too many nails, expecting a certain failure rate. When a nail bends or breaks, you don't get too emotional about it.

Similarly, when a server in the data-center breaks, you cannot get emotional. It happens all the time.

Variety leads to complexity. Complexity leads to unpredictability and a lack of website availability.

The law of large numbers is probably true. The law of large numbers says that individual pieces may be unreliable but the aggregate expected failure rate is predictable. While you cannot know the exact outcome of the toss of a pair

of dice, you can know the expected rate over time. The more tosses you have, the better the confidence in the average.

Similarly, in a large datacenter, the more servers you run, the easier it is to plan for expected failures. The larger the set of servers, the more likely something is broken. Figure 1 shows a typical server failure rate.

It's predictability that matters, not reliability. Stuff can fail... Stuff *will* fail. With tens of thousands of servers, lots of stuff fails every day! You just need to predict how often.

Hardware at Web Scale

Typical Web-scale datacenters think about hardware differently from typical IT shops. They avoid variety and stick to

common hardware as often as possible. It's assumed that each piece of hardware may very well fail and the software running on it needs to work successfully when that happens. Furthermore, the inexorable pace of improvement means it's not cost effective to keep hardware too long. The new stuff will offer more "bang for the buck" and for the electricity it consumes.

Bespoke is baroque. When something is *bespoke*, you tweak and modify it until it is perfect. In some environments, systems and servers are designed with their own special configuration.

Baroque is an architectural style with a lot of ornate detail and tremendous variety. In a large-scale environment, bespoke becomes baroque. The individ-

ual details of the various types of servers become overwhelming. The aggregate system with its large number of server types is full of detail and complexity.

Attack of the clones. In a typical datacenter, you pick a standard server configuration and insist everyone use the same type. Just like Henry Ford's Model T, you can have any color you like as long as it's black.

With one SKU (stock keeping unit) to order, you gain huge leverage with vendors in buying servers. In addition, there is a single pool of spares for that SKU.

Now, there are some exceptions. Each company is phasing in a new server type while phasing out an old one. Also, it's common to have a very few special servers—such as one for compute loads and one for storage.

Still, tightly controlling the variety of server types is essential.

The short life of hardware in the datacenter. Messing with stuff in the datacenter causes problems. Upgrading servers can cause inconsistencies. Just don't do it! When repairing, you really only want to replace the server with an identical spare and then repave its software. Maybe the broken server can be fixed and become a spare.

Servers and other gear provide less value over time. New servers offer more computation and storage for the same form factor and same electricity. The value for the electrical cost diminishes.


Datacenter hardware is typically decommissioned and discarded (or returned to its lessor) after three years. It's just not worth keeping.

Datacenter servers and roast beef are worth a lot less after a few years.

That means there is a lot of pressure to place new servers into production quickly. Say it takes two months to commission, activate, and load data into new servers. In addition, it may take one month to decommission the servers and get the data out of them. That is three months out of a total three-year lifetime not productively used. The life cycle of servers is a big financial concern.

Operations at Web Scale

Operations at Web scale are very different from operations at smaller scale. It's not practical to be hands on. This leads to autonomous datacenter management.



Typical Web-scale datacenters think about hardware differently from typical IT shops. They avoid variety and stick to common hardware as often as possible.



People don't scale—dermatological issues notwithstanding. Say your operations staff can manage 100 servers per person. This is very typical in a DevOps environment. Each server needs attention to validate state, handle failures, and perform repairs. Frequently, the servers come in many flavors and you want to optimize the hardware for each server. Does everyone understand all the server types? What are their different operational tasks?

Some 50,000 servers at 100 servers per person require 500 operators. This gets out of hand very quickly as you scale.

Zen and the art of datacenter maintenance. To support Web-scale datacenters, we have had to evolve from Ops to DevOps to NoOps, as detailed in Figure 2. Historically, with manual operations, people not only decided what to do, but also did all the actual work required to operate the servers. DevOps is a huge step forward as it automates the grunt work of operations. Still, this is not adequate when scaling to tens of thousands of servers. In a NoOps or autonomously managed system, the workflow and control over the operational tasks are also automated.

Software at Web Scale

Software must embrace failures with pools of stateless servers and special storage servers designed to cope with the loss of replicas.

Stateless servers and whack-a-mole. Stateless servers accept requests, may read data from other servers, may write data to other servers, and then return results. When a stateless server dies, its work must be restarted. Stateless servers are designed to fail.

Stateless servers must be idempotent. Restarting the work must still give the correct answer. Learning about idempotent behavior is an essential part of large-scale systems. Idempotence is not that hard!

Frequently, stateless servers run as a pool of servers with the number increasing or decreasing as the demand fluctuates. When a server fails, the demand increases on its siblings, and that likely will cause a replacement to pop back to life. Just like the arcade game whack-a-mole, as soon as you hit one, another one pops up.

To each according to its need. Concurrent requests for a stateless service

need assignment to servers. Load balancing requests across a pool of servers is much like spraying work.

When requests to servers are taking longer than hoped, it is likely because there is a queue waiting for the individual servers. If you are spraying work across more servers, then the per-server queue is shorter. This will result in a faster response.

Adding servers to a server pool usually lowers the response time for requests. Removing servers reclaims resources. Given a response-time goal, this can be done by an automated robot.

Avoiding memory loss from traumatic server injury. Most distributed storage systems keep each piece of data on three separate servers in three separate racks in a datacenter. The choice of three replicas is a function of the durability goals, the MTBF (mean time between failures), and the MTTR (mean time to repair).

$$\text{Availability} = \text{MTBF} / (\text{MTBF} + \text{MTTR})$$

Since we assume one in five servers fail every year, our MTBF (Mean Time Between Failures) is relatively short. To improve availability, we either need a longer MTBF or a shorter MTTR (Mean Time To Repair). By shortening our MTTR, we can dramatically improve our availability and data durability.

Assume the data contained in each server is cut into pieces and the pieces have their additional replicas on many different servers, as shown in Figure 3. For example, the data on server-A is cut into 100 pieces. Each of those pieces has its secondary and tertiary replicas on different servers, perhaps as many as 200 total in addition to server-A. If server-A fails, the other 100 secondary servers will try to find a new place on potentially yet another 100 servers. In the limit, this parallelism can reduce the MTTR by 100 times.

Notice in Figure 3 that each slot of data in server-B (S9, S2, S4, S8, and S5) has two other replicas on different servers. If server-B fails, each of these slots is replicated onto a new third server. The placement of the new third replica preserves the guarantee that each replica lives in a separate server.

This approach is tried and true with GFS (Google File System),² HDFS (Hadoop Distributed File System),³

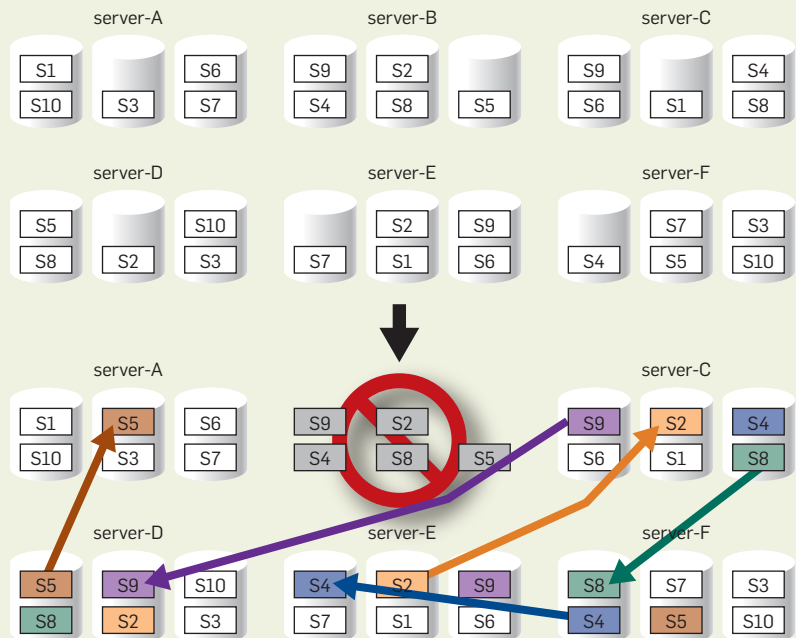
Figure 1. Typical server failure rate.

Assume 4 SATA Disks per Server
 Assume 4% disk failure rate per year
 4 disks * 4% per disk means 16% of the servers fail from disk
 Assume 4% miscellaneous failures (e.g. power supplies, etc)
 20 % of servers fail each year
 1 in 1825 servers fails each day (1825 = 5 * 365)
50,000 servers → expect 27 servers to fail each day

Figure 2. Ops, DevOps, and NoOps.

| Tasks | Manual Operations "Ops" | Automated Operations "DevOps" | Autonomous Operations "No Ops" |
|---------------------------------|-------------------------|-------------------------------|--------------------------------|
| Who sets the goals? | Human | Human | Human |
| Who decides when to start? | Human | Human | Machine |
| Who adjudicates priorities? | Human | Human | Machine |
| Who does the work? | Human | Machine | Machine |
| Who generates validation data? | Human | Machine | Machine |
| Who interprets validation data? | Human | Human | Machine |
| Who handles failures? | Human | Human | Machine |
| Who handles exceptions? | Human | Human | Human |

Figure 3. Data replication.



Microsoft Bing's Cosmos distributed file system,¹ and others leveraging this technique.

Large datacenters need to consider two essential aspects of this approach. First, software-driven robotic recovery is essential for high availability. Humans take too long and would be overwhelmed by managing the failures. Second, the larger the cluster, the more storage servers are possible. The more storage servers, the smaller the pieces smeared around the cluster and the faster the recovery time.

Development, Test, and Deployment at Web Scale

Development and testing in large-scale Web environments are best suited for sharing resources with production. Successfully developing, testing, and deploying software presents many challenges. To top it off, you are pretty much constantly working to keep up with the changing environment.

The isolation ward. It's important to concentrate datacenter resources. Separating datacenters for development and test from production may be tempting, but that ends up creating problems. Managing demand and resources is difficult when they are separate. It is also hard to ensure that production is in sync with dev/test and that you are testing what you will run in production.

Developing and testing using production datacenters means sharing resources. You must separate the resources using containers or VMs (virtual machines). Your customers will be using the same servers as your testers so you must isolate the production data. In general, development and test personnel must not have access to customer data to comply with the security team.

While ensuring safe isolation of workloads presents many challenges, the benefits outweigh the hassles. You don't need dedicated datacenters for dev/test, and resources can ebb and flow.

Sign-off, rollout, canaries, and rollback. The management of software change in a large-scale environment has complexities, too. Formalized approvals, rollout via a secure path, automated watchdogs looking for problems, and automated rollback are all essential.

Formal approvals involve release rules. There will be code review and automated test suites. The official signoff typically involves at least two people.

Rollout to production includes code signing whereby a cryptographic signature is created to verify the integrity of the software. The software is released to the needed servers. Sometimes, tens of thousands of nodes may be receiving the new version. Each node is told the code signature, and it verifies that the correct bits are there. The old version of the software will be kept side-by-side with the new one.

A few servers are assigned to be *canaries* that will try the new version before all the others. Like the little birds taken into underground mines to check for dangerous gases (the canaries would die of the gas before the miners, who would then skedaddle to safety), in software releases, the automated robot tries a few servers first, and only when successful, rolls out more.

Rollback is the automated mechanism to undo the deployment of a new version. Each server keeps the old version of the software and can pop back to it when directed.

Conclusion

Running large datacenters requires some fundamental changes in approach. Everything must be simpler, be automated, and expect failure.

A simple model for server and service behavior. Web-scale systems run on three important premises:

► *Expect failure.* Any component may fail and the system continues automatically without human intervention.

► *Minimal levers.* The underlying system provides support for deployment and rollback. When a server is sick, it's better to kill the whole server than deal with partial failures.

► *Software control.* If it can't be controlled by software, don't let it happen. Use version control for everything with human-readable configuration.


Reliability is in the service, not the servers!

Embrace failure so it doesn't embrace you. Running hundreds of thousands of servers requires a different approach. It requires consistent hardware (servers and network) with minimal variety. Datacenters must have simple and predictable configurations.

They must embrace failure and automatically place services onto the hardware as needed.

Operations must evolve from manual to automated to autonomous. Humans should set the goals and handle major exceptions. The system does the rest.

Software must embrace failures with pools of stateless servers and special storage servers designed to cope with the loss of replicas.

Integrated development, test, and deployment are built deeply into the system with controlled and automated deployment of software. 

Related articles on queue.acm.org

Order from Chaos

Natalya Noy

<http://queue.acm.org/detail.cfm?id=1103835>

Resilience Engineering: Learning to Embrace Failure
A discussion with Jesse Robbins, Kripa Krishnan, John Allspaw, and Tom Limoncelli

<http://queue.acm.org/detail.cfm?id=2371297>

Schema.org: Evolution of Structured Data on the Web

R.V. Guha, Dan Brickley, and Steve Macbeth

<http://queue.acm.org/detail.cfm?id=2857276>

References

1. Chaiken, R., Jenkins, R., Larson, P.-A., Ramsey, B., Shakib, D., Weaver, S., Zhou, J. SCOPE: Easy and efficient parallel processing of massive data sets. In *Proceedings of ACM VLDB*, 2008; <http://www.vldb.org/pvldb/1/1454166.pdf>
2. Ghemawat, S., Gobioff, H. and Leung, S.-T. The Google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, 2003, 29–43.
3. Shvachko, H., Kuang, H., Radia, S., Chansler, R. Hadoop Distributed File System. In *Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies*, 2010, 1–10.

Pat Helland has been implementing transaction systems, databases, application platforms, distributed systems, fault-tolerant systems, and messaging systems since 1978. For recreation, he occasionally writes technical papers. He currently works at Salesforce.

Ed Harris leads the Infrastructure Compute team at Salesforce.com, building the compute, network, and storage substrate for the world's most trusted enterprise cloud. Prior to Salesforce, Ed worked in the Bing Infrastructure at Microsoft, working on the Cosmos big data service.

Simon Weaver has been devising and building distributed and autonomous systems for 18 years solving problems in the fields of big data, lights-out infrastructure, and artificial intelligence. Simon currently works at Salesforce.

**Great engineers are able
to maximize their mental power.**

BY KATE MATSUDAIRA

Conversations with Technology Leaders: Erik Meijer

THERE ARE SMART people in the world. And then there are *really* smart people. You know the ones I am talking about—those who are so impressive that it doesn't matter what they do (academia, programming, engineering, or management); you know if they are doing it, then they are doing it well.

For this article, I wanted to share with you some of my favorite engineering and leadership lessons from one of the smartest people I know: **Erik Meijer**.

Whether you are a leader, a programmer, or just someone aspiring to be better, I am sure there are some smart takeaways from our conversation that will help you grow in your role. Oh, and if you read to the end, you can find out what his favorite job interview question is—and see if you would be able to pass his test.

What qualities make someone into an amazing engineer?

There is a paper called “The Humble Programmer,”³ and even though it was written in 1972, after all these years it is still super-relevant. In the early days of computing, programming was perceived as puzzle solving and optimizing the computational process—it is astonishing how those archaic ideas are still there when it comes to interviewing developers. Our world today is very complicated—we are dealing with distributed systems, all kinds of models, neural nets, frameworks, new languages. We don't have the mental power to keep on top of every new in-

novation and idea. Mental power is our most precious resource.

Part of this is being able to leverage the *power of abstraction*—focusing on what is important and leaving out the unnecessary details. Sometimes details are important; other times they are not. We cannot talk about everything in absolute terms. Compared to assembly code, C is declarative. But compared to transistors and gates, assembly code is declarative. Developers need to recognize these levels of abstractions.

A good engineer knows how to handle leaky abstractions and can swiftly go up a level or dive deeper down when needed. But that same engineer also has to accept that you can never understand everything.

We need to be asking, “How can we design systems so that computers can handle more of the work for us?” For example, a lot of developers are still creating programs as text. A lot of the tools we use to manipulate programs are still too primitive because they treat code as sequences of bytes. We need to be much more mindful of how we can use computers to help do our jobs.


The whole point of “The Humble Programmer” is that your brainpower is your most limited resource, so using smart tools is a good thing. Good developers understand that they can’t do everything, and they know how to leverage tools as prosthetics for their brains.

Kate’s takeaway: You should read (or reread) “The Humble Programmer.” And always be on the lookout for ways to work smarter—better tools, intelligent systems, and enlisting help. Focus your mental energy on the task with the most dividends.


What qualities make someone into an amazing engineering manager?

First, you must have deep technical knowledge. But it’s also important to have self-awareness, empathy, and emotional intelligence. You have to be able to understand other people, and you have to be able to steer people and move people.

In management, there is a communication feedback loop. In one direction, a manager interacting with his or her reports requires emotional intelligence. He or she has to know what drives the other person to get optimal results. A great manager will help people do their best work.



“Once you get stuck and stop pushing yourself, you are toast.”



The second part of the loop is the reports back to their manager, and the skill that matters here is empathy. You have to understand what they are trying to say despite the noisy channel between you and your reports.

Each direction of the loop introduces uncertainty. You might hear something, but that isn’t what was said, and vice versa. It is your job as the manager to make sure this communication is optimized. It is on you to make sure that the feedback loop is a virtuous cycle—the better you understand your reports, the more empathy and emotional intelligence you have in that relationship.

By taking a Bayesian approach, you can increase your empathy by performing error correction on what you hear and increase your emotional intelligence by inserting redundancy into your communication. One way you get that error correction and redundancy is through peer feedback and 360 reviews to train your neural net continuously.

In your mind you create a model of someone. When something happens, you hear something or observe something; then you are updating your prior assumptions. This is where you must watch your biases. In the beginning, you don’t know anything about someone, but the more interactions you have over time, the more the uncertainty in your model diminishes.

Kate’s takeaway: The feedback loop is an interesting way to think about your interactions and relationships. If you want another lens on a similar topic, Erik wrote a paper on “The Responsive Enterprise” that talks about these loops in an organizational context.⁴

What book do you wish all software engineers would read, and why?

How to Win Friends and Influence People.¹ That book gives you really complete ways of thinking about human relationships and how you interact with other people. It is written in a way that makes you consider the lessons by putting yourself in other people’s shoes. How do they think or feel in these situations? And what can you do differently?

I print out the Wikipedia summary of the book and glue it into the notebooks I carry around everywhere to



keep notes. Every two weeks I reread the rules and refresh myself into doing the right thing.

The books I recommend for managers are ones by Jeffrey Pfeffer and Robert Sutton (professors at Stanford) since they are more evidence driven. A lot of business books are about what people believe, but there is hardly any proof.

Kate's takeaway: Whether it is *How to Win ...* or another book, figure out your own rules and revisit them regularly. Without some sort of external stimulus, most of us will fall back into our default modes of socially awkward introvert, and so a paper taped to the inside of your planner or notebook is a smart idea.

What is the best piece of career advice you have ever received?

When I did my Ph.D., afterward in the celebration, my advisor, Kees Koster, said to focus at the intersection of theory and practice. There is no progress without friction.

It is easy to dive into theory, or all the way into just practice—but the real interesting work happens between theory and practice. Try to understand

both sides. The safe spot is to retreat to one of the extremes.

There are so many online courses these days, so many blogs, and so many white papers that it is easier than ever to stay up to speed on both sides. You can subscribe to Adrian Colyer's *The Morning Paper*,² go through the ACM Digital Library, read the Research for Practice column in *acmqueue*—a lot of people are making it easier to bridge gaps. Going back to “The Humble Programmer,” understand that you can't keep up with all of the knowledge that is produced. You don't have to throw your hands in the air and say it is too much—you have to hone your Google skills.

Kate's takeaway: It is never enough just to do what is obvious. You have to dig deep. Devote time in your schedule to learning new things. Try to read a white paper per week, or per month.

What is your team process? How does work get done? How do you communicate status?

A lot of what you read about process and agile has very little evidence be-

hind it. I don't believe a lot of process is scientific. Instead, I define general guidelines about what I want to see happen, and within those I don't care how things happen.

My thinking has two main sources of inspiration: the military and the hacker way.

Over thousands of years, armies have figured out how to get things done and achieve their goals in an environment that is really chaotic and completely unpredictable. That is the environment we live in as developers as well. If you read the U.S. Marine Corps *Warfighting* manual, and replace the word *war* with *software*, everything in there holds true.

So how do you deal with uncertainty? When people attempt to solve with process, they are trying to fight or control uncertainty. For example, someone can say just adopt zero inbox and your life will be awesome. In reality, though, that isn't really the case.

One of the things I like about Facebook is “the hacker way.”⁶ It is an approach to creating software that involves continuous improvement and feedback. It is about computational thinking: how do you program the system, and how do

you make the system do things that no one thought was possible?

Being agile is about communication. The process needs to change with the situation. You have to have a big picture of where you want to go, but any plan or process will shatter immediately when you hit your first bug or something happens out of your control.

In most projects there are two phases: an explorative phase and an execution phase. Your project should progress like a damped sine wave, where the amplitude gets smaller over time (see the accompanying figure). You have to figure out what to build, and figure out what question you are trying to answer. In the beginning you want to increase the vertical velocity to get uncertainty under control, and then you want horizontal velocity to increase when you get into execution.

With prescriptive processes, people are looking for a silver bullet to solve problems, but it doesn't exist. It comes back again to "The Humble Programmer." The world is super-confusing, and you have to embrace it and work with it.

Kate's takeaway: You have to make your process work for you. Imagine your projects progressing on a damped sine wave—first focus on finding the right questions, and then the answers.

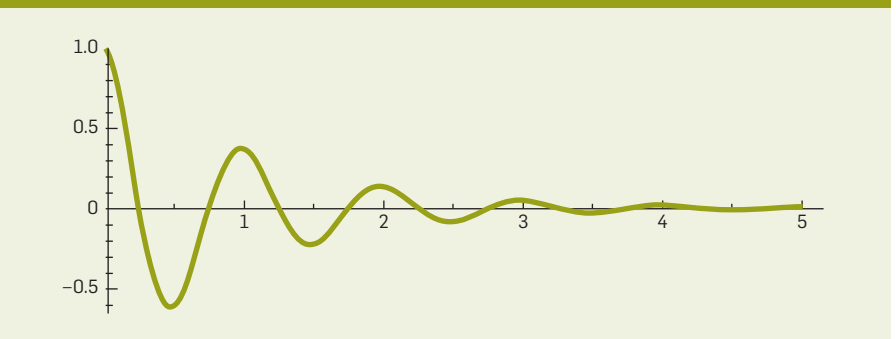
Who is the best manager you ever worked for? What made this person so great?

William Adams. He was my manager at Microsoft. He is an inspiration, and I am still trying to emulate him in my work.

There are several things I like about him. One is the importance he sees in diversity. For example, when dealing with feedback loops and prior assumptions, you need diversity to challenge your thinking. You have to actively put energy into creating a diverse environment so you are always challenging the status quo and maybe resetting your accumulated state. Don't get stuck in a local optimum.

The other thing is that he always focused on people first. You want to create the circumstances where everyone can focus on their strengths. Always find the best job for the person. Try to get a sense of the progress and circumstances so you can get ahead of what is next. For example, if the project is winding down,

Damped sine wave.



make sure there is always a pipeline of new ideas. You have to make sure the pipelines are set up so they never stall—keeping things innovative.

Kate's takeaway: Think about the people around you. Do you have enough different opinions to keep your team out of a local optimum? How can you get more diversity?

What are the common mistakes that even good engineering managers sometimes make?

Your prior assumptions are not higher order—that is, you are not making assumptions about your assumptions. I keep reading *How to Win Friends ...* because I understand it is easy to fall back into my default behavior. That is the big thing: your work is never done. You never know and you aren't perfect. There is always stuff to learn. You have to keep up with your trade and keep learning. You have to keep pushing yourself to get better.

Once you get stuck and stop pushing yourself, you are toast.

Kate's takeaway: Think about some of the past lessons you've learned. What could you use a refresher on? What are some new things you want to learn?

What is your best interview question?

Given a generic type `Cont r a = (a -> r) -> r`, prove that this type forms a monad.

If you try to solve this question by brute force, you are going to fail. But if you look at it from the right level of abstraction, it is easy. So it forces you to problem solve your own problem-solving skills.

The particular formulation using monads and type sounds really theoretical, but it is super practical. When you are using JavaScript to write event

handlers for button clicks, you are using continuations. It is a microexample of everything above in one single type.

Being a great developer is difficult. It requires constant learning and a passion for technology and science. The same thing is true for great technical leaders. There are a lot of smart lessons, but perhaps the most important one is always to be pushing yourself, and to be smart about your brainpower and energy (working smart). □

Related articles on queue.acm.org

Lean Software Development: Building and Shipping Two Versions

Kate Matsudaira

<http://queue.acm.org/detail.cfm?id=2841311>

It Probably Works

Tyler McMullen

<http://queue.acm.org/detail.cfm?id=2855183>

A Conversation with Erik Meijer and José Blakeley

<http://queue.acm.org/detail.cfm?id=1394137>

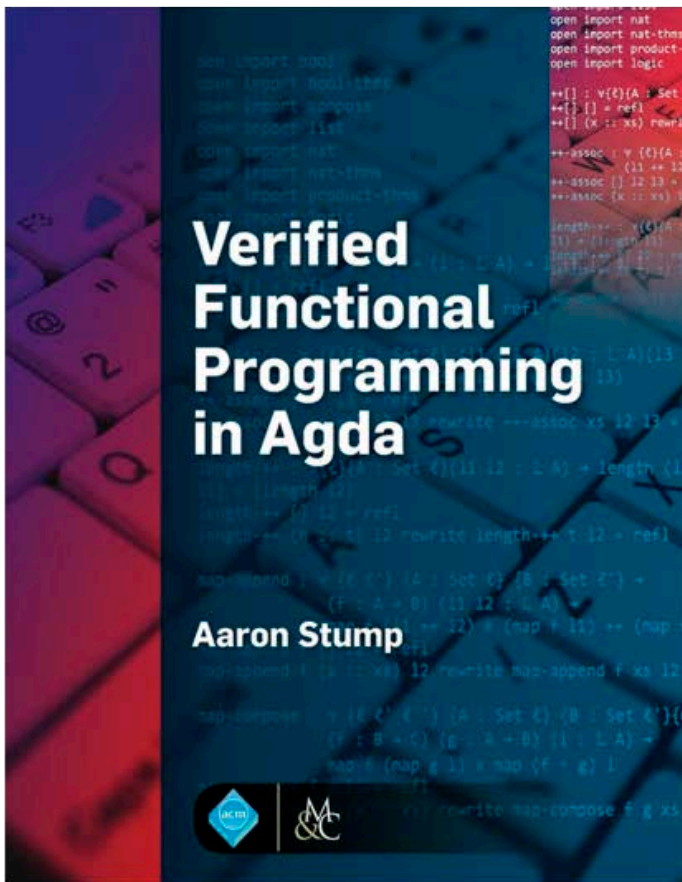
References

1. Carnegie, D. *How to Win Friends and Influence People*. Simon and Schuster, 1936.
2. Colyer, A. The Morning Paper; <https://blog.acolyer.org>.
3. Dijkstra, E.W. The humble programmer. *Commun. ACM* 15, 10 (Oct. 1972), 859–866; <http://dl.acm.org/citation.cfm?id=361591>.
4. Meijer, E., Kapoor, V. The responsive enterprise. *acmqueue* 12, 10 (2014); <http://queue.acm.org/detail.cfm?id=2685692>.
5. U.S. Marine Corps. *Warfighting*, 1997; <http://www.marines.mil/Portals/59/Publications/MCDP%201%20Warfighting.pdf>.
6. Zuckerberg, M. Mark Zuckerberg's letter to investors: "The hacker way". *Wired*; <https://www.wired.com/2012/02/zuck-letter/>.

Erik Meijer has been working on "democratizing the cloud" for the past 15 years. He is known for his work on, among others, the Haskell, C#, Visual Basic, and Dart programming languages, as well as for his contributions to LINQ and the Rx (Reactive Extensions) framework.

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

Copyright held by authors/owners.
Publication rights licensed to ACM. \$15.00



“When people ask me about Agda, I just give them this book.”

*Jesper Cockx
K. U. Leuven*

Agda is an advanced programming language based on Type Theory. Agda’s type system is expressive enough to support full functional verification of programs, in two styles. In external verification, we write pure functional programs and then write proofs of properties about them. The proofs are separate external artifacts, typically using structural induction. In internal verification, we specify properties of programs through rich types for the programs themselves. This often necessitates including proofs inside code, to show the type checker that the specified properties hold. The power to prove properties of programs in these two styles is a profound addition to the practice of programming, giving programmers the power to guarantee the absence of bugs, and thus improve the quality of software more than previously possible.

Verified Functional Programming in Agda is the first book to provide a systematic exposition of external and internal verification in Agda, suitable for undergraduate students of Computer Science. No familiarity with functional programming or computer-checked proofs is presupposed.

The book begins with an introduction to functional programming through familiar examples like booleans, natural numbers, and lists, and techniques for external verification. Internal verification is considered through the examples of vectors, binary search trees, and Braun trees. More advanced material on type-level computation, explicit reasoning about termination, and normalization by evaluation is also included. The book also includes a medium-sized case study on Huffman encoding and decoding.



<http://books.acm.org>

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

DOI:10.1145/3085591

Adhering to the end-to-end principle even more than the current Internet yields highly available point-to-point communication.

BY DAVID BARRERA, LAURENT CHUAT, ADRIAN PERRIG, RAPHAEL M. REISCHUK, AND PAWEŁ SZALACHOWSKI

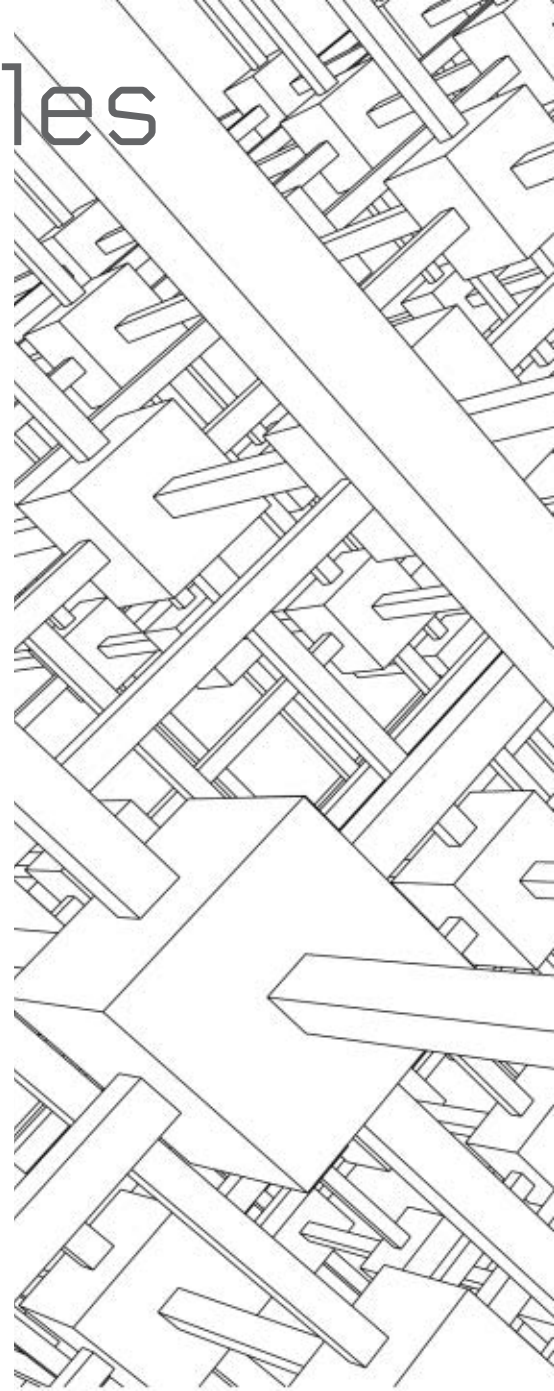
The SCION Internet Architecture

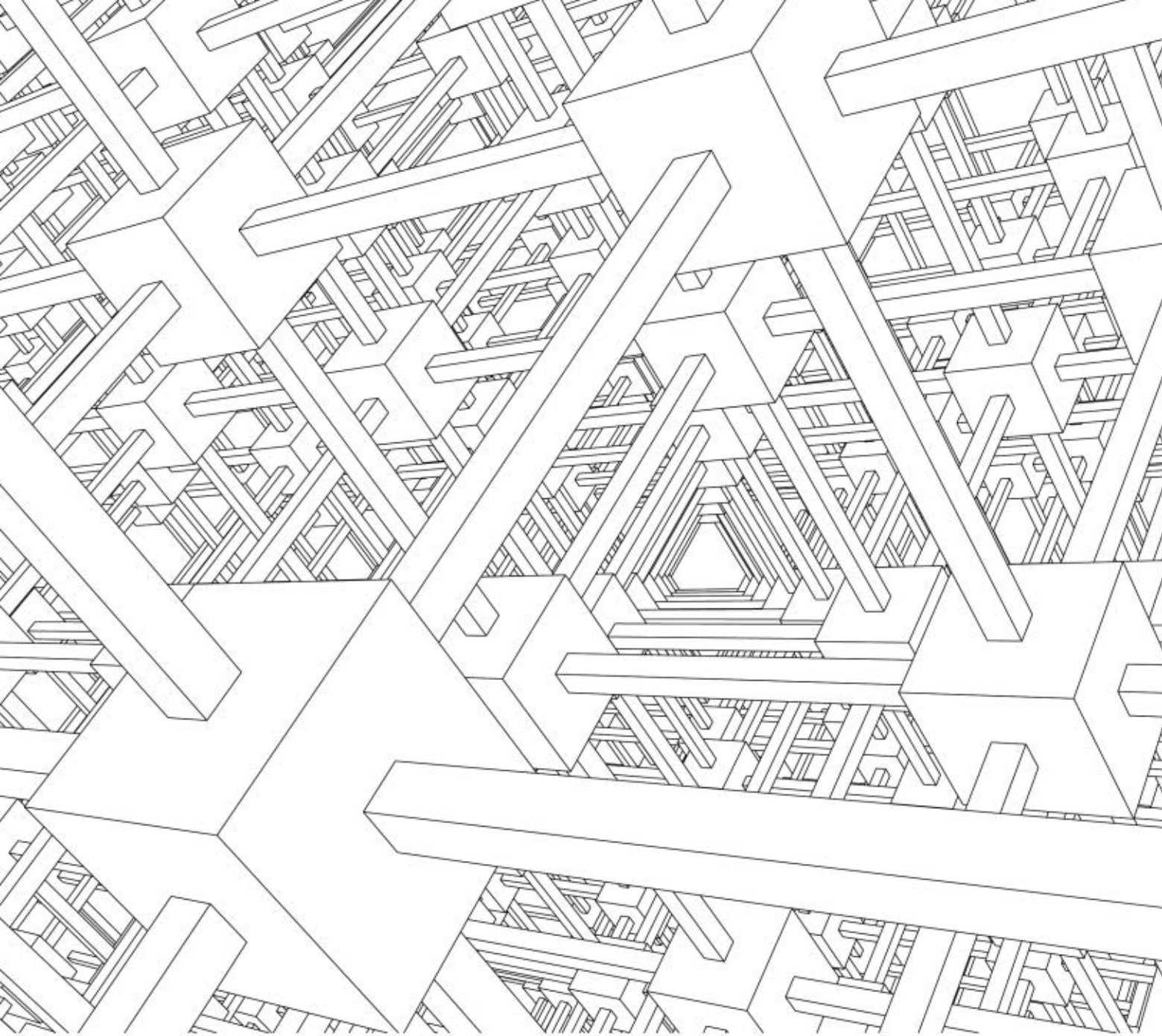
THE INTERNET HAS been successful beyond even the most optimistic expectations. It permeates almost every aspect of our society and economy worldwide. This success has created universal dependence on communication, as many of the processes underpinning modern society would grind to a halt if it were unavailable. However, the state of the safety and availability of the Internet is far from commensurate with its importance.

Although we cannot conclusively determine what the impact of even a one-minute outage of Internet connectivity would be, anecdotal evidence suggests that even a brief outage would have a profound negative effect on governmental, economic, and societal operations.¹¹ Making matters worse, the Internet is not designed primarily for high availability in the face of malicious actions by adversaries. Recent patches to improve Internet security and availability are indeed constrained by the design of the current

» key insights

- Patching the current Internet is an undesirable long-term solution; a clean-slate redesign of inter-domain routing would provide many benefits and is surprisingly simple to deploy using legacy protocols for intra-domain communication.
- SCION's isolation domains offer control-plane isolation and scoped trust; rather than restrict communication, they provide transparency for path selection, packet forwarding, and authentication.
- SCION's packet-carried forwarding state eliminates the need for inter-domain routing table lookups, improves forwarding performance, and supports multipath communication; packet-carried forwarding state gives path control to senders, providing scalability, security, and availability benefits.





Internet architecture. A new Internet architecture must offer availability, security by design, and incentives for deployment, as well as address economic, political, and legal issues at the design stage.

Such features require a completely new cohesive architecture that provides one fundamental building block—highly available point-to-point communication—on which other proposed Internet architectures that provide content-centric,^{15,21} extensibility-centric,¹⁴ or mobility-centric²³ properties can build.

This article describes SCION, or Scalability, Control, and Isolation On Next-generation networks, an inter-domain network architecture

designed to address these issues, covering SCION's goals, design, and functionality, as well as the results of six years of research we have conducted since our initial publication.²⁸

Objectives

We begin with the high-level goals an inter-domain point-to-point communication architecture must be able to accomplish.

Availability in the presence of adversaries. Our aim is to offer a point-to-point communication infrastructure that remains highly available even in the presence of distributed adversaries; as long as an attacker-free path between endpoints exists, that path

can be discovered and used with guaranteed bandwidth between the endpoints, and is an exceedingly challenging property to achieve.

An “on-path adversary” may drop, delay, or alter packets instead of forwarding them or inject packets into the network. The architecture must thus provide mechanisms to counteract malicious operations. An “off-path adversary” could launch a hijack attack to attract traffic to flow through network elements under its control. Such traffic attraction can take several forms; for instance, an adversary could announce a desirable path to a destination by using forged paths or attractive network metrics. Conversely, the adversary

could render paths not traversing its network less desirable (such as by inducing congestion). An adversary controlling a large botnet could also perform distributed denial-of-service (DDoS) attacks, congesting selected network links. And an adversary could interfere with the discovery of legitimate paths (such as by announcing bogus paths).

Transparency and control. When the network offers path transparency, end hosts know (and can verify) the forwarding path taken by network packets. Applications that transmit sensitive data can benefit from this property, as packets are ensured of being able to traverse certain Internet service providers (ISPs) and avoid others.

In addition to path transparency, we aim for SCION to achieve end-host “path control,” a stronger property that allows receivers to select the incoming paths through which they are reachable and senders to select the end-to-end path. This seemingly benign requirement has multiple repercussions that are beneficial but also fragile if implemented incorrectly.

The beneficial aspects of path control include:

Separation of network control plane and data plane. Ensuring that forward-

ing cannot be retroactively influenced by control plane operations (such as routing changes);

Enabling multipath communication. Improving availability by allowing senders to select multiple paths to their destinations; and

Defending against network attacks. Including DDoS and traffic interception by rogue networks, since destinations can observe a packet’s traversed path in the packet header.

Particular care must be taken for the proper handling of the fragile aspects of communication, including:

Respecting ISPs’ forwarding policies. By offering policy-compliant paths from which senders can choose;

Preventing malicious path creation. Including paths that contain loops;

Ensuring scalability of path control. By allowing sources to select paths from among a relatively small set, as opposed to full-edged source routing; and

Enabling ISP traffic engineering. Despite end hosts’ path control, giving ISPs the ability to balance their load across the links to their neighbor autonomous systems (ASes).

Transparency and control over trust roots. Roots of trust are used to verify entities in the current Internet, as in

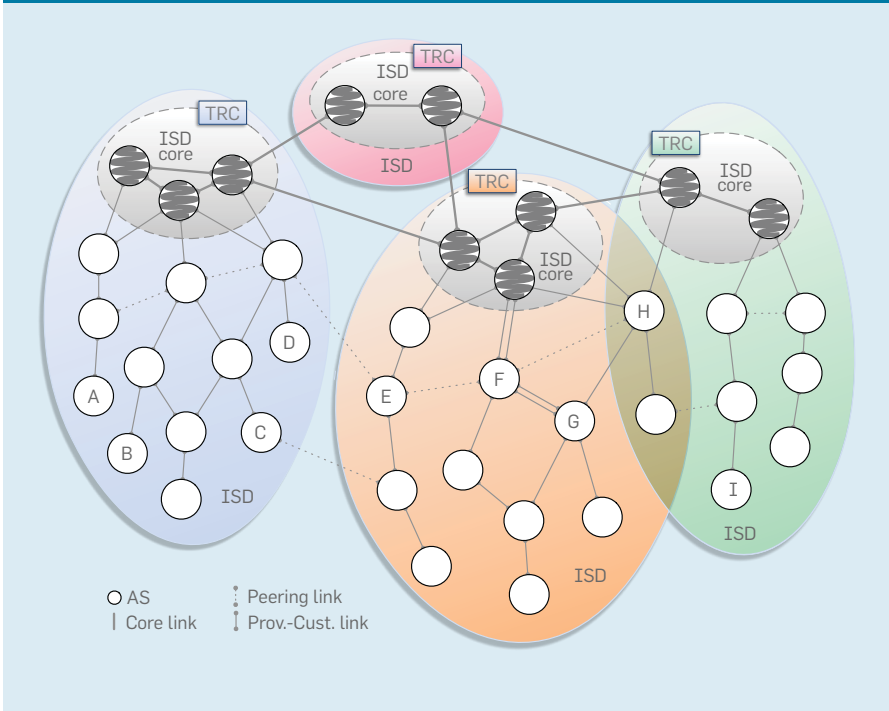
verification of a server’s public key in a Transport Layer Security (TLS) certificate or of a Domain Name System (DNS) response in DNSSEC (DNS Security Extensions).⁵ Transparency of trust roots provides end hosts and users knowledge of the complete set of trust roots relied upon for entity-certificate validation. Enumerating trust roots is difficult due to intermediate certification authorities that are trusted implicitly. Control over trust-root selection enables trust agility, allowing users to readily select or exclude the roots of trust they wish to rely upon.

Efficiency and scalability. Despite the lack of availability and transparency, the current Internet also suffers from efficiency and scalability deficiencies; for instance, the Border Gateway Protocol (BGP) has scaling issues in cases of network fluctuations, where routing protocol convergence can take minutes²⁴ or even days.⁸ Moreover, routing tables have reached the limit of their scalability due to multi-homing and prefix de-aggregation or announcement of more-specific IP address spaces. Increasing memory size for routing tables is problematic, as the underlying hardware is expensive and power-hungry, accounting for approximately one-third of a router’s total power consumption.

Security and high availability usually come at a cost, resulting in less efficiency and potentially diminished scalability. High performance and scalability are, however, required for economic viability. We thus explicitly seek high efficiency such that packet-forwarding latency and throughput are at least as fast as current IP forwarding. Moreover, we seek improved scalability compared to the current Internet, most notably with respect to BGP and to the growing size of routing tables.

One approach for achieving efficiency and scalability is to avoid router state wherever possible. We thus aim to place state into packet headers and protect that state cryptographically. Since modern block ciphers (such as AES) can be computed faster than performing DRAM memory lookups, packet-carried state can enable greater packet processing speeds and simpler router architectures compared to today’s IP routers. Avoiding state on routers also prevents state-exhaustion

Figure 1. ASes grouped into four ISDs. Core ASes are connected through core links. Non-core ASes are connected through customer-to-provider or peering links. Some ASes are contained in multiple ISDs.



attacks²⁶ and state inconsistencies across routers. Our goal of efficiency and scalability is in line with the end-to-end principle, which states that a function should be implemented at the network layer in which it can operate most effectively.²⁵ Since the end host has the most information about its own internal state, network functions related to that state (such as error detection and correction, acknowledgment of receipt, and retransmission) are handled by the end host. Moreover, SCION end hosts are involved in path selection, as they have the knowledge of preferred or undesirable network paths; that is, SCION adheres to the end-to-end principle even more than the current Internet.

Extensibility. To future-proof SCION, we designed the core architecture and code base to be extensible such that additional functionality are easily built and deployed. SCION end hosts and routers should—without overhead or expensive protocol negotiations—be able to discover the minimum common feature set supported by all intermediate nodes.

Support for global but heterogeneous trust. Given the diverse nature of the constituents in the current Internet, with its multiple legal jurisdictions and interests, an important challenge is how to scale authentication of entities (such as AS ownership for routing, name servers for DNS, and domains for TLS) to the global environment. The roots of trust of currently prevalent public key infrastructure (PKI) models (monopoly and oligopoly) do not scale to a global environment because mutually distrustful entities cannot agree on a single trust root (monopoly model) and because the security of a plethora of roots of trust is only as strong as its weakest link (oligopoly model). We thus seek a trust architecture that supports meaningful trust roots in a global environment with inherently distrustful entities.

Deployability. A new Internet architecture should offer a multitude of features that incentivize its deployment. We thus aim for SCION to provide high availability even under control-plane and data-plane attacks (thanks to built-in DDoS defenses), path transparency and control, trust-root transparency and control, robustness to configura-

tion errors, fast recovery from failure, high forwarding efficiency, and multipath forwarding. Economic and business incentives are also critical, making it possible for ISPs to define new business models and sell new services.

Migration to the new architecture must involve minimal added complexity (and cost) to the existing infrastructure. Deployment should be possible by utilizing an ISP's internal switching infrastructure and require only installation or upgrade of a few border routers. Moreover, configuration of the new architecture must be similar to the existing architecture (such as in the configuration of BGP policies), minimizing additional personnel training.

Foundation for other architectures. To achieve a simple, scalable, secure, efficient architecture, we now focus on the most basic communication mode: point-to-point communication. Other architectures that provide support for higher-level properties (such as for content distribution,^{15,21} extensibility,¹⁴ and mobility²³) all require a working point-to-point communication infrastructure.

SCION Architecture

SCION introduces the concept of isolation domain (ISD), a building block for achieving high availability, transparency, scalability, and support for heterogeneous trust, constituting a logical grouping of ASes, as outlined in Figure 1.

An ISD is administered by multiple ASes that form the ISD core; we refer to them as “core ASes.” The ISD is governed by a policy we call “trust root configuration” (TRC), which is negotiated by the ISD core. The TRC defines the roots of trust used to validate bindings between names and public keys or addresses.

An AS joins an ISD by purchasing connectivity from another AS in the ISD. Joining an ISD constitutes acceptance of the ISD's TRC. We envision ISDs spanning areas with uniform legal environments that provide enforceable contracts. If two ISPs have a contract dispute they are unable to resolve by themselves, such a legal environment would provide an external authority to resolve the dispute. All ASes within an ISD also agree on the TRC, or the entities that operate the trust roots and set the ISD policies. One possible model is thus for ISDs to

be formed along national boundaries or federations of nations, as entities within a legal jurisdiction can enforce contracts and agree on a TRC. ISDs can also overlap, so an AS may be part of several ISDs. Although an ISD ensures isolation from other networks, the central purpose of an ISD is to provide transparency and support heterogeneous trust environments.

SCION includes two levels of routing—*intra-ISD* and *inter-ISD*—that use “path-segment construction beacons” (PCBs) to explore routing paths, as outlined in Figure 2a.

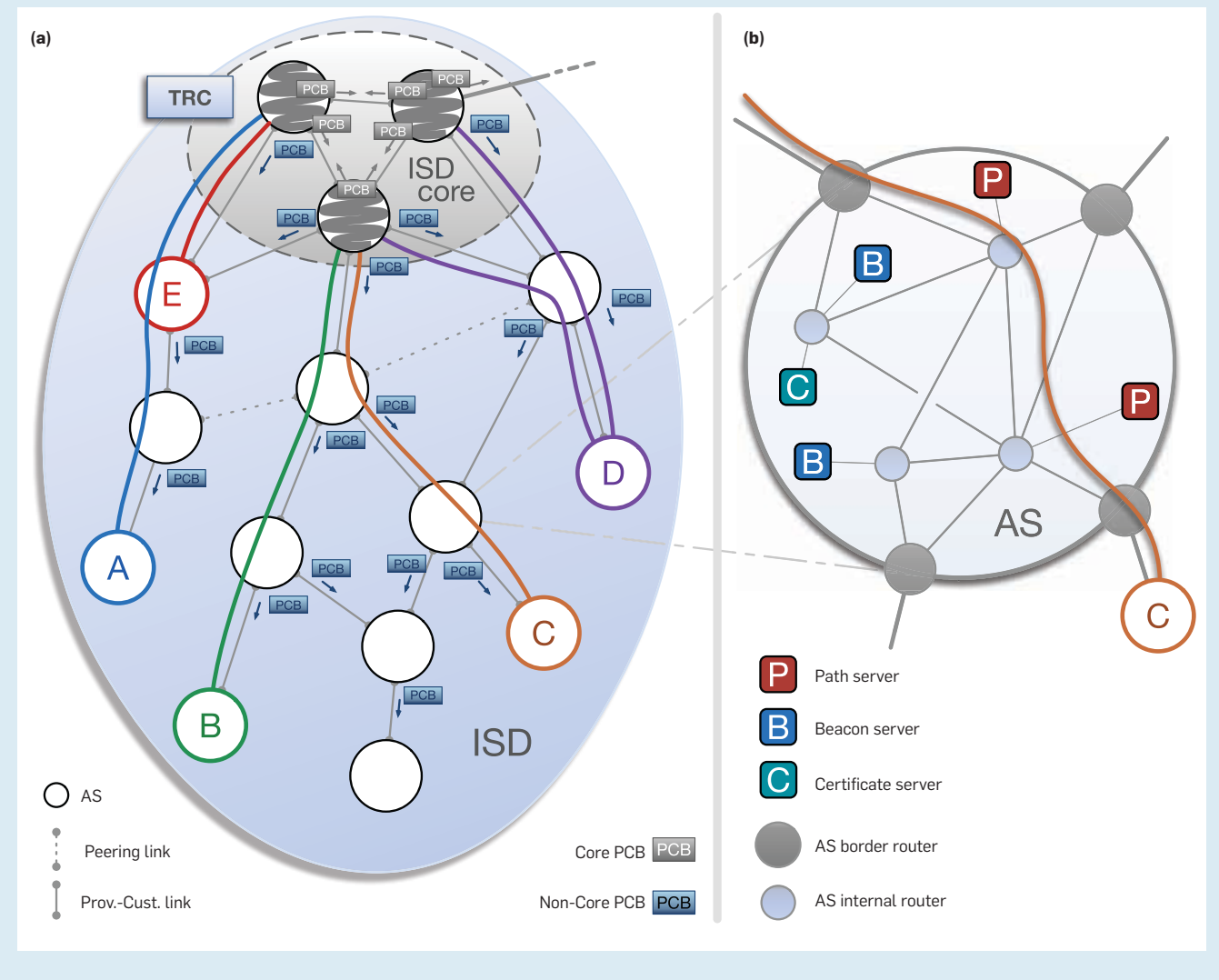
A core AS announces a PCB and disseminates it as a policy-constrained multi-path flood either within an ISD (to discover *intra-ISD* paths) or among core ASes (to discover *inter-ISD* paths), a process we call “beaconing.” PCBs accumulate cryptographically protected AS-level path information as they traverse the network. These protected contents within received PCBs are chained together by sources to create a path segment that enables packets to traverse a sequence of ASes. Packets thus contain AS-level path information, avoiding the need for border routers to maintain *inter-domain* routing tables, a concept we call “packet-carried forwarding state” (PCFS).

Through beaconing, ASes identify paths between themselves and core ASes. Path registration allows ASes to turn a few selected PCBs into path segments and make them available to other ASes. Path resolution then allows end hosts to create a forwarding path to the destination. This process consists of path lookup, where an end host obtains path segments to the destination, and path combination, where a forwarding path is created from the path segments.

Control plane. The control plane is responsible for discovering paths and making those paths available to end hosts.

Servers and routers. Figure 2b outlines the main AS components that perform control-plane operations in SCION, whereby beacon servers discover path information, path servers disseminate path information, and certificate servers assist with validating path information. In addition, border routers provide connectivity between ASes, while internal routers forward packets

Figure 2. SCION components at different scales: (a) SCION ISD with PCBs propagated from the ISD core down to customer ASes, and path segments for ASes A, B, C, D, and E to the ISD core; and (b) magnified view of an AS with its routers and servers. The path from AS C to the ISD core traverses two internal routers.



inside ASes. We did not include name servers in Figure 2b, as their operation is similar to today’s DNS.

Beacon servers are responsible for disseminating PCBs, as in Figure 2a. Beacon servers in a core AS generate intra-ISD PCBs that are sent to non-core ASes of the ISD. Non-core AS beacon servers receive these PCBs and re-send them to their customer ASes, resulting in AS-level path segments. Figure 3 outlines PCBs propagated from the ISD core down to customer ASes. At every AS, information about the AS’s interfaces is added to the PCB. The beacon servers generates a set of PCBs it forwards to its customer ASes. In the case of inter-ISD communication, the beaconing process is similar to BGP’s route-advertising process, although it is periodic and

PCBs are flooded through multiple paths over policy-compliant paths to discover multiple paths between any pair of core ASes. SCION’s beacon servers can be configured to implement current BGP policies, as well as additional properties (such as control of upstream ASes) BGP is unable to express.

Path servers store mappings from AS identifiers to sets of such announced path segments and are organized as a hierarchical caching system similar to today’s DNS. ASes, through the master beacon servers, select the set of path segments through which they want to be reached, uploading them to a path server in the ISD core.

Certificate servers store cached copies of TRCs retrieved from the ISD core, store cached copies of other ASes’

certificates, and manage keys and certificates for securing intra-AS communication. Beacon servers require certificate servers when validating the authenticity of PCBs.

Border routers forward packets between ASes supporting SCION. In the case of a control packet, the border router forwards it to the appropriate server, and, in the case of a data packet, forwards it either to a host inside the AS or toward the next border router.

Since SCION can operate using any communication fabric inside an AS, the internal routers do not need to be changed.

Path exploration and registration. Through inter-domain beaconing, core ASes discover paths to other core ASes. Through intra-domain beaconing,

ASes discover path segments leading to core ASes that enable an AS to communicate with the ISD core; Figure 2a outlines path segments from ASes *A*, *B*, *C*, *D*, and *E* to the core. The beaconing process is asynchronous; that is, the PCB generation is local, based on a per-AS timer, and PCBs are not propagated immediately upon arrival.

Paths are represented at AS-level granularity, which by itself is insufficient for fine-grain path diversity; ASes often have several diverse connection points, and a disjoint path is possible despite the AS sequence being identical. For this reason, SCION encodes AS ingress and egress interfaces as part of the path, exposing a finer level of path diversity. Figure 3 outlines this feature; AS *F* receives two different PCBs via two different links from the core. Moreover, AS *F* uses two different links to send two different PCBs to AS *G*, each with the respective egress interfaces. AS *G* extends the two PCBs, forwarding both over a single link to its customer.

An important requirement of the architecture is that SCION also supports peering links between ASes. Consistent with AS policies in the current Internet, PCBs do not traverse peering links, though peering links are announced, along with a regular path in a PCB. Figure 3 outlines how AS *F* includes its two peering links in the PCB. If the same peering link is announced in two path segments, then the peering link can be used to shortcut the end-to-end path without going through the core. SCION also supports peering links that cross ISD boundaries, highlighting the importance of SCION's path-transparency property; a source host knows the exact set of ASes and ISDs traversed during the delivery of each packet.

An AS typically receives several PCBs representing path segments to various core ASes. Figure 2a outlines two path segments for AS *D*. We call a path segment that leads toward an ISD core an “up-segment” and a path segment that leads from the ISD core to an AS a “down-segment,” though path segments are typically bi-directional and thus support packet forwarding in both directions. More precisely, up-segments and down-segments are invertible; by flipping the sequence of ASes, an up-segment is converted to a down-segment and vice versa. Path

servers learn up-segments by extracting them from PCBs they obtain from the local beacon servers. Path servers in core ASes also store core-segments to reach other core ASes.

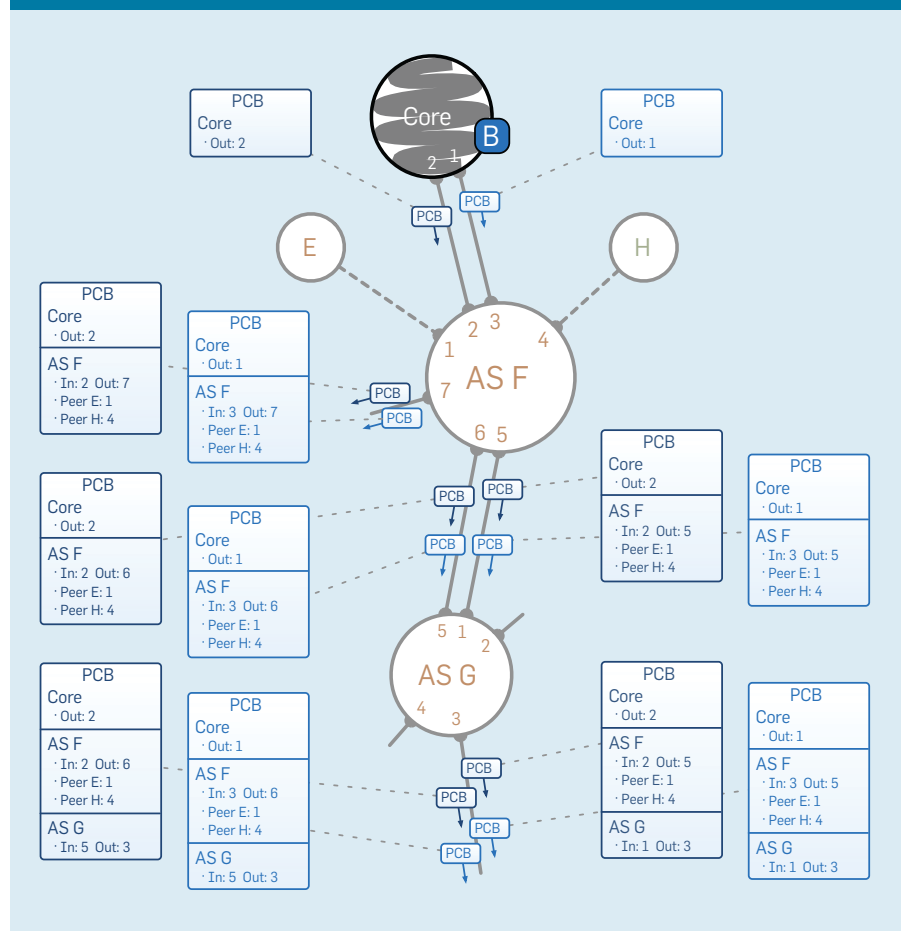
The beacon servers in an AS select the down-segments through which the AS prefers to be reached and register them at the core path servers. When links fail, segments expire or better segments become available, the beacon servers keep updating the down-segments registered for their AS.

Path lookup. To reach a remote destination, a host first queries a SCION name server to obtain the <ISD, AS, end-host address> triplet of the destination. The ISD and AS identifiers are needed to perform a path lookup, and the end-host address is used by the destination AS to deliver the packet to the destination host. To obtain up-segments to reach its ISD core, a host performs a path lookup at its local path server. To obtain down-segments to

reach the destination, the host queries the local path server with the destination's <ISD, AS> tuple. If the local path server has no cached down-segments, it will automatically query the destination AS's core path server.

PCB and path-segment selection. The PCBs to propagate and path segments to register are selected by each AS based on a path-quality metric with the goal of identifying consistent, diverse, efficient, and policy-compliant paths. “Consistency” refers to the requirement that there exists at least one property along which the path is uniform (such as an AS capability like anonymous forwarding) or link property (such as low latency). “Diversity” refers to the set of paths that are announced over time, being as path-disjoint as possible to provide high-quality multipath options. “Efficiency” refers to the length, bandwidth, latency, utilization, and availability of a path, where more-efficient paths are naturally preferred.

Figure 3. Intra-ISD PCB propagation from the ISD core down to customer ASes. For the sake of illustration, the interfaces of each AS are numbered with consecutive integer values. In practice, each AS can choose any encoding for its interfaces; only the AS itself needs to understand its encoding.




“Policy compliance” refers to the requirement that the path adheres to the AS’s routing policy. Based on past PCBs that were sent, a beacon server scores the current set of candidate path segments and sends the k best segments as the next PCB. SCION intra-ISD beaconing can scale to networks of arbitrary size because each inter-AS link carries the same number of PCBs regardless of the number of PCBs received by the AS.

Inter-ISD beaconing is similar to intra-ISD beaconing, except inter-ISD PCBs traverse only ISD core ASes. The same path-selection metrics apply in which an AS attempts to forward the set of most-desirable paths to its neighbors. Like BGP, the process is inherently not scalable, but, as the number of ISDs and the corresponding number of core ASes is small, the approach is viable for SCION.

Link failures. Unlike the current Internet, link failures are not resolved automatically by the network but require active handling by end hosts. Since SCION forwarding paths are static, they break when a link fails. Link failure is handled by a three-pronged approach that typically masks the failure without any outage to the application and rapidly re-establishes fresh working paths like this: Beaconing occurs every few seconds, constantly establishing new working paths; the SCION control message protocol (SCMP), a SCION equivalent of ICMP, is used for link revocation; and SCION end hosts use multipath communication by default, masking link failures to an application with another working path. As multipath communication can increase availability (even in environments with a limited number of paths⁴), SCION beacon servers actively attempt to create disjoint paths and select and announce disjoint paths, and end hosts compose path segments to achieve maximum resilience to path failure. We thus expect most link failures in SCION to go unnoticed by the application, unlike with the numerous short outages in the current Internet.^{16,18}

Intra-AS communication. Communication within ASes is handled through existing intra-domain communication protocols (such as IP, Open Shortest Path First, Multiprotocol Label Switching, and Software-Defined Networking).



We explicitly seek high efficiency such that packet-forwarding latency and throughput are at least as fast as current IP forwarding.




Figure 2b outlines one possible intra-domain path through the magnified AS.

Data plane. While the control plane is responsible for providing end-to-end paths, the data plane ensures packet forwarding using the provided paths. A SCION packet minimally contains a path; source and destination addresses are optional in case the packet’s context is unambiguous without addresses. Consequently, SCION border routers forward packets to the next AS based on the AS-level path in the packet header (augmented with ingress and egress interface identifiers for each AS) without having to inspect the destination address and also without consulting a routing table. Only the border router at the destination AS needs to inspect the destination address or packet purpose to be able to forward it to the appropriate local host(s).

An interesting aspect of forwarding is enabled by the split of “locator” (the path toward the destination AS) and “identifier” (the destination address);¹³ since only the destination AS needs to consider the local identifier, the identifier can have any format the destination can interpret. A domain can thus select an arbitrary addressing format for its hosts (such as a 4B IPv4, 6B medium access control, 16B IPv6, 20B accountable IP, and AIP³). A nice consequence is that an IPv4 host can communicate with an IPv6 host directly through SCION.

Routers forward packets efficiently in the SCION architecture. In particular, absence of inter-domain routing tables and absence of complex longest-prefix matching performed by current routers enable construction of faster, more-energy-efficient routers. During forwarding, a border router first verifies that the packet entered through the correct ingress interface. If the packet has not yet reached the destination AS, the egress interface maps out the next hop.

Path combination. End-to-end communication in SCION is enabled by a combination of up to three path segments that form a SCION forwarding path. After path lookup, and depending on the returned segments, a forwarding path can be created as follows:

- ▶ Immediate combination of path segments (such as $B \rightarrow D$ in Figure 2a), the last AS on the up-segment (ending

at a core AS) is the same AS as the first AS on the down-segment (starting at a core AS). In this case, the simple combination of an up-segment and a down-segment creates a valid forwarding path;

- ▶ Peering shortcut (such as $A \rightarrow B$ in Figure 2a). A peering link exists between the two segments, so a shortcut via the peering link is possible. As in the case of the AS shortcut, the extraneous path segment is cut off; the peering link could also traverse to a different ISD;

- ▶ AS shortcut (such as $B \rightarrow C$ in Figure 2a). The up-segment and down-segment intersect at a non-core AS. In this case, a shorter forwarding path can be created by removing the extraneous part of the path. The special case where the source's up-segment contains the destination AS is treated the same way or the intersection of both segments is omitted from the path;

- ▶ Combination with a core-segment (such as $A \rightarrow D$ in Figure 2a). The last AS on the up-segment is different from the first AS on the down-segment. This case requires an additional core-segment to connect the up- and down-segment. If the communication remains within the same ISD ($A \rightarrow D$), an intra-ISD core-segment is needed; otherwise, an inter-ISD core segment is needed; and

- ▶ On-path (such as $A \rightarrow E$ in Figure 2a). The destination AS is directly on the path to the ISD core, so a single up-segment is sufficient to create a forwarding path.

Once the host chooses a forwarding path, it is encoded in the SCION packet header, making inter-domain routing tables unnecessary for border routers; both the egress and the ingress interface of each AS on the path are encoded as PCFS in the packet header. The destination can respond to the source by inverting the end-to-end path from the packet header or perform its own path lookup and combination.

Security. For protection against malicious entities and provide secure control and data planes, SCION is equipped with an arsenal of security mechanisms.

As in BGPsec,¹⁹ each AS signs the PCB it forwards, enabling PCB validation by all entities. To ensure path correctness, the forwarding information within each PCFS also needs to be cryptographically protected, but signature

verification would hamper efficient forwarding. Each AS thus uses a secret symmetric key that is shared among beacon servers and border routers and used to efficiently compute a message authentication code (MAC) over the forwarding information. The per-AS information includes the ingress and egress interfaces, an expiration time, and the MAC computed over these fields, which are (by default) all encoded within an 8B field we refer to as a “hop field” (HF). The structure of the HF is largely at the discretion of each AS and requires no coordination with any other AS, as long as the AS itself can determine how to forward the packet on to the next AS.

The specified ingress and egress interfaces uniquely identify the links to the previous and following ASes. If, for example, a router is connected via the same outgoing interface to three different neighboring ASes, three different egress-interface identifiers would be assigned by network administrators. The HF's expiration time can be set to the granularity of seconds or hours, depending on path type. For this article, we consider only the common case where paths are long-lived and HFs have an expiration time of approximately 12 hours.

Algorithm agility. In terms of cryptographic mechanisms, SCION includes built-in algorithm agility, meaning cryptographic methods are easily updated and exchanged. The MAC validation of HFs is per-AS, so an AS can independently (without interaction with any other entity) update its keys or cryptographic mechanisms. SCION supports multiple signatures by an AS, meaning an AS can readily deploy a new signature algorithm and start adding those signatures as well. A component of the selection metric favors creating paths where each AS on the path supports the new algorithm.

Authentication. Authentication in SCION is based on digital certificates that bind identifiers to public keys and carry digital signatures that are verified by roots of trust. One notable challenge is how to achieve trust agility to enable flexible selection of trust roots, resilience to private key compromise, and efficient key revocation.²⁰

A central question we have had to address is how to structure the Internet's trust roots. The current Internet follows

two trust models: monopoly and oligopoly. In the monopoly model, a single root of trust is used for authentication. The DNSSEC PK¹⁵ or the Resource Public-Key Infrastructure (RPKI)² used in BGPsec are examples of the monopoly model, as both essentially rely on a single public key that serves as a root of trust to verify all subsequent entities. The monopoly model suffers from two main drawbacks: All parties must agree on a single root of trust, and the single root of trust represents a single point of failure, the misuse of which enables forging a certificate for an arbitrary entity, and its revocation can result in a kill-switch for all its entities. The oligopoly model fares no better; instead of a single root of trust, the oligopoly model relies on several roots of trust, all equally and completely trusted. Instead of a single point of failure in the monopoly model, the oligopoly model thus exposes several points of failure. The prime example is the TLS PKI, featuring approximately 1,500 trusted signing certificates with approximately 300 roots of trust.^{1,12} Attacks reported since 2011 against authorities (such as Comodo, DigiNotar, and GlobalSign) demonstrate how compromise of a single trusted certificate authority enables issuing server certificates for any domain, including those with which there is no business relationship.

SCION allows each ISD to define its own set of trust roots, along with the policy governing their use. Such scoping of trust roots within an ISD greatly improves security, as compromise of a private key associated with a trust root cannot be used to forge a certificate outside the ISD. An ISD's trust roots and policy are encoded in the TRC, which has a version number, a list of public keys that serve as roots of trust for various purposes, and policies governing the number of signatures required for performing different types of actions. The TRC serves as a way to bootstrap all authentications within SCION.

The TRC provides important properties. Trust agility enables users to select trust roots used to initiate certificate validation. Users can thus select an ISD they believe maintains a non-compromised set of trust roots. A challenge with trust agility is how to maintain global verifiability of all entities, regardless of the user's selection. SCION of-

The Future Looks Bright with SCION

The SCION inter-domain network architecture enables new systems that can take advantage of the isolation, scalability, and transparency properties it indeed provides.

Path validation. Through its use of packet-carried forwarding state (PCFS), SCION paves the way for the Origin and Path Trace (OPT) mechanism,¹⁷ enabling senders, receivers, and routers to cryptographically verify the exact path the packets have traversed, with negligible overhead. OPT allows transmission of banking or medical data that is typically bound to strict data-privacy regulations to be constrained to traverse only selected authorized ASes.

Anonymity and privacy. PCFS also provides advantages for privacy. For example, with PCFS and path transparency, the source is able to select paths that appear more trustworthy (such as those that do not traverse certain ASes). In addition, the packet header can be further obfuscated such that ASes on the path cannot learn identifying details about the source or the destination, unless they are immediately connected to one of them. The High-speed Onion Routing at the Network Layer (HORNET)¹⁰ leverages SCION's path-selection infrastructure to deliver high-bandwidth, low-latency anonymous communication.

Highly available communication. Critical infrastructure (such as financial networks and industrial control systems used for power distribution) requires a high degree of availability. Internet outages have been known to disrupt day-to-day operations by, for example, preventing ATM withdrawals or payment terminal operations.²⁷ Numerous such outages are due to the malicious or erroneous announcement of IP address spaces, or “prefix hijacking.” Perhaps the most well-known example is the 2008 hijack of YouTube by Pakistan Telecom for the purpose of censorship, resulting in a global outage of YouTube.⁹ In fact, hijacks affecting only a small portion of the Internet happen on a daily basis. SCION's control-plane isolation through ISDs, its stable data plane, and its multipath operation all contribute to dramatically higher availability. With ISDs, misconfigurations and attacks in one ISD do not affect other ISDs; digitally signed route announcements prevent unauthorized injection of routes; and digitally signed path distribution allows verification of paths by the sender.

DDoS prevention. Bandwidth guarantees are enabled by the Scalable Internet Bandwidth Reservation Architecture (SIBRA),⁶ preventing DDoS attacks at the architectural level; independent of the number of distributed bots, end hosts gain protection against Internet-wide link-flooding attacks, a major threat in the current Internet. SIBRA provides ISDs with dynamic bandwidth guarantees to permanently enable communication. Critical infrastructures can additionally keep some network paths to a destination secret, preventing an adversary from even sending traffic to that destination because the cryptographic HFs are necessary to use a path but are unknown to an adversary.

High-speed Web browsing. Through the SIBRA extension, the sender performs a resource reservation with its initial packet, and the receiver will likely obtain a reservation with a high sending rate it can use immediately on the reverse path. With such a reservation, no congestion control is needed; consequently, Web servers can start sending content immediately at a high rate to the client.

Mobility support. With the ongoing proliferation of mobile devices, supporting reliable communication can be a challenge for any architecture, as these devices frequently connect and disconnect from (sometimes multiple) networks. SCION supports high availability through multipath communication and provides a header extension to inform the other party of new down segments as it connects to a new network. Failing paths are discarded, and new paths are discovered dynamically.

Protection from forged TLS certificates. The government of Iran in 2011 infamously used compromised roots of trust to create rogue TLS certificates for Google and Yahoo services to perform man-in-the-middle attacks on its own citizens. Iran is suspected of having mounted the attack on the DigiNotar certificate authority (CA) that signed these certificates. ISDs and the Attack Resilient Public-Key Infrastructure (ARPKI)⁷ system used in SCION prevent such attacks, as a CA's authority is scoped to the ISDs in which the CA is active. Moreover, in the ARPKI, multiple trusted entities must be compromised to perform a successful man-in-the-middle attack, and revocation of trust roots is possible within a minute, enabling quick recovery from the compromise.

fers this property by requiring all ISDs with a link between them to sign each other's TRCs; as long as a network path exists, a validation path exists along that network path. Efficient revocation of trust roots is the second important property. In the current Internet, trust

roots are revoked manually or through operating system or browser updates, often requiring a week or more before a large fraction of the Internet population has seen the revocations. There is also a long tail of devices and installations that apply revocations very late or

never. In SCION, PCBs carry the version number of the current TRC, and the updated TRC is required to validate that PCB. An AS that realizes it needs a newer TRC can contact the AS from which it has received the PCB. Following distribution of PCBs, an entire ISD updates the TRC within tens of seconds.

SCION Control Message Protocol. The SCMP is similar to ICMP in the current Internet but is authenticated and adapted to SCION. One challenge we have had to address in the design of SCMP is how to enable efficient authentication of SCMP messages, as the naïve approach of adding a digital signature to SCMP messages could create a processing bottleneck at routers when many SCMP messages would be created in response to a link failure. The SCION architecture thus makes use of an efficient symmetric key derivation mechanism called the “Dynamically Re-creatable Key” (DRKey)¹⁷ in which each AS uses a local secret key known to SCION border routers to derive on-the-fly a per-AS secret key using an efficient “pseudorandom function.” Hardware implementations of modern block ciphers enable faster computation than a memory lookup from DRAM, and such dynamic key derivation can thus result in a speedup even over fetching the key from memory. For verification of SCMP messages, the destination AS can fetch the derived key through an additional request message from the originating AS, which is protected by a relatively slow asymmetric operation. However, local caching ensures this key needs to be fetched only infrequently. As a consequence, SCION provides fully secured control messages with minimal overhead.

Deployment

As of April 2017, we had deployed a global SCION testbed we use to vet SCION's functionality and security, including deployment nodes in five continents with four ISDs and 15 ASes, including ISPs—KDDI, Swisscom, and SWITCH—and financial and academic institutions. SCION's open-source code and information for how to deploy a SCION node is available at <http://www.scion-architecture.net/>

Obtaining SCION's full benefits requires a direct connection among multiple ASes. When a direct link is not pos-

sible, remote ASes can be connected via IP tunnels, but their communication depends on the BGP routing protocol. As the testbed expands, we expect more participants will connect directly to benefit from SCION's full feature set.

To use SCION, ISPs at a minimum must deploy a border router capable of “encapsulating” and “decapsulating” SCION traffic as it leaves or enters their networks. SCION ASes must also deploy certificate, beacon, name, and path servers that can run on commodity hardware. Deploying SCION in homes or businesses is designed to require little effort, initially with no changes to existing software or networking stacks or replacement of end-user network devices. This ready connection is achieved through a gateway device that transparently switches communication over to SCION if the remote endpoint is also SCION-enabled. Several companies are currently exploring commercialization of these technologies, notably the startup Anapaya Systems, which offers SCION routers.

Conclusion

SCION is an Internet architecture that provides security, availability, transparency, control, scalability, and more (see the sidebar “The Future Looks Bright with SCION”). SCION offers numerous advantages over the current Internet and supports other future Internet proposals as an underlying building block for highly reliable point-to-point communication.

Despite its research maturity following six years of effort, SCION is still in its infancy in terms of deployment. While requiring relatively small changes by ISPs and domains, broadening adoption is SCION's foremost goal. We expect the benefits for various stakeholders will provide strong incentives for adoption, leading to islands of SCION deployment. In the long term, connections and mergers among islands will enable ever-increasing numbers of native SCION end-to-end connections.

Working on SCION has let us consider Internet architectures from a clean-slate perspective. The absence of limiting constraints (imposed by the current Internet environment) has been particularly rewarding, as the deep exploration of this problem space enables us ask not how a future Internet can achieve what the current Internet has

already achieved, but rather what additional features can and should a future Internet offer. We anticipate the insight into the possible applications of a secure, dynamic, highly available network will help engage the network community to leverage SCION for its applications and contribute to the project.

Our 2017 book *SCION: A Secure Internet Architecture* describes the architecture in more detail, including authentication, name resolution, deployment, operation, extensions, and specifications.²² **C**

References

1. Abadi, M., Birrell, A., Mironov, I., Wobber, T., and Xie, Y. Global authentication in an untrustworthy world. In *Proceedings of the 14th Workshop on Hot Topics in Operating Systems* (Santa Ana Pueblo, NM, May 13–15). Usenix Association, Berkeley, CA, 2013.
2. American Registry for Internet Numbers. Resource Public Key Infrastructure (RPKI); <https://www.arin.net/resources/rpki/>
3. Andersen, D.G., Balakrishnan, H., Feamster, N., Koponen, T., Moon, D., and Shenker, S. Accountable Internet Protocol (AIP). In *Proceedings of ACM SIGCOMM* (Seattle, WA, Aug. 17–22). ACM Press, New York, 2008.
4. Andersen, D.G., Balakrishnan, H., Kaashoek, M.F., and Morris, R. Resilient overlay networks. In *Proceedings of the ACM Symposium on Operating Systems Principles* (Chateau Lake Louise, Banff, Canada, Oct. 21–24). ACM Press, New York, 2001.
5. Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. *DNS Security Introduction and Requirements*. RFC 4033 (Proposed Standard), 2005; <https://www.ietf.org/rfc/rfc4033.txt>
6. Basescu, C., Reischuk, R.M., Szalachowski, P., Perrig, A., Zhang, Y., Hsiao, H.-C., Kubota, A., and Urakawa, J. SIBRA: Scalable Internet Bandwidth Reservation Architecture. In *Proceedings of Network and Distributed System Security Symposium* (San Diego, CA, Feb. 21–24). Internet Society, Reston, VA, 2016.
7. Basin, D., Cremers, C., Kim, T.H.-J., Perrig, A., Sasse, R., and Szalachowski, P. ARPKI: Attack Resilient Public-Key Infrastructure. In *Proceedings of the ACM Conference on Computer and Communications Security* (Scottsdale, AZ, Nov. 3–7). ACM Press, New York, 2014.
8. BBC News. Asia communications hit by quake. Dec. 27, 2006; <http://news.bbc.co.uk/2/hi/asia-pacific/6211451.stm>
9. Brown, M. Pakistan Hijacks YouTube; <http://research.dyn.com/2008/02/pakistan-hijacks-youtube-1/>
10. Chen, C., Asoni, D., Barrera, D., Danezis, G., and Perrig, A. HORNET: High-speed onion routing at the network layer. In *Proceedings of the ACM Conference on Computer and Communications Security* (Denver, CO, Oct. 12–16). ACM Press, New York, 2015.
11. Dübendorfer, T., Wagner, A., and Plattner, B. An economic damage model for large-scale Internet attacks. In *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (University of Modena and Reggio Emilia, Italy, June 14–16). IEEE Press, 2004.
12. Electronic Frontier Foundation. SSL Observatory, 2010; <https://www.eff.org/observatory>
13. Farinacci, D., Fuller, V., Meyer, D., and Lewis, D. *The Locator/ID Separation Protocol (LISP)*. RFC 6830, 2013; <https://tools.ietf.org/html/rfc6830>
14. Han, D., Anand, A., Dogar, F., Li, B., Lim, H., Machado, M., Mukundan, A., Wu, W., Akella, A., Andersen, D.G., Byers, J.W., Seshan, S., and Steenkiste, P. XIA: Efficient support for evolvable internetworking. In *Proceedings of the Ninth USENIX Symposium on Networked Systems Design and Implementation* (San Jose, CA, Apr. 25–27). USENIX Association, Berkeley, CA, 2012.
15. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., and Braynard, R.L. Networking named content. In *Proceedings of the Fifth International Conference on Emerging Networking Experiments and Technologies* (Rome, Italy, Dec. 1–4). ACM Press, New York, 2009.
16. Katz-Bassett, E., Scott, C., Chones, D., Cunha, I.,

- Valancius, V., Feamster, N., Madhyastha, H., Anderson, T., and Krishnamurthy, A. LIFEGUARD: Practical repair of persistent route failures. In *Proceedings of ACM SIGCOMM* (Helsinki, Finland, Aug. 13–17). ACM Press, New York, 2012.
17. Kim, T. H., Basescu, C., Jia, L., Lee, S.B., Hu, Y., and Perrig, A. Lightweight source authentication and path validation. In *Proceedings of ACM SIGCOMM* (Chicago, IL, Aug. 17–22). ACM Press, New York, 2014.
 18. Kushman, N., Kandula, S., and Katabi, D. Can you hear me now? It must be BGP. *ACM SIGCOMM Computer Communication Review* 37, 2 (Apr. 2007), 75–84.
 19. Lepinski, M. and Turner, S. *An Overview of BGPsec*. IETF draft, May 8, 2012; <http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-overview-02>
 20. Matsumoto, S., Reischuk, R.M., Szalachowski, P., Kim, T.H.-J., and Perrig, A. Authentication challenges in a global environment. *ACM Transactions on Privacy and Security* 20, 1 (Feb. 2017), 1–34.
 21. Palo Alto Research Center. The CCNx Project (Content-Centric Networking); <http://blogs.parc.com/ccnx/>
 22. Perrig, A., Szalachowski, P., Reischuk, R.M., and Chuat, L. *SCION: A Secure Internet Architecture*. Springer, Berlin, Germany, 2017.
 23. Raychaudhuri, D., Nagaraja, K., and Venkataramani, A. MobilityFirst: A robust and trustworthy mobility-centric architecture for the future Internet. *ACM SIGMOBILE Mobile Computing and Communications Review* 16, 3 (July 2012), 2–13.
 24. Sahoo, A., Kant, K., and Mohapatra, P. BGP convergence delay under large-scale failures: Characterization and solutions. *Computer Communications* 32, 7 (May 2009), 1207–1218.
 25. Saltzer, J.H., Reed, D.P., and Clark, D.D. End-to-end arguments in system design. *ACM Transactions on Computer Systems* 2, 4 (Nov. 1984), 277–288.
 26. Schuchard, M., Vasserman, E.Y., Mohaisen, A., Kune, D.F., Hopper, N., and Kim, Y. Losing control of the Internet: Using the data plane to attack the control plane. In *Proceedings of the Network and Distributed System Security Symposium* (San Diego, CA, Feb. 6–9). Internet Society, Reston, VA, 2011.
 27. Toonk, A. Massive route leak causes Internet slowdown. BGPmon, June 12, 2015; <http://www.bgpmon.net/massive-route-leak-cause-internet-slowdown/>
 28. Zhang, X., Hsiao, H.-C., Hasker, G., Chan, H., Perrig, A., and Andersen, D.G. SCION: Scalability, control, and isolation on next-generation networks. In *Proceedings of IEEE Symposium on Security and Privacy* (Oakland, CA, May 22–25). IEEE Press, 2011.

David Barrera (david.barrera@inf.ethz.ch) is a postdoc in the Network Security Group at ETH Zürich in Switzerland.

Laurent Chuat (laurent.chuat@inf.ethz.ch) is a Ph.D. student in the Network Security Group at ETH Zürich in Switzerland.

Adrian Perrig (aperrig@inf.ethz.ch) is a professor in the Department of Computer Science and leads the Network Security Group at ETH Zürich in Switzerland and an ACM Fellow.

Raphael M. Reischuk (reischuk@inf.ethz.ch) is a senior IT-security researcher at ETH Zürich in Switzerland focusing on network and Web security.

Pawel Szalachowski (psz@inf.ethz.ch) is a senior researcher in the Network Security Group at ETH Zürich in Switzerland.

DOI:10.1145/3081884

Conflict is averted by separating work and family time and responsibility, as reflected in millions of tweets.

BY YILI LIU AND LINA ZHOU

The Dynamics of Work-Family Conflict

WORK-FAMILY CONFLICT (WFC) is an important line of inquiry in organizational behavior and human resource management research. The topic is relevant to the computing and communication field not only because modern communication technologies allow for more integration of work and family roles than ever before¹⁵ but because recent advances in computing technology offer new ways to respond to and understand WFC. WFC has been empirically associated with employees' job and life dissatisfaction, poor physical and psychological health, and rising voluntary turnover rates and work stress.⁵ WFC has also been found to have a negative correlation with various aspects of organizations, including performance, commitment, psychological contract, and even strategy planning.²⁰ Our analysis of American Community Survey and Census data²¹ from 2015 found over 75.2% of males and over 60.5% of females in married couples have their own earnings,

with dual-income families emerging as the predominant family structure in the U.S.

Considerable effort has gone toward trying to understand the antecedents and role of WFC. Research shows individual characteristics and experience influence perception of WFC,⁶ with two significant implications for the dynamics of WFC: Different individuals may respond to the same WFC differently, and individuals may react to the same WFC differently over time through their attempts to cope with WFC and their changing situations. However, not enough research has considered the dynamics of WFC, especially regarding individual differences. In addition, no one fully understands the relationships between WFC and job and family satisfaction sufficiently due to inconsistent findings about their relationships across different studies.^{14,17} Moreover, emerging social media could be reshaping the dynamics of WFC and remains unexplored.

Our study began to fill these gaps in the literature by recognizing the great potential of social-media data to help social scientists, as well as business managers, discover the dynamics of WFC and advance understanding of the relationships between WFC and job and family satisfaction in modern society. In addition, the social-media analytics approach we used has significant methodological implications for WFC research. First, work-family (WF)

» key insights

- **Work-family conflict can potentially undermine employees' feelings of satisfaction for job and family life.**
- **Our study analyzes streams of social-media data collected from more than 93,000 users, overcoming many limitations of survey and diary methods that dominate traditional research in work-family conflict.**
- **The findings highlight the dynamics of time-based and strain-based conflict, helping explain mixed findings about the relationship between work-family conflict and job and family satisfaction generally found in the literature.**



PHOTO BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

research has been criticized for being overly reliant on cross-sectional designs and self-report survey data at the individual level of analysis.⁷ In contrast, diary methods are promising for examining WF phenomena over time or to increase the field's understanding of dynamic WF relations.⁷ Users of social media, particularly Twitter, write brief time-stamped text updates (such as tweets) about their lives on the go, ranging from daily activity to current events, news stories, and their interests in real time. The ability to observe and analyze high-volume, continuous streams of sample data as they are being generated can effectively support the study of WFC dynamics. Second, much WF research has been conducted with a homogeneous population, but the studies

“must begin to use large heterogeneous populations.”¹⁴ Twitter, as a social-media platform, had approximately 320 million active users worldwide per month as of September 2015. The large scale and diversity of the Twitter-user population reflects variations in individual and organization settings that may influence WFC and satisfaction relationships. Third, the most common survey method suffers from recall error due to misinterpretation of survey questions, frailty of memory, close-categories bias, and lack of intrinsic motivation.¹⁶ In contrast, self-recorded information about individual lives at the moment it happens in tweets can effectively alleviate recall error. We thus innovatively adopted tweets as the lens to examine WFC in our study.

Work and Family

Work and family are two important domains for all people. WFC is experienced when there is conflict between pressure in either domain.¹² WFC can be classified into time- and strain-based categories, along with others.¹² Specifically, the time devoted to and the strain produced by work make it difficult to fulfill requirements of family and vice versa. Past research has shown that working adults explicitly experience mood spillover across their work and family lives; the stress in these two areas also increases with the amount of spillover.¹³ On the other hand, individuals can learn to live with WFC as their experience and response tactics improve, in turn lowering their stress.⁵ Additionally, family

Figure 1. WFC trends by day of the week.

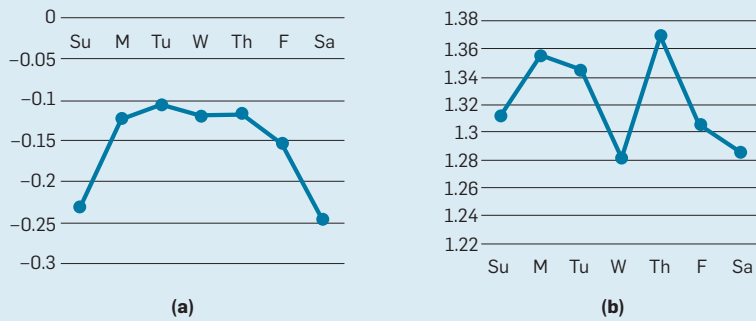
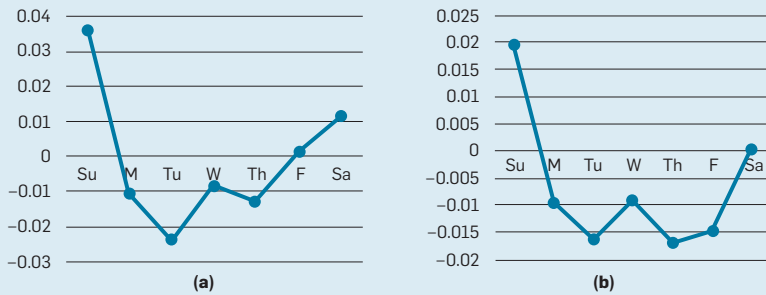


Figure 2. Satisfaction trends by day of the week.



and work situations fluctuate depending on one’s circumstances, and a person may reply differently at different times.²⁰ However, the dynamics of WFC were not addressed in previous studies because they generally adopted one-time, cross-sectional measure with few exceptions. For instance, one study published in 2013 collected survey data at two different points in time—2004 and 2006—to investigate the relationship between WFC and pay satisfaction.¹ Although a diary method was used to examine WFC and work-family facilitation,³ finding considerable variation in the same individuals, the traditional diary method is difficult to scale up in terms of number of participants. Further, despite that previous studies established negative correlations between WFC and job and family satisfaction, the strength of relationships varied greatly from one study to another, ranging from nearly negligible to strong.^{14,17} This variability raises the need to explain the inconsistent findings.

To address these limitations, we propose a social-media analytics approach to investigating the dynamics

of WFC, particularly time-based and strain-based WFC, and the relationships between WFC and job and family satisfaction through the lens of tweets for the first time. Twitter provides abundant data where user opinions on certain topics or events can be mined and is expected to present a precise picture of dynamics and influence of WFC and related user experience and perception. Tweets have been used to examine the changing patterns of diurnal and seasonal mood with work, sleep, and length of day,¹¹ but they have yet to be explored to help understand the dynamics of WFC. Our research thus takes a significant step toward expanding research methods for examining WFC.

Social-Media Analytics Method

We used a dataset of Twitter users and their tweets collected through a combination of random sampling and social sub-graph extraction that was representative of the actual population of the U.S.⁸ To study WFC, we filtered those users who did not have a job based on whether none of their tweets involved work-related topics. We also filtered those users who posted a to-

tal of 10 tweets or fewer to assemble a sufficient base for understanding how they experienced WFC. The final sample, as of February 2010, included 3,327,715 tweets from 93,700 users, with 35 tweets per user on average over six months in 2009 and 2010.

To account for time-zone differences among Twitter users, we converted the timestamps on all tweets to users’ local time. Our identification of topics in tweets employed Linguistic Inquiry and Word Count, a tool widely used for text analysis,¹⁹ providing typical word dictionaries that measure psychological (such as social) processes, and personal concerns. Personal concerns consist of sub-categories (such as work and home), and social processes consist of sub-categories (such as family). We classified tweets with words in the work category as job-related and tweets with words in the family and home sub-categories as family-related.

We used two sets of variables: WFC and satisfaction. We measured all variables first at the individual level by date and then aggregated them over all active users for each date. To account for individual differences, we measured the baselines of all individuals and standardized all measures by removing baseline differences across individuals.

Following previous conceptualizations,^{4,12} we measured WFC along two dimensions: time and strain. Time-based WFC (TC) is a consequence of competition for an individual’s time from work and family responsibilities. One classical example is “The time I spend with my family (work) often causes me not to spend time in activities at work (family).”⁴ Strain-based WFC (SC) arises when role stressors at work (or with family) induce strain in the individual, hampering fulfillment of role expectations in the family (or work) domain. Two classical examples are “I am often so emotionally drained when I get home from work that it prevents me from contributing to my family” and “Because I am often stressed from family responsibilities, I have a hard time concentrating on my work.”⁴

TC. We used the proportion of tweets on work- and family-related topics posted by a user on a given day as a proxy for the time the user reported

spending in the respective domain. TC arose when an individual allocated above-average time for work and simultaneously below-average time for family and vice versa. Accordingly, we defined TC as the average difference between the time allocation for work and for family.

SC. Strain can be deduced from psychological and physical dimensions,¹² though it is difficult, if not impossible, to measure physical strain directly from social-media data. We thus measured SC from the psychological aspect of participant data, specifically through negative mood (such as anxiety, anger, and sadness). Negative mood in one domain is associated with pessimism and rumination, causing individuals to neglect requirements in the other domain.¹⁸ SC arose when an individual simultaneously experienced above-average negative mood in both work and family. Accordingly, we defined SC as the average negative mood across work and family.

We measured satisfaction in two domains—work and family—we labeled as job satisfaction (JS) and family satisfaction (FS), respectively. Satisfaction can be explained as a causal sequence linking mood to performance and reward in a domain.¹⁸ Positive mood can facilitate role performance by enhancing cognitive functioning, increasing task activity and persistence, and promoting positive interactions with others. Meanwhile, intrinsic and extrinsic rewards earned through role performance can thus enhance positive mood.¹⁸ We viewed positive mood as a proxy measure of satisfaction.

JS (or FS). We defined satisfaction with work (or family) as the proportion of work- or family-related tweets expressed in positive moods.

Results and Discussion

To understand the dynamics of WFC, we first analyzed the trends of time-based WFC (TC) and strain-based WFC (SC) at different levels of time granularity after removing the data associated with American national holidays (such as Thanksgiving, Christmas, and New Year’s). The results show the trends of WFC at levels of month and week are both relatively stable. Nevertheless, WFC fluctuates by day of the week (see Figure 1).

Figure 1a shows TC is heightened during weekdays relative to weekends. This observation confirms previous findings on work-family time allocation that both men and women in the U.S. spend more time in domestic work and caring for children on a weekend day than on a weekday.¹⁰ In addition, TC shows an increasing trend from Sundays to Tuesdays and the opposite trend from Thursdays to Saturdays, peaking on Tuesdays and dipping on Saturdays. There is thus a sharp increase in TC when transitioning from weekends to weekdays and a sudden drop during the opposite transition. Note TC stays in the negative range, indicating family-to-work spillover events are more prevalent than work-to-family spillover.

Figure 1b shows SC is much greater on weekdays than weekends, except for Wednesdays. If the trend profile of TC resembles a normal distribution, then the trend profile of SC simulates a bimodal distribution, with Mondays and Thursdays being two peaks. Unlike TC, which peaks on Tuesdays, SC reaches a peak immediately after weekdays begin. The abrupt drop in SC on Wednesdays could be explained by people reaching a state of work-family balance by the middle of the week. A sharply elevated SC on Thursdays may be attributed to the pressure of trying to complete scheduled weekly tasks, onset of work exhaustion, or preparation for upcoming family commitments.

Overall, the results show both TC and SC vary markedly with the day of the week, an observation that helps explain the inconsistent findings about the form and intensity of WFC in the literature. They also imply a major limitation of survey-based methods that have dominated traditional WFC research. Depending on the day a survey is administered, participants’ responses can vary greatly.

Figure 2 shows the trends for job satisfaction (JS) and family satisfaction (FS), including that individuals’ JS and FS are both higher on weekends than on weekdays. People feel satisfied when their criteria are met, and satisfaction can be viewed as a degree of realized expectation. On weekdays, the expectation of work performance is high, and, consequently, the possibil-

ity of unfulfilled expectation may also be high, negatively influencing JS. In comparison, the expectation is lower on weekend days, indirectly contributing to a higher level of JS.

JS reaches its highest level on Fridays, as in Figure 2a, a finding that could be attributed to actual fulfillment of weekly work expectations. At the other end, JS is at its lowest on Tuesdays, followed by an increasing trend the rest of the week. One possible explanation is work stress peaks on Tuesdays when the weekend-to-weekday transition is over and work expectations begin to mount.

FS peaks on Wednesdays, as in Figure 2b, a finding that could be attributed to people’s state of work-family balance, as manifested in both low SC and relatively low TC, as in Figure 1. There is a sharp drop in FS from Sundays through Tuesdays, as expected. Surprisingly, FS barely makes it into the positive range on Saturdays. One possible explanation is that employed women shift child care and housework to weekend days.⁹ Given individuals’ high expectation of family commitments on weekends, it is difficult for them to feel highly satisfied, particularly as weekends begin to unfold. In comparison, FS is relatively more stable than JS, as reflected in a smaller variance across different days, as in Figure 2a and Figure 2b.

To understand the association between WFC (TC and SC) and satisfaction (JS and FS) we performed pairwise correlation analyses between the two sets of variables. The results reported in the accompanying table reflect that all correlation coefficients are negative. In addition, the correlations between SC and JS/FS, but not between TC and JS/FS, are significant ($p < .001$).

The findings on the relationship between TC and satisfaction challenge current mainstream thinking that WFC has a negative influence on

Results of correlation analysis.

| | JS | FS |
|----|-----------|-----------|
| TC | -0.056 | -0.108 |
| SC | -0.430*** | -0.316*** |

*** $p < 0.001$

job and life satisfaction.¹⁵ Employees could adapt themselves to WFC as their experience and responding tactics improve through empowerment of modern communication technologies. For instance, telecommunicating could support employees in performing some or all of their job functions outside the workplace, even as they stay connected to work during non-work hours.² As the boundary between family and work activities blurs in some situations, and despite the distinct norms and requirements of the two roles,¹² TC inevitably loses its influence on JS and FS.

Our findings on SC emphasize the importance of employees' psychological well being, with significant managerial implications for human resource management in organizations. Minimizing employees' distress, anxiety, fear, anger, and disgust is thus instrumental to boosting their JS and FS. For example, an increasing spillover between work and family activities might contribute to both increased SC and decreased JS and FS. Managers looking to control employees' stress try to avoid assigning work activities for non-work hours. And, while enjoying a flexible work-family arrangement, employees are able to reduce their psychological strain by minimizing interference of family responsibilities in non-family situations and non-family hours.

These results also show that both JS and FS are subject to the influence of the same type of WFC, highlighting the importance of WFC in our lives and significant spillover between our personal experience of work and family.

Conclusion

WFC garners widespread attention in modern society beyond human resources management. Despite extensive research in this area, different studies report inconsistent and even contradictory findings on the effects and intensity of WFC. Additionally, the overlap in time and place between traditional family and work roles may also introduce new opportunities for WFC to manifest in people's everyday lives.

Ours is the first study to investigate the dynamics of WFC and explain mixed findings concerning the relationships between WFC and satisfac-

tion in the literature. Our results show WFC is markedly higher on weekdays than weekends and, more important, fluctuates on weekdays. Our comparison of the two types of WFC shows the dynamics of strain-based conflict is more pronounced than time-based conflict on weekdays. Interestingly, we found strain-based conflict reaches its lowest point on Wednesdays. We also found the relationships between WFC and job and family satisfaction, suggesting that, while people may adapt to the inherent conflict between work and family activity due to the flexibility of work place and time, they also feel dissatisfaction from connecting with work during non-work hours.

The social-media analytics we employed address the limitations of survey methods dominating traditional WFC studies. High-volume, high-velocity Twitter data provides a dynamic, fine-grain view of individuals' behavior in a naturally occurring setting that serves as an ideal testbed for understanding WFC.

This research can be improved and continue in several directions. Tracking a larger number of Twitter users over a longer period of time would improve the general applicability of the findings. In addition, some jobs have distinctive busy and off-peak seasons. In view of country differences in work-family time¹⁰ and workweek,¹¹ WFC outside the U.S. deserves its own separate investigations. Alternative techniques should be explored to improve extracting work- and family-related topics and mood from social-media data. And resolving multiple online identities for the same Twitter users can help refine the findings of our study.

Acknowledgments

We thank the reviewers for their constructive comments. Portions of this work are supported by the National Science Foundation under Award Number SES-152768. The research reported here is that of the authors and does not reflect the official policy of the National Science Foundation. C

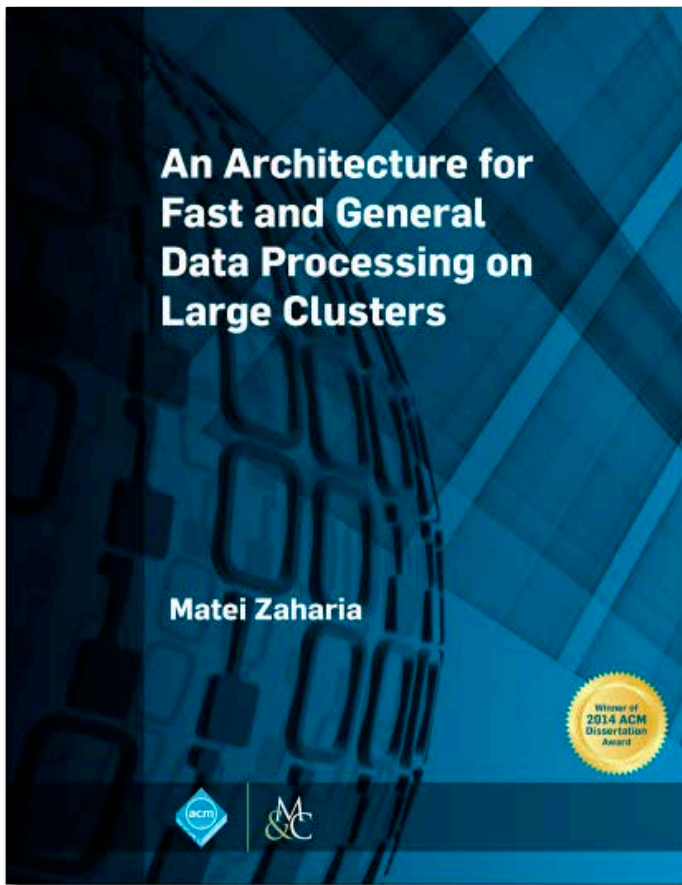
References

1. Bhava, D.P., Kramer, A., and Glomb, T.M. Pay satisfaction and work-family conflict across time. *Journal of Organizational Behavior* 34, 5 (July 2013), 698–713.
2. Boswell, W.R. and Olton-Buchanan, J.B. The use of communication technologies after hours: The role

- of work attitudes and work-life conflict. *Journal of Management* 33, 4 (Aug. 2007), 592–610.
3. Butten, A.B., Grzywacz, J.G., Bass, B.L., and Linney, K.D. Extending the demands-control model: A daily diary study of job characteristics, work-family conflict, and work-family facilitation. *Journal of Occupational and Organizational Psychology* 78, 2 (June 2005), 155–169.
4. Carlson, D.S., Kacmar, K.M., and Williams, L.J. Construction and initial validation of a multidimensional measure of work-family conflict. *Journal of Vocational Behavior* 56, 2 (Apr. 2000), 249–276.
5. Carlson, D.S. and Perrewé, P.L. The role of social support in the stressor-strain relationship: An examination of work-family conflict. *Journal of Management* 25, 4 (Aug. 1, 1999), 513–540.
6. Carr, J.C., Boyar, S.L., and Gregory, B.T. The moderating effect of work-family centrality on work-family conflict, organizational attitudes, and turnover behavior. *Journal of Management* 34, 2 (Apr. 2008), 244–262.
7. Casper, W.J., Eby, L.T., Bordeaux, C., Lockwood, A., and Lambert, D. A review of research methods in IO/OB work-family research. *Journal of Applied Psychology* 92, 1 (Jan. 2007), 28–43.
8. Cheng, Z., Caverlee, J., and Lee, K. You are where you tweet: A content-based approach to geolocating Twitter users. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (Toronto, ON, Canada, Oct. 26–30). ACM Press, New York, 2010, 759–768.
9. Craig, L. How employed mothers in Australia find time for both market work and childcare. *Journal of Family and Economic Issues* 28, 1 (Mar. 2007), 69–87.
10. Craig, L. and Mullan, K. Parenthood, gender, and work-family time in the United States, Australia, Italy, France, and Denmark. *Journal of Marriage and Family* 72, 5 (Oct. 2010), 1344–1361.
11. Golder, S.A. and Macy, M.W. Diurnal and seasonal mood vary with work, sleep, and day length across diverse cultures. *Science* 333, 6051 (Sept. 30, 2011), 1878–1881.
12. Greenhaus, J.H. and Beutell, N.J. Sources of conflict between work and family roles. *The Academy of Management Review* 10, 1 (Jan. 1985), 76–88.
13. Grzywacz, J.G., Almeida, D.M., and McDonald, D.A. Work-family spillover and daily reports of work and family stress in the adult labor force. *Family Relations* 51, 1 (Jan. 2002), 28–36.
14. Kossek, E.E. and Ozeki, C. Work-family conflict, policies, and the job-life satisfaction relationship: A review and directions for organizational behavior-human resources research. *Journal of Applied Psychology* 83, 2 (Apr. 1998), 139–149.
15. Maertz, C.P. and Boyar, S.L. Work-family conflict, enrichment, and balance under 'levels' and 'episodes' approaches. *Journal of Management* 37, 1 (Jan. 2011), 68–98.
16. Marsden, P.V. and Wright, J.D., Eds. *Handbook of Survey Research*. Emerald Group Publishing Ltd., Howard House, U.K., 2010.
17. Michel, J.S., Kotrba, L.M., Mitchelson, J.K., Clark, M.A., and Baltes, B.B. Antecedents of work-family conflict: A meta-analytic review. *Journal of Organizational Behavior* 32, 5 (July 2011), 689–725.
18. Nolen-Hoeksema, S., Parker, L.E., and Larson, J. Ruminative coping with depressed mood following loss. *Journal of Personality and Social Psychology* 67, 1 (July 1994), 92–104.
19. Tausczik, Y.R. and Pennebaker, J.W. The psychological meaning of words: LIWC and computerized text-analysis methods. *Journal of Language and Social Psychology* 29, 1 (Mar. 2010), 24–54.
20. Taylor, B.L., DelCampo, R.G., and Blanco, D.M. Work-family conflict/facilitation and the role of workplace supports for U.S. Hispanic professionals. *Journal of Organizational Behavior* 30, 5 (July 2009), 643–664.
21. U.S. Census Bureau. *America's Families and Living Arrangements: 2015: Adults (A Table Series)*; <https://www.census.gov/hhes/families/data/cps2015A.html>

Yili Liu (liuyili@tongji.edu.cn) is an assistant professor in the Department of Management Science and Engineering at Tongji University, Shanghai, China.

Lina Zhou (zhoul@umbc.edu) is an associate professor in the Department of Information Systems at University of Maryland, Baltimore County, Baltimore, MD.



Proposing an architecture for cluster computing systems that can tackle emerging data processing workloads at scale.

This book, a revised version of the 2014 ACM Dissertation Award winning dissertation, proposes an architecture for cluster computing systems that can tackle emerging data processing workloads at scale. Whereas early cluster computing systems, like MapReduce, handled batch processing, our architecture also enables streaming and interactive queries, while keeping MapReduce's scalability and fault tolerance. And whereas most deployed systems only support simple one-pass computations (e.g., SQL queries), ours also extends to the multi-pass algorithms required for complex analytics like machine learning. Finally, unlike the specialized systems proposed for some of these workloads, our architecture allows these computations to be combined, enabling rich new applications that intermix, for example, streaming and batch processing.

We achieve these results through a simple extension to MapReduce that adds primitives for data sharing, called Resilient Distributed Datasets (RDDs). We show that this is enough to capture a wide range of workloads. We implement RDDs in the open source Spark system, which we evaluate using synthetic and real workloads. Spark matches or exceeds the performance of specialized systems in many domains, while offering stronger fault tolerance properties and allowing these workloads to be combined. Finally, we examine the generality of RDDs from both a theoretical modeling perspective and a systems perspective.

This version of the dissertation makes corrections throughout the text and adds a new section on the evolution of Apache Spark in industry since 2014. In addition, editing, formatting, and links for the references have been added.



<http://books.acm.org>

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



DOI:10.1145/3015455

New blocks frameworks open doors to greater experimentation for novices and professionals alike.

BY DAVID BAU, JEFF GRAY, CAITLIN KELLEHER, JOSH SHELDON, AND FRANKLYN TURBAK

Learnable Programming: Blocks and Beyond

A GLOBAL PUSH to broaden participation in computer science has led to an explosion of interest in blocks-based programming. Visual blocks are used by numerous programming tools (see the sidebar). Millions of students receive their first exposure to programming via these tools in courses and activities like Code.org's *Hour of Code*. Blocks allow beginners to compose programs without struggling with the frustrations of syntax (Figure 1).

There is increasing interest in developing and studying blocks languages. At VL/HCC 2015, a small workshop session called Blocks and Beyond^a ballooned to a large event, with 51 submissions and 36 presenters. Researchers shared work in new blocks languages,

^a <http://cs.wellesley.edu/blocks-and-beyond>

interface innovations, domain-specific applications of blocks, and ways to make blocks languages more effective and accessible for diverse coders.

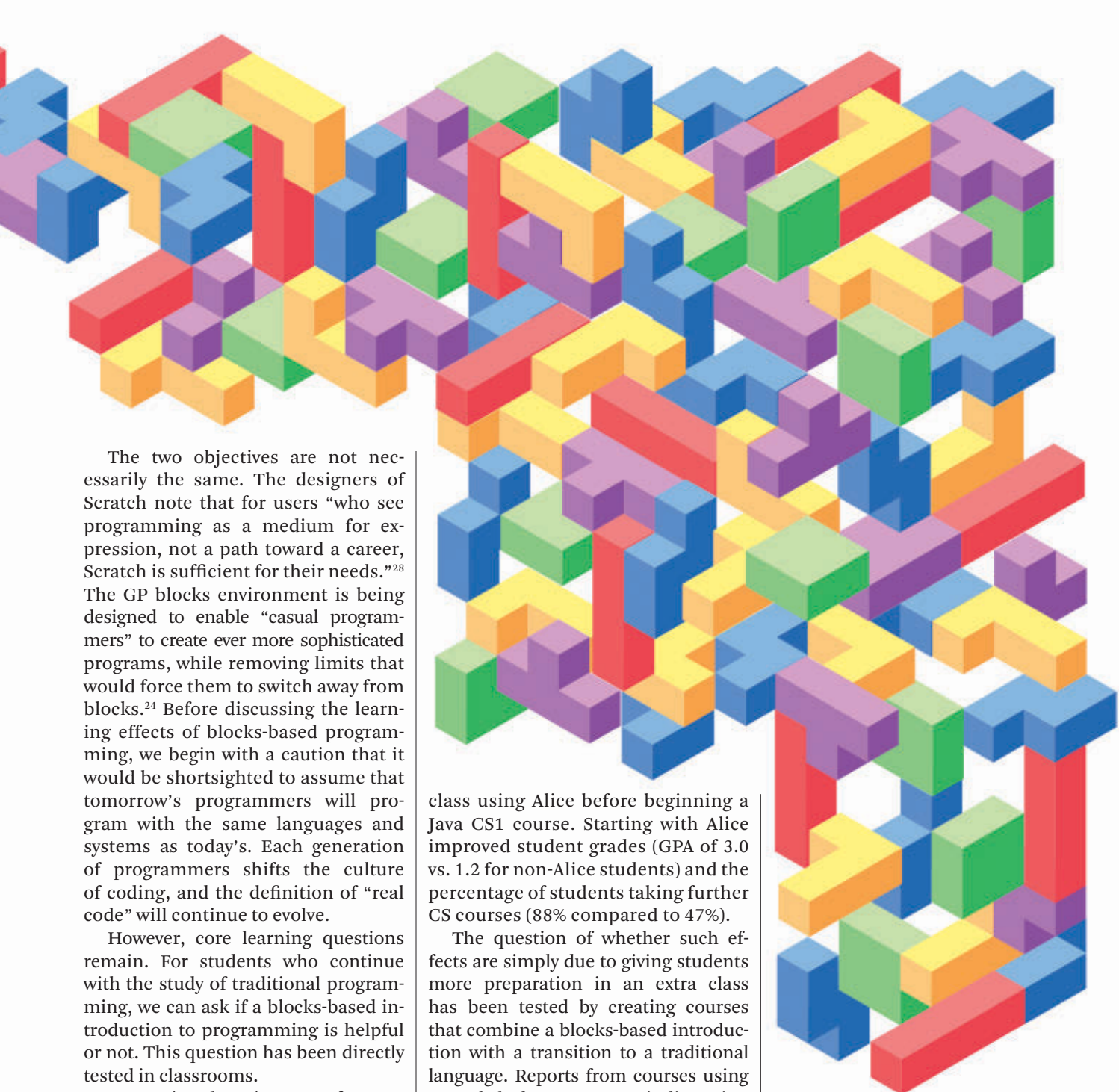
This article explores how blocks impact the learnability of programming. We begin by reviewing studies on the effectiveness of blocks languages. Then we discuss the key features of blocks languages and how they relate to learning. Finally, we look at applications of blocks in new domains and discuss tools for creating your own blocks language.

Watching beginners create their first programs with blocks can be simultaneously inspiring and unsettling. Empowered by blocks, novices will rapidly build complex, often-delightful creations. But just as quickly, they fill their screen with clumsy and intricate code.²² A seasoned programmer inspecting a beginner's disordered assembly might worry that snapping together colorful blocks has nothing to do with "real code." But what is "real code," and why learn it?

What is "real code?" The purpose of a blocks-based tool is to make programming easy to learn. But programming education can have two distinct endpoints: Development of expertise to support professional programming, and the ability to accomplish other goals by creating programs.

» key insights

- **Blocks programming environments have emerged as a popular way to introduce coding and as a stepping stone to traditional text-based languages, but they can also be used to write "real code."**
- **Blocks environments improve learnability for novices by favoring recognition over recall; reducing cognitive load by chunking computational patterns into blocks; and using direct manipulation of blocks to prevent errors and enhance understanding of program structure.**
- **Learnability is also enhanced by key features beyond blocks, including online environments, high-level abstractions, visible state, and easy-to-find examples.**
- **Tookits are available to enable you to enhance your own block language.**



The two objectives are not necessarily the same. The designers of Scratch note that for users “who see programming as a medium for expression, not a path toward a career, Scratch is sufficient for their needs.”²⁸ The GP blocks environment is being designed to enable “casual programmers” to create ever more sophisticated programs, while removing limits that would force them to switch away from blocks.²⁴ Before discussing the learning effects of blocks-based programming, we begin with a caution that it would be shortsighted to assume that tomorrow’s programmers will program with the same languages and systems as today’s. Each generation of programmers shifts the culture of coding, and the definition of “real code” will continue to evolve.

However, core learning questions remain. For students who continue with the study of traditional programming, we can ask if a blocks-based introduction to programming is helpful or not. This question has been directly tested in classrooms.

Measuring learning transfer: Research indicates that learning a blocks language can improve later learning of a traditional textual language. In a study of 10th graders learning C# or Java,¹ those who had taken a Scratch course in 9th grade learned more quickly, understood loops better, and were more engaged and confident than their peers who had not. However, in the final test, a significant difference was seen in only one of three cognitive dimensions. In a study at two colleges,²⁵ students with little or no previous programming experience and weak math preparation completed a CS0 programming

class using Alice before beginning a Java CS1 course. Starting with Alice improved student grades (GPA of 3.0 vs. 1.2 for non-Alice students) and the percentage of students taking further CS courses (88% compared to 47%).

The question of whether such effects are simply due to giving students more preparation in an extra class has been tested by creating courses that combine a blocks-based introduction with a transition to a traditional language. Reports from courses using Scratch before Java or C indicate improved student engagement and understanding of some concepts.^{19,39} In one study focused on learning transfer,⁸ introductory Java course at CMU was modified to begin with *Alice*. Students in this class that used both languages averaged 10% or more better performance on every section of the same Java final exam, including expression evaluation, control structures, arrays, and working with class definitions.

That result is remarkable because one might assume that spending more time programming with blocks meant less time to learn Java. The study used a version of Alice that generated Java

code from Alice blocks, and a mediated transfer pedagogy that made explicit connections between programming concepts in Alice and Java.

Other studies of CS1 courses that switch from blocks to text without these features have identified potential challenges to learning with blocks.^{10,27} Switching from a blocks language to text can involve both a change in syntax and semantics, and Shapiro and Ahrens propose teaching the transitions separately, by introducing syntax before generalizing semantics.³² Additional research is needed to identify the circumstances under which blocks are effective.

Figure 1. Snap! is an example of a blocks-based programming environment. Users drag blocks from a palette of programming elements (left) into workspace (center), where blocks can be assembled into programs. Snap! also provides an output window (top-right) and a sprite picker (bottom-right).



Today, many introductory computer science courses use a blocks-before-text approach. In Harvard's CS50, students move from Scratch to C; Berkeley's CS10 progresses from Snap! to Python; Project Lead The Way's Computer Science Principles (CSP) course uses both Scratch and App Inventor before moving on to Python; and Code.org's CSP App Lab course moves from Droplet blocks to JavaScript.

Why Blocks Are Learnable

In 2004, Ko, Myers, and Aung¹⁵ identified six learning barriers encountered by non-programmers in programming tasks. Three of these—*selection*, *use*, and *coordination*—reflect the difficulty of simply assembling a program. We believe the learnability of blocks languages arises from how they address the usability challenges underlying these three learning barriers:

1. Learning a programming vocabulary is difficult. Blocks simplify this problem because picking a block from a palette is far easier than remembering a word: blocks rely on *recognition instead of recall*.

2. Code is difficult to use because it presents a high cognitive load for new programmers. Blocks reduce the cognitive load by *chunking code* into a smaller number of meaningful elements.

3. Assembling code is error-prone. Blocks help users assemble code without basic errors by providing *con-*

strained direct manipulation of structure (for example, two incompatible concepts do not have connecting parts).

Recognition versus recall. Programming with a simple language or library typically involves a vocabulary of about 100–200 words. For example, HTML has 100 tags and 100 attributes, and SQL has about 200 keywords; Scratch is similar, with 130 blocks. Recalling 100–200 concepts can overwhelm a newcomer.

Unlike text languages, blocks languages are intimately tied to their programming environments, and nearly all block environments have adopted a few interface conventions that address key usability problems. One such convention is tackling vocabulary by organizing blocks in functionally related palettes on the screen.

Palettes differ from autocomplete menus in professional code editors because they persist instead of disappearing and they organize concepts by topic instead of by name. This design simplifies discovery and exploration. Figure 2a shows the Sound palette in Scratch. It is an instructive reference showing all 13 methods for audio in that environment. Similar organization is seen in larger blocks environments. To help manage the complexity of creating mobile apps, App Inventor provides a dynamic set of blocks, with additional blocks available in programs that have interactions with more components

(Figure 2b). Yet, the basic idiom is the same as Scratch: explorable palettes organized by function.

Remembering the order, type, and valid values of operands is also daunting for newcomers. Many block languages address this by supplying blocks with default operand values, drop-down menus and specialized editors to specify operands, and extra words to indicate operand meanings (Figure 3a).

Chunking information with blocks. Programming languages present a high cognitive load to a student who is learning a new syntax. For example, consider the `for` loop in JavaScript syntax:

```
for (var i = 0; i < 50; i++) { ___ }
```

This dense notation is a barrier to beginners. In the words of one student, JavaScript “is really confusing to understand with all the parentheses and brackets and all of that.”³⁸

To understand the difficulty, consider that this code contains five words (`for` `var` `i` `i` `i`), 10 pieces of punctuation (`(` `=` `;` `<` `;` `++` `{` `}` `)`), and two numbers (0 and 50), a total of 17 units of information. Studies of human cognitive capacity have established that people have a working memory of about seven chunks of information.²³ Trying to understand this line of code as 17 separate items may overwhelm the working memory of a new programmer.

Experienced JavaScript programmers have no problem understanding the line of code noted here because they have learned to interpret the code in larger chunks. Because a `for` loop follows a very common pattern, it can be read in just two chunks: first, the typical `for` loop that uses the conventional looping pattern (`i` starting at 0 and incrementing by 1); second, the particular choice of 50 as the upper limit. Figure 4 illustrates different ways of chunking the code.

Blocks help reduce cognitive load by showing new programmers how to read larger chunks. In the Code.org Computer Science Principles course, blocks for JavaScript `for` loops are drawn just as an expert would see the code: as two chunks with a single block with a single socket for the loop upper bound. Complexity can also be reduced by nesting chunks within chunks. For ex-

ample, Code.org reveals finer-grained for structure in more advanced portions of the same course, as illustrated at the bottom of Figure 4.

The use of blocks to chunk code aids readability even for simple commands, because blocks can forgo the punctuation that text code uses to denote structure and use explanatory words instead. For example, as illustrated in Figure 3b, a simple call in Python requires reading delimiters and knowing argument order, whereas the equivalent block in Scratch reads naturally, using appropriate abstractions (for example, a piano keyboard).

By organizing code as visible chunks, blocks help new programmers concentrate on what the code means rather than the notation that is used to write it.

Direct manipulation of visible structure. The visual form of blocks alleviates the burden of assembling syntactically correct units by typing one character at a time. But there are other advantages of directly manipulating program fragments that have visual constraints.

One benefit is the blocks can help prevent errors by making the grammar of the program visible. Blocks can be seen as a form of syntax-directed editing with constrained direct manipulation. In 1981, creators of an early structure-editing tool noted, “Programs are not text; they are hierarchical compositions of computational structures and should be edited, executed, and debugged in an environment that consistently acknowledges and reinforces this viewpoint.”³⁵

Block shapes help beginners understand which grammatical phrases (expressions vs. commands vs. declarations) are legal in what contexts. In Scratch, commands connect vertically with nubs and notches, whereas expressions are smooth shapes that fit into smooth holes. Constraints on drag-and-drop prevent the two types from being confused (Figure 5). Students report that the puzzle shapes are helpful for assembling programs.³⁸

Visualization of types by shape can be applied to richer type systems: OpenBlocks provides 14 connector shapes to represent different types,²⁹ and researchers have created experimental block languages with dynamically generated shapes to represent compositional type systems.^{18,36}

Figure 2. The sound palette in Scratch (a) and a voice synthesis palette in App Inventor (b). Palettes simplify the selection of programming elements by exploiting the ease of *recognition over recall*. Palettes organize concepts by topic, not name, and they remain open when used, allowing the user to discover and tinker with blocks based on their function.

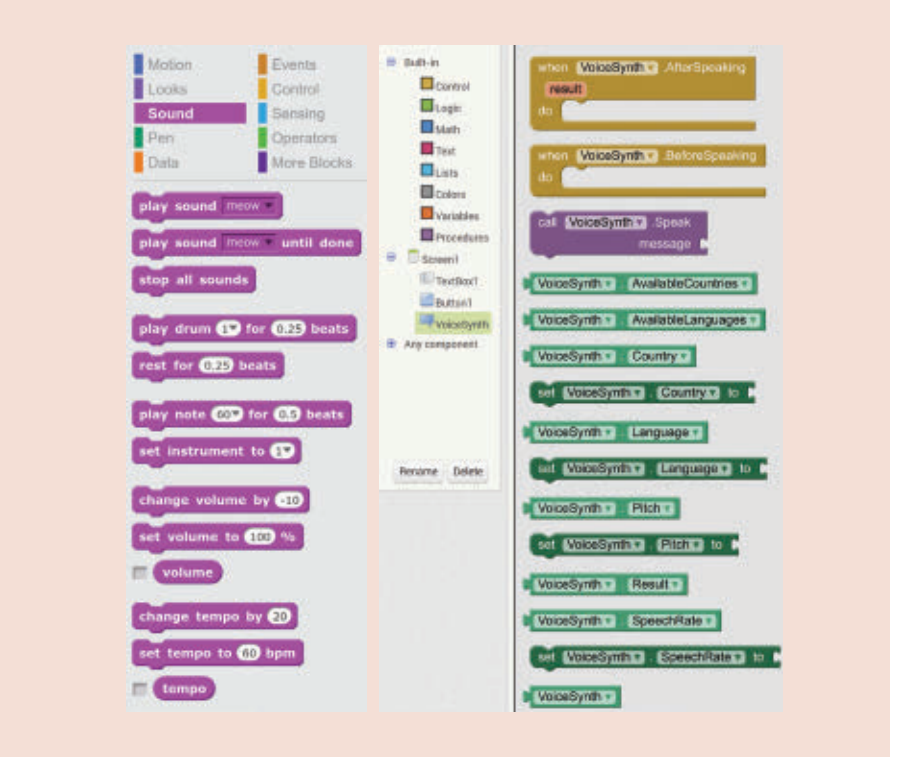


Figure 3. Blocks show structure visually (a. Scratch) instead of using punctuation. They can aid learnability using plain language, default values, and value pickers.

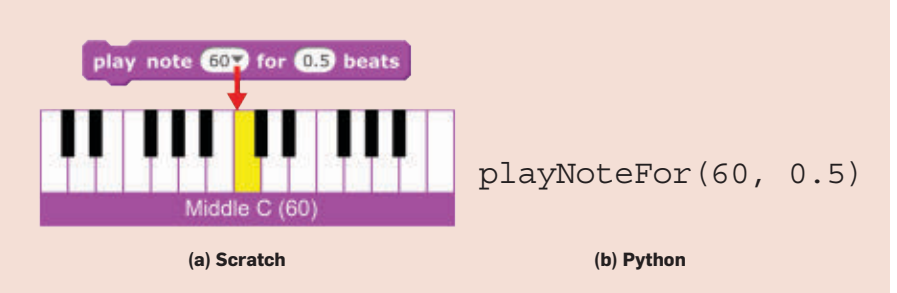
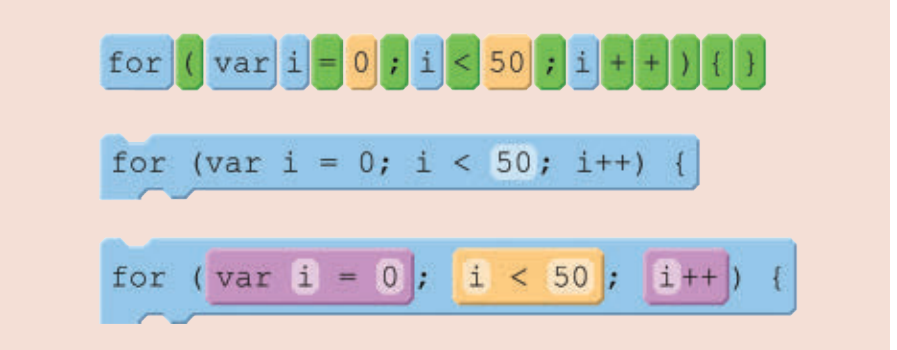


Figure 4. Three ways of reading a for loop in chunks. A naive reading of code (top) interprets the code as 17 chunks, but an expert reading of code (middle) interprets the most common form of loop as a single chunk, with a second chunk for the loop limit 50. An alternative (bottom) reading interprets three clauses as chunks. Code.org uses the middle rendering when introducing loops to high-school students for the first time, and switches to the bottom rendering when students are familiar with for loops.



Directly manipulable blocks also encourage bottom-up tinkering with program pieces in ways not directly supported by raw text. Blocks programmers experiment with blocks by connecting them to build islands of code fragments on the programming surface that are isolated from the main program.^{22,38} In blocks environments supporting liveness,²⁰ these fragments can be executed by pointing and clicking, providing a key benefit of interpreted text-based languages without a read-eval-print loop console separate from the editor. The program gradually grows as it is augmented by dropping in these fragments when they behave as desired.

Learnability Beyond Blocks. Blocks aid in the construction of code, but blocks alone are not enough to make a programming language learnable. Users new to a language face additional learning challenges:


- ▶ They must wrestle with practical aspects like installing language tools, saving/loading programs, and so on;
- ▶ They must learn the vocabulary of the language and understand the concepts denoted by its words;
- ▶ They need to understand runtime semantics such as flow of control and changes in state over time; and,
- ▶ They eventually need to learn common patterns of use, moving beyond isolated concepts.

Each of these learning hurdles can be helped or hindered by the programming environment, and each of these problems is an area of active research and development.


Programming online. To simplify installation, programming tools are moving online. When a programming environment is in a Web browser, a new programmer is just a few clicks away from creating a first program. A cloud-based programming tool can provide a complete and consistent programming environment with fewer potential problems.

Although blocks programming environments for beginners have long been offered online, text-based programming environments are also becoming available online. With tools such as Cloud 9, CodeAnywhere, and CodeEnvy, programmers of all levels can benefit from working online.

Words, concepts, and abstractions. The names chosen for language con-



By organizing code as visible chunks, blocks help new programmers concentrate on what the code means rather than the notation that is used to write it.



structs can have an impact on learnability. Empirical studies by Stefik and Seibert³⁴ have found that common keywords like `for` or operators such as `!=` serve as hard-to-learn jargon, and are not as easily learnable as familiar words such as `repeat` or `unequal`. They found the syntax of languages such as Java and Perl are no more learnable than a synthetic programming language with randomly selected punctuation used for keywords. Even more important for learnability are the abstractions chosen by language designers to allow users to build simple programs with compelling behavior. For example, the designers of Alice worked with users to develop intuitive abstractions for controlling 3D animations. During their design process, the Alice team eliminated jargon such as transformation matrices and substituted more intuitive concepts like object-relative motions. These new abstractions made it easier for users to specify 3D animations.⁷

Designing languages and libraries focused on learnability based on empirical evidence is a major area for future work.

Runtime understanding. The dynamic state of a program can be made more understandable by making its state visible. For example, Code.org highlights the individual block actively running so the correspondence between code and action can be seen. Snap! provides widgets for every variable to show the current state.

Even with highly visible state, understanding actions in the past or future can be difficult. Liveness²⁰ is one approach to addressing this problem. A live system aims to make actions concrete by applying them immediately to the current state. For example, in App Inventor, Scratch, and Snap!, many edits to a running block program take effect immediately, without the need to restart the program.

Another approach for making the evolution of state understandable is to allow the programmer to travel in time by inspecting, advancing or rewinding the timeline of a program. The concept of omniscient debugging was first described in the early 1970s as a capability to trace backward in time through execution history to identify the loca-

tion of the fault that caused a later observed failure.⁴⁰ TRAKLA2,²⁶ UUhistle³³ and Online Python Tutor¹² provide this capability for Java and Python.

Examples and reuse. The activity of programming has changed with the availability of large repositories containing examples of shared code. Programmers of all levels report finding and adapting examples as a core programming activity.^{6,9} In response, professional programming and end-user programming environments are beginning to incorporate example support.^{5,31} Novices, too, want to learn via examples, but may struggle to do so.³⁰

Blocks-based languages like Scratch and Looking Glass use online sharing and remixing programs to provide example access. But there is a trade-off between the simplicity of reuse and the robustness of reused code. For example, Scratch simplifies sharing of code examples for novices by providing a “backpack” for collecting program snippets and assets that can be shared and dragged into a new project. However, the backpack does not necessarily guarantee the code will run correctly in a new project.

In contrast, Looking Glass uses a more complex process for reuse in which users select the beginning and end of behavior they want to use. Coupled with execution history information, this can ensure the selected code will function within the context of a new program. A Play & Explore feature allows users to connect program output to the line or lines of code that caused it, helping users to understand and begin to modify reused code.

Scaling Blocks Code

Why don't professionals program with block interfaces? One reason is that direct manipulation has efficiency disadvantages when making small edits. When creating an expression such as $(a/2 + b/2)$ in blocks, the programmer must find and drag blocks for each of the three arithmetic operators, and then fill in holes with variables and numbers. Similarly, when rearranging an expression from $(a/2 + b/2)$ to $(a+b)/2$, the expression tree must be pulled apart and put together again, requiring more gestures and more forethought than making the edits in text. HCI researchers observed that visual programming

languages can have a higher *viscosity* than text code because they make small changes more difficult.¹¹

Beyond viscosity, blocks environments can have several other usability disadvantages compared to textual programming languages:

- ▶ Low density: Blocks take more space on the screen than equivalent text code.

- ▶ Search and navigation: It can be challenging to find and navigate to the relevant part of a blocks program in a 2D workspace, only part of which may be visible.

- ▶ Source control: collaboration and version control systems are difficult to use without a text representation.

The newest generation of blocks programming tools includes features designed to resolve the tension between usability advantages of text vs. blocks. There are two approaches:

text-style entry and bidirectional mode switching.

Text-style entry of blocks. Some new blocks environments, such as Greenfoot's frame-based Stride editor¹⁶ and GP,²⁴ are designed to be used by programmers to create large programs, so efficient editing is an important design goal.

Both Stride and GP improve efficiency by providing text-based editing shortcuts within a blocks-oriented interface. To allow users to circumvent the step of finding a block on the palette, these systems let programmers enter blocks through an in-line auto-complete mechanism. Blocks can still be chosen from a palette, but a knowledgeable programmer can insert them by typing. The Stride editor also introduces a hybrid approach to editing code, differentiating between low-level and high-level structure (Figure 6). For

Figure 5. Block shapes show and enforce rules composition. Scratch commands compose vertically, and expressions fit into holes. Here, a Boolean expression (diamond shape) is being dropped into a matching hole for a loop test condition.

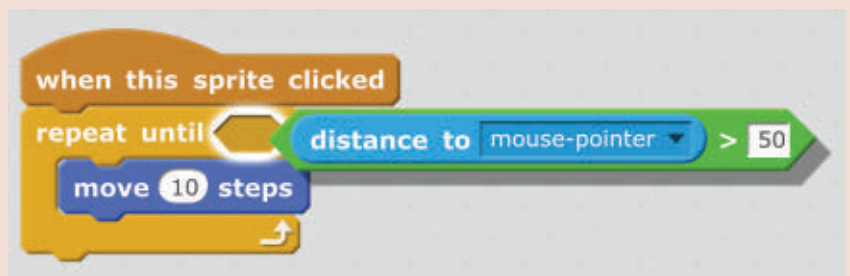


Figure 6. Greenfoot's Stride editor combines text-style editing for expression-level details with drag-and-drop blocks for higher-level program structure.

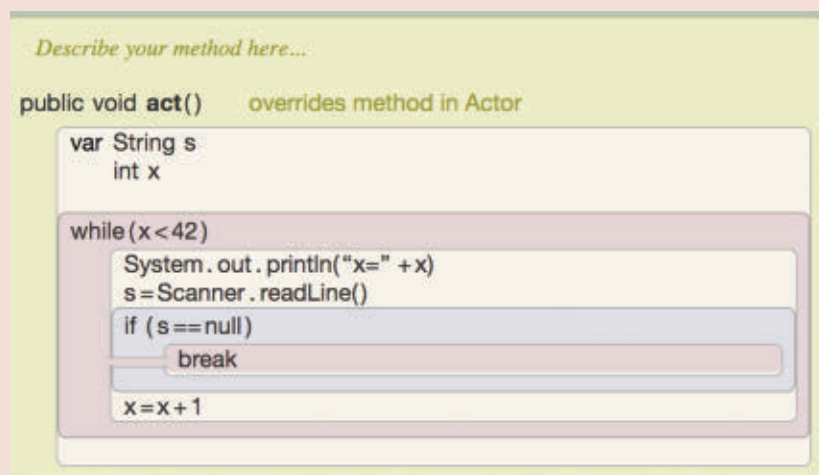


Figure 7. Pencil Code provides bidirectional switching between blocks and text. Mode switching allows users to learn with blocks and edit quickly with text.

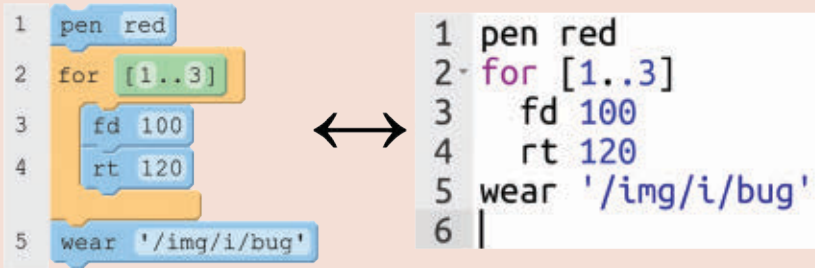


Figure 8. Blocks programming in MadeUp. 3D printing is an area of rapid innovation, and blocks make it possible to use new 3D modeling languages without a steep learning curve.

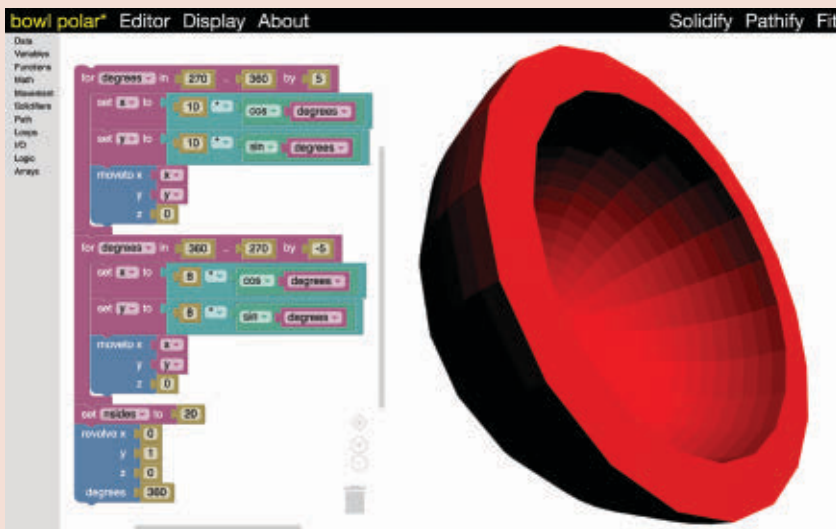
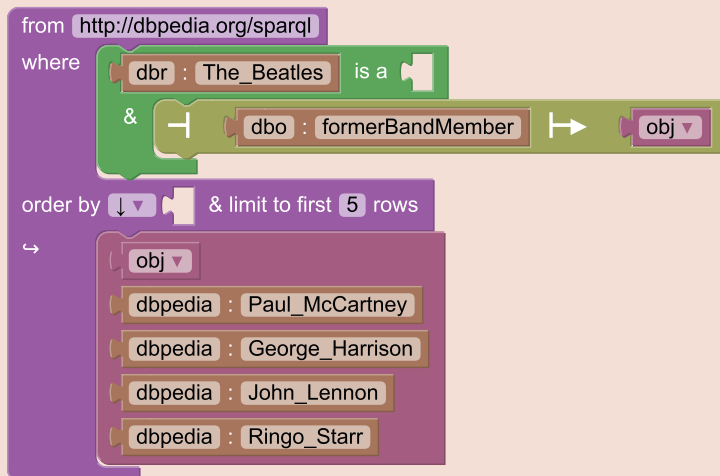


Figure 9. The SPARQL playground is a blocks-based query execution tool that provides blocks for constructing queries of RDF data. Query results (bottom) are also provided as blocks, and they can be dragged to build into other queries.



expression-level code, it hides syntactic structure and allows traditional text editing, providing high-density display and lower viscosity. Visible tree structure and drag-and-drop manipulation are used for higher-level code such as control flow and class declarations.

Bidirectional Mode Switching. Some blocks environments provide a bidirectional transformation between a traditional text language and a blocks representation of that language. These include Pencil Code (CoffeeScript)² (Figure 7), Code.org’s App Lab^b (JavaScript), BlockEditor (Java),²¹ and Tiled Grace (Grace).¹³ Alice and Blockly provide non-editable views of text code.

The hypothesis that motivates the design of dual-mode tools is that users may benefit from the learnability of blocks in one mode, while they learn syntax and get the efficiency of text in the other mode. This goal requires the views be linked. For text to be safe for users who may want to return blocks, it must be possible to switch between modes.

In dual-mode editors, the text code is the primary representation of the program, and blocks are a projected user interface view derived by parsing. This approach allows the editor to fully represent text information such as spacing, but it also means the editor must allow syntax errors that would not have been possible in blocks. Error-recovery heuristics convert simple text syntax errors to special error blocks, but complex errors can prevent mode switching.

Comparing approaches. There is a trade-off between the two approaches to unifying blocks and text. While the dual-mode editors provide direct support for learning traditional text syntax such as JavaScript or Java, they also impose the cognitive overhead of working with syntax errors that can only be introduced in text mode. Visualization research on multiple coordinated views suggests the benefits of providing more than one view need to be balanced against the cognitive overhead imposed by switching between views.³⁷

Single mode structure editors have the advantage of a conceptual model free of syntax errors, because the primary object being edited is the abstract syntax tree. However, to main-

^b <https://code.org/educate/applab>

tain consistency, many kinds of textual edits must be prevented, and other edits may require special tree-editing commands. These constraints raise editing viscosity, and they may present additional cognitive barriers. Interfaces that effectively bridge the gap between blocks and text are an active area of research.

Applying Blocks: Two Examples

Let us take a look at experimental blocks languages in two specific domains that are unfamiliar to most programmers.

Programming 3D printers. Traditionally, 3D printing models are drawn interactively by direct manipulation using CAD software. Writing custom code to create a model is a powerful alternative approach, but coding for 3D fabrication has traditionally been the province of a few expert programmers. Now, the falling cost and rising availability of 3D printers has made it possible for non-specialists to write their own custom 3D modeling code.

Two recently developed languages bring programming for 3D printing to novice users: BeetleBlocks¹⁷ and MadeUp.¹⁴ Although the two systems have different languages, there are several commonalities: both are Web-based interfaces with a live 3D rendering of the shape being created, and both provide a blocks language to simplify learning. Beetle-Blocks follows the principles of turtle graphics: a beetle moves a “pen” that can be turned on and off, and 3D shapes can be created out of iterated strokes. MadeUp (Figure 8) takes a more abstract approach, allowing users to trace out both paths and parametric surfaces. Special functions can rotate or extrude the paths to create solids.

The two languages offer different levels of power and abstraction. Which language is the right one to use? This domain is an excellent example of the advantage of the learnability of blocks. Both blocks languages have a very shallow learning curve, and it is easy to try both.

Querying the Semantic Web. The frequent need to query large datasets is another domain where learnability is essential. Consider the problem of querying Resource Description Framework (RDF) data from the Semantic Web. SPARQL is the standard language

for querying RDF; it includes several constructs for working with RDF triples that distinguish it from other query languages such as SQL. However, potential users of SPARQL face two hurdles: First, programmers must learn the vocabulary and syntax of SPARQL, with its specialized operators and constructs. Second, querying RDF requires not just knowledge of the language, but also knowledge of instance and schema data.

To address both problems, Paolo Bottoni and Miguel Ceriani have created a blocks language they call the *SPARQL Playground*.⁴ Rather than helping users learn about sequential programs, their blocks language helps users select, filter, and join data using SPARQL primitives.

The SPARQL Playground is interesting for a second reason: all query results in the playground are also returned as draggable blocks (Figure 9). This feature allows users to save instance data on the programming workspace to be incorporated into new queries. With the SPARQL Playground, it is easy to begin with general queries to explore the types of data available, and then use these discoveries to make refinements.

Making New Blocks Languages

It is now possible to create your own domain-specific blocks environment using a blocks-based language toolkit. Blocks language authors should be aware of at least three toolkits: Blockly,^c Droplet,³ and OpenBlocks.²⁹ The original blocks metalanguage is MIT OpenBlocks. Created in 2007 by Ricarose Roque as the basis of StarLogo TNG, it allows a large degree of geometric customization. OpenBlocks has also been used in App Inventor Classic and BlockEditor. One drawback of OpenBlocks is that it requires users to download and install the Java JDK. The challenge of installation is addressed by Blockly, an HTML-based block language toolkit by Neil Fraser of Google. Blockly is currently the most popular blocks language toolkit: it is the tool behind App Inventor, the SPARQL playground, and MadeUp, as well as the Code.org *Hour of Code* puzzles. Future versions of Scratch will also use Blockly. Droplet is the newest of the blocks lan-

^c <https://developers.google.com/blockly>

Blocks-Based Educational Tools

On the Web:

- ▶ Scratch (<https://scratch.mit.edu>): In-browser animation and game creation, with support for extensions
- ▶ Code.org (<http://code.org>): A variety of tools including puzzle programming exercises with tutorial videos
- ▶ Snap! (<http://snap.berkeley.edu>): Enhanced language inspired by Scratch that includes first class functions
- ▶ App Inventor (<http://appinventor.mit.edu>): Creation of Android apps using in-browser blocks IDE
- ▶ Pencil Code (<https://pencilcode.net>): Creates Coffee-Script Web apps, transforming between text and blocks
- ▶ StarLogo Nova (<http://www.slnova.org>): Multi-agent simulations and games in a 3D rendered world
- ▶ Blockly Games (<https://blockly-games.appspot.com/>): A set of puzzles to solve with programming blocks
- ▶ GameBlox (<https://gameblox.org>): Game creation that includes clonable agents, physics, and more

Downloadable:

- ▶ AgentSheets / AgentCubes (<http://www.agentsheets.com>): Pioneering blocks environments for creating rule-based games and simulations
- ▶ Alice (<http://www.alice.org>): Pioneering blocks environment for creating 3D virtual worlds; support for exporting to Java
- ▶ Looking Glass (<https://lookingglass.wustl.edu>): 3D animated story creation; supports independent learning
- ▶ Kodu (<http://www.kodugamelab.com>): Rule-based programming of games for Xbox and PC

On mobile:

- ▶ Scratch Jr (<http://www.scratchjr.org>): Programming of animated scenes aimed at preliterate children
- ▶ Pocket Code (<http://www.catrobat.org>): Blocks programming for small form factor of mobile devices
- ▶ Tynker (<https://www.tynker.com>): Polished commercial platform for game and animation creation
- ▶ Hopscotch (<https://www.gethopscotch.com>): Creating games and animations on iPhone and iPad

guage creation toolkits, developed by Anthony Bau for use in Pencil Code, and also used by Code.org in their App Lab. It is newer and less mature than Open-Blocks and Blockly, but takes a unique approach that allows seamless bidirectional transformation between all blocks and textual code.

Summary

When a programming language is provided as a user interface that welcomes novice users, rather than as a technical tool only for experienced developers, we arrive at a new picture of what the programming environment should provide:

- ▶ Vocabulary should derive from recognition, not recall;
- ▶ Cognitive load should be lowered by chunking code;
- ▶ Grammar rules and types should be made visible;
- ▶ Program chunks should be directly manipulable;
- ▶ Low-viscosity editing should also be possible;
- ▶ Coding environments should be available without the need for tool installation;
- ▶ Simple concepts should be described with clear words and high-level abstractions;
- ▶ Runtime state and behavior should be visible; and,
- ▶ Examples should be easy to find and apply.

In short, for a programming tool to be usable by new or casual programmers, its design must focus on learnability. Blocks have proven to be effective at solving many of these problems. Although programming is still not nearly as widely learned as it should be, the progress made by blocks language interfaces can inspire us all to see that programming can be made more learnable. The art of programming is the original human-computer interaction, and it remains an unsolved usability challenge. We can still do more to make programming available to all. □

References

1. Armoni, M., Meerbaum-Salant, O. and Ben-Ari, M. From Scratch to 'real' programming. *Trans. Computing Education* 14, 4 (Feb. 2015).
2. Bau, D., Bau, D.A., Dawson, M., and Pickens, C.S. Pencil Code: Block code for a text world. In *14th International Conference on Interaction Design and Children*, (2015), 445–448.
3. Bau, D.A. Droplet, a blocks-based editor for text code. *J. Computing Sciences in Colleges* 30, 6 (June 2015), 138–144.
4. Bottoni, P. and Ceriani, M. Using blocks to get more

- blocks: Exploring linked data through integration of queries and result sets in block programming. In *IEEE Blocks and Beyond Workshop*, Oct. 2015, 99–102.
5. Brandt, J., Dontcheva, M., Weskamp, M. and Klemmer, S.R. Example-centric programming: Integrating Web search into the development environment. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*, 2010, 513–522.
 6. Brandt, J., Guo, P.J., Lewenstein, J., Dontcheva, M., and Klemmer, S.R. Two studies of opportunistic programming: Interleaving Web foraging, learning, and writing code. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*, 2009, 1589–1598.
 7. Conway, M., Audia, S., Burnette, T., Cosgrove, D. and Christiansen, K. Alice: Lessons learned from building a 3D system for novices. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*, 2000, 486–493.
 8. Dann, W., Cosgrove, D., Slater, D., Culyba, D. and Cooper, S. Mediated transfer: Alice 3 to Java. In *43rd ACM Technical Symposium on Computer Science Education*, 2012, 141–146.
 9. Dorn, B. and Guzdial, M. Graphic designers who program as informal computer science learners. In *Proceedings of 2nd International Workshop on Computing Education Research*, 2006, 127–134.
 10. Garlick, R. and Cankaya, E.C. Using Alice in CS1: A quantitative experiment. In *15th Annual Conference on Innovation and Technology in Computer Science Education*, 2010, 165–168.
 11. Green, T.R.G. Cognitive dimensions of notations. *People and Computers V*. A. Sutcliffe and L. Macaulay, Eds. Cambridge University Press, Cambridge, UK, 1989, 443–460.
 12. Guo, P.J. Online Python tutor: Embeddable Web-based program visualization for CS education. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 2013, 579–584.
 13. Homer, M. and Noble, J. Combining tiled and textual views of code. In *Proceedings of the 2014 IEEE Working Conf. Software Visualization*, Sept. 2014, 1–10.
 14. Johnson, C. and Bui, P. Blocks in, blocks out: A language for 3D models. *IEEE Blocks and Beyond Workshop*, Oct. 2015, 77–82.
 15. Ko, A.J., Myers, B.A. and Aung, H.H. Six learning barriers in end-user programming systems. In *Proceedings of the IEEE Symp. Visual Languages and Human Centric Computing*, 2004, 199–206.
 16. Kölling, M., Brown, N.C.C., and Altadmri, A. Frame-based editing: Easing the transition from blocks to text-based programming. In *Proceedings of the 10th Workshop in Primary and Secondary Computing Education*, Nov. 2015.
 17. Koschitz, D. and Rosenbaum, E. Exploring algorithmic geometry with 'Beetle Blocks': A graphical programming language for generating 3D forms. In *Proceedings of the 15th International Conference on Geometry and Graphics*, Aug. 2012, 380–389.
 18. Lerner, S., Foster, S.R., and Griswold W.G. Polymorphic blocks: Formalism-inspired UI for structured connectors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, 3063–3072.
 19. Malan, D.J. and Leitner, H.H. Scratch for budding computer scientists. *ACM SIGCSE Bulletin* 39, 1 (Mar. 2007), 223–227.
 20. Maloney, J.H. and Smith, R.B. Directness and liveness in the morphic user interface construction environment. In *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, 1995, 21–28.
 21. Matsuzawa, Y., Ohata, T., Sugiura, M. and Sakai, S. Language migration in non-CS introductory programming through mutual language translation environment. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, 185–190.
 22. Meerbaum-Salant, O., Armoni, M. and Ben-Ari, M. Habits of programming in Scratch. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, 2011, 168–172.
 23. Miller, G.A. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review* 63, 2 (1956), 81.
 24. Mönig, J., Ohshima, Y. and Maloney, J. Blocks at your fingertips: Blurring the line between blocks and text in GP. *IEEE Blocks and Beyond Workshop*, 2015, 51–53.
 25. Moskal, B., Lurie, D. and Cooper, S. Evaluating the effectiveness of a new instructional approach. *ACM SIGCSE Bulletin* 36, 1 (2004), 75–79.
 26. Nikander, J., Korhonen, A., Seppälä, O., Karavirta, V.,

- Silvasti, P. and Malmi, L. Visual algorithm simulation exercise system with automatic assessment: Trakla2. *Info. Education—An International J.* 32 (2004), 267–288.
27. Powers, K., Eocott, S. and Hirshfield, L.M. Through the looking glass: Teaching CS0 with Alice. *ACM SIGCSE Bulletin* 39, 1 (Mar. 2007), 213–217.
 28. Resnick, M., Malone, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J. and Silverman, B. Scratch: Programming for all. *Commun. ACM* 52, 11 (Nov. 2009), 60–67.
 29. Roque, R. Openblocks: An extendable framework for graphical block programming systems. Master's thesis, MIT, May 2007.
 30. Rosson, M.B., Ballin, J. and Nash, H. Everyday programming: Challenges and opportunities for informal Web development. In *Proceedings of the 2004 IEEE Symposium on Visual Languages and Human Centric Computing*, Sept. 2004, 123–130.
 31. Sawadsky, N. and Murphy, G.C. Fishtail: From task context to source code examples. In *Proceedings of the 1st Workshop on Developing Tools As Plug-ins*, 2011, 48–51.
 32. Shapiro, R.B. and Ahrens, M. Beyond blocks: Syntax and semantics. *Communications of the ACM* 59, 5 (May 2016), 39–41.
 33. Sorva, J. and Sirkkiä, T. Context-sensitive guidance in the UUIstle program visualization system. In *Proceedings of the 6th Program Visualization Workshop*, 2011, 77–85.
 34. Stefik, A. and Siebert, S. An empirical investigation into programming language syntax. *Trans. Computing Education* 13, 4 (Nov. 2013), 19:1–19:40.
 35. Teitelbaum, T. and Reps, T. The Cornell program synthesizer: A syntax-directed programming environment. *Commun. ACM* 24, 9 (Sept. 1981), 563–573.
 36. Vasek, M. Representing expressive types in blocks programming languages. Undergraduate thesis, Wellesley College, May 2012.
 37. Wang Baldonado M.Q., Woodruff, A., and Kuchinsky, A. Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, 2000, 110–119.
 38. Weintrop, D. and Wilensky, U. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*, 2015, 199–208.
 39. Wolz, U., Leitner, H.H., Malan, D.J., and Maloney J. Starting with Scratch in CS1. *ACM SIGCSE Bulletin* 41, 1 (Mar. 2009), 2–3.
 40. Zelkowitz, M. Reversible execution. *Commun. ACM* 16, 9 (Sept. 1973), 566.

David Bau (davidbau@csail.mit.edu) is a Ph.D. student at MIT's Computer Science and Artificial Intelligence Lab, Cambridge, MA.

Jeff Gray (gray@cs.ua.edu) is a professor in the Department of Computer Science at the University of Alabama, Tuscaloosa.

Caitlin Kelleher (ckelleher@cse.wustl.edu) is an assistant professor in the Department of Computer Science at Washington University, St. Louis, MO.

Josh Sheldon (jsheldon@csail.mit.edu) is Director of Programs and MIT App inventor at MIT's Computer Science and Artificial Intelligence Lab, Cambridge, MA.

Franklyn Turbak (fturbak@cs.wellesley.edu) is an associate professor in the Department of Computer Science at Wellesley College, Wellesley, MA.

© 2017 ACM 0001-0782/17/05 \$15.00.



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

research highlights

P. 82

Technical Perspective What Led Computer Vision to Deep Learning?

By Jitendra Malik

P. 84

ImageNet Classification with Deep Convolutional Neural Networks

By Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton

P. 91

Technical Perspective Low-Depth Arithmetic Circuits

By Avi Wigderson

P. 93

Unexpected Power of Low-Depth Arithmetic Circuits

By Ankit Gupta, Pritish Kamath,
Neeraj Kayal, and Ramprasad Saptharishi

Technical Perspective

What Led Computer Vision to Deep Learning?

By Jitendra Malik

WE ARE IN the middle of the third wave of interest in artificial neural networks as the leading paradigm for machine learning. The first wave dates back to the 1950s, the second to the 1980s, and the third to the 2010s. The following paper by Krizhevksy, Sutskever and Hinton (henceforth KSH) is the paper most responsible for this third wave. Here, I sketch the intellectual history surrounding this work.

The current wave has been called “deep learning” because of the emphasis on having multiple layers of neurons between the input and the output of the neural network; the main architectural design features, however, remain the same as in the second wave, the 1980s. Central to that era was the publication of the back-propagation algorithm for training multilayer perceptrons by Rumelhart, Hinton and Williams.⁷ This algorithm, a consequence of the chain rule of calculus, had been noted before, for example, by Werbos.⁸ However, the Rumelhart et. al. version was significantly more impactful as it was accompanied by interest in distributed representations of knowledge in cognitive science and artificial intelligence, contrasted with the symbolic representations favored by the mainstream researchers.

The second intellectual strand comes from neuroscience, most specifically from Hubel and Wiesel’s studies of cat and monkey visual cortex.^{4,5} They developed a hierarchical model of the visual pathway with neurons in lower areas such as V1 responding to features such as oriented edges and bars, and in higher areas to more specific stimuli (“grandmother cells” in the cartoon version). Fukushima² proposed a neural network architecture for pattern recognition explicitly motivated by Hubel and Wiesel’s hierarchy. His model had alternating layers of simple cells and complex cells, thus incorporating down sampling, and shift invariance, thus

incorporating convolutional structure. LeCun et al.⁶ took the additional step of using backpropagation to train the weights of this network, and what we today call convolutional neural networks were born.

The 1990s and 2000s saw diminished interest in neural networks. Indeed, one of the inside jokes was that having the phrase “neural networks” in the title of a paper was a negative predictor of its chance of getting accepted at the NIPS conference!

A few true believers such as Yoshua Bengio, Geoffrey Hinton, Yann LeCun, and Juergen Schmidhuber persisted, with a lot of effort directed towards developing unsupervised techniques. These did not lead to much success on the benchmark problems that the field cared about, so they remained a minority interest. There were a few technical innovations—max-pooling, dropout, and the use of half-wave rectification (a.k.a ReLU) as the activation function nonlinearity—but before the publication of the KSH paper in 2012, the mainstream computer vision community did not think that neural network based techniques could produce results competitive with our hand designed features and architectures. I was one of those skeptics, and I recall telling Geoff Hinton that con-

vincing the computer vision community would require results on the real-world datasets that we used. Geoff did take this advice to heart and I like to think that conversation was one of the inspirations behind KSH.

What was the secret sauce behind KSH’s success? Besides the technical innovations (such as the use of ReLUs), we must give a lot of credit to “big data” and “big computation.” By big data here I mean the availability of large datasets with category labels, such as ImageNet from Fei-Fei Li’s group, which provided the training data for these large, deep networks with millions of parameters. Previous datasets like Caltech-101 or PASCAL VOC did not have enough training data, and MNIST and CIFAR were regarded as “toy datasets” by the computer vision community. This strand of labeling datasets for benchmarking and for extracting image statistics itself was enabled by the desire of people to upload their photo collections to the Internet on sites such as Flickr. The way big computation proved most helpful was through GPUs, a hardware development initially driven by the needs of the video game industry.

Let me turn now to the impact of the KSH paper. As of this writing, it has 10,245 citations on Google Scholar, remarkable for a paper not yet five years old. I was present at the ECCV ImageNet workshop where the KSH results were presented. Everyone was impressed by the results, but there was debate about their generality. Would the success on whole image classification problems extend to more tasks such as object detection? Was the finding a very fragile one, or was it a robust one that other groups would be able to replicate? Stochastic gradient descent (SGD) can only find local minima, so what is the guarantee the minima we find will be good?

In the true spirit of science, many

**It is my opinion
the following paper
is the most
impactful paper
in machine learning
and computer vision
in the last five years.**

of us, skeptics and believers, went back to our laboratories to explore these questions. Within a year or two, the evidence was quite clear. For example, the R-CNN work of Girshick et al.³ showed the KSH architecture could be modified, by making use of computer vision ideas such as region proposals, to make possible state of the art object detection on PASCAL VOC. Getting SGD to work well is an art, but it could be mastered by students and researchers and corporate employees and yield reproducible results in many different settings. We do not yet have convincing theoretical proof of the robustness of SGD but the empirical evidence is quite compelling, so we leave it to the theoreticians to find an explanation while experimentalists forge ahead. We have realized that generally deeper networks work better, and that overfitting fears are overblown. We have new techniques such as “batch normalization” to deal with regularization, and dropout is not so crucial any more. Practical applications abound.

It is my opinion the following paper is the most impactful paper in machine learning and computer vision in the last five years. It is the paper that led the field of computer vision to embrace deep learning. □

References

1. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Li, F.-F. ImageNet: A Large-scale hierarchical image database. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, (June 20–25, 2009).
2. Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern* 34, 4 (1980), 193–202.
3. Girshick, R., Donahue, J., Darrell, T. and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Vision and Pattern Recognition*, (2014).
4. Hubel, D.H. and Wiesel, T.N. Receptive fields, binocular interactions and functional architecture in the cat's visual cortex. *J. Physiology* 160, 1 (Jan. 1962), 106–154.
5. Hubel, D.H. and Wiesel, T.N. Receptive fields and functional architecture of monkey striate cortex. *J. Physiology* 195, 1 (Mar. 1968), 215–243.
6. LeCun, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Computation* 1 (1989), 541–551.
7. Rumelhart, D.E., Hinton G.E. and Williams R.J. Learning representations by back-propagating errors. *Nature* 323 (Oct. 9, 1986), 533–536.
8. Werbos P. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, 1974.

Jitendra Malik is the Arthur J. Chick Professor of EECS at the University of California at Berkeley.

Copyright held by author.

ACM LEARNING CENTER

RESOURCES FOR LIFELONG LEARNING

learning.acm.org



Online Courses from Skillssoft

Online Books from Safari, Books24x7, Morgan Kaufmann and Syngress

Webinars on today's hottest topics in computing



Association for Computing Machinery

ImageNet Classification with Deep Convolutional Neural Networks

By Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0%, respectively, which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully connected layers we employed a recently developed regularization method called “dropout” that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

1. PROLOGUE

Four years ago, a paper by Yann LeCun and his collaborators was rejected by the leading computer vision conference on the grounds that it used neural networks and therefore provided no insight into how to design a vision system. At the time, most computer vision researchers believed that a vision system needed to be carefully hand-designed using a detailed understanding of the nature of the task. They assumed that the task of classifying objects in natural images would never be solved by simply presenting examples of images and the names of the objects they contained to a neural network that acquired all of its knowledge from this training data.

What many in the vision research community failed to appreciate was that methods that require careful hand-engineering by a programmer who understands the domain do not scale as well as methods that replace the programmer with a powerful general-purpose learning procedure. With enough computation and enough data, learning beats programming for complicated tasks that require the integration of many different, noisy cues.

Four years ago, while we were at the University of Toronto, our deep neural network called SuperVision almost halved the error rate for recognizing objects in natural images and triggered an overdue paradigm shift in computer vision. Figure 4 shows some examples of what SuperVision can do.

SuperVision evolved from the multilayer neural networks

that were widely investigated in the 1980s. These networks used multiple layers of feature detectors that were all learned from the training data. Neuroscientists and psychologists had hypothesized that a hierarchy of such feature detectors would provide a robust way to recognize objects but they had no idea how such a hierarchy could be learned. There was great excitement in the 1980s because several different research groups discovered that multiple layers of feature detectors could be trained efficiently using a relatively straight-forward algorithm called backpropagation^{18, 22, 27, 33} to compute, for each image, how the classification performance of the whole network depended on the value of the weight on each connection.

Backpropagation worked well for a variety of tasks, but in the 1980s it did not live up to the very high expectations of its advocates. In particular, it proved to be very difficult to learn networks with many layers and these were precisely the networks that should have given the most impressive results. Many researchers concluded, incorrectly, that learning a deep neural network from random initial weights was just too difficult. Twenty years later, we know what went wrong: for deep neural networks to shine, they needed far more labeled data and hugely more computation.

2. INTRODUCTION

Current approaches to object recognition make essential use of machine learning methods. To improve their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting. Until recently, datasets of labeled images were relatively small—on the order of tens of thousands of images (e.g., NORB,¹⁹ Caltech-101/256,^{8, 10} and CIFAR-10/100¹⁴). Simple recognition tasks can be solved quite well with datasets of this size, especially if they are augmented with label-preserving transformations. For example, the current-best error rate on the MNIST digit-recognition task (<0.3%) approaches human performance.⁵ But objects in realistic settings exhibit considerable variability, so to learn to recognize them it is necessary to use much larger training sets. And indeed, the shortcomings of small image datasets have been widely recognized (e.g., Ref.²⁵), but it has only recently become possible to collect labeled datasets with millions of

The original version of this paper was published in the *Proceedings of the 25th International Conference on Neural Information Processing Systems* (Lake Tahoe, NV, Dec. 2012), 1097–1105.

images. The new larger datasets include LabelMe,²⁸ which consists of hundreds of thousands of fully segmented images, and ImageNet,⁷ which consists of over 15 million labeled high-resolution images in over 22,000 categories.

To learn about thousands of objects from millions of images, we need a model with a large learning capacity. However, the immense complexity of the object recognition task means that this problem cannot be specified even by a dataset as large as ImageNet, so our model should also have lots of prior knowledge to compensate for all the data we do not have. Convolutional neural networks (CNNs) constitute one such class of models.^{9,15,17,19,21,26,32} Their capacity can be controlled by varying their depth and breadth, and they also make strong and mostly correct assumptions about the nature of images (namely, stationarity of statistics and locality of pixel dependencies). Thus, compared to standard feedforward neural networks with similarly sized layers, CNNs have much fewer connections and parameters and so they are easier to train, while their theoretically best performance is likely to be only slightly worse.

Despite the attractive qualities of CNNs, and despite the relative efficiency of their local architecture, they have still been prohibitively expensive to apply in large scale to high-resolution images. Luckily, current GPUs, paired with a highly optimized implementation of 2D convolution, are powerful enough to facilitate the training of interestingly-large CNNs, and recent datasets such as ImageNet contain enough labeled examples to train such models without severe overfitting.

The specific contributions of this paper are as follows: we trained one of the largest CNNs to date on the subsets of ImageNet used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)-2010 and ILSVRC-2012 competitions² and achieved by far the best results ever reported on these datasets. We wrote a highly optimized GPU implementation of 2D convolution and all the other operations inherent in training CNNs, which we make available publicly.^a Our network contains a number of new and unusual features which improve its performance and reduce its training time, which are detailed in Section 4. The size of our network made overfitting a significant problem, even with 1.2 million labeled training examples, so we used several effective techniques for preventing overfitting, which are described in Section 5. Our final network contains five convolutional and three fully connected layers, and this depth seems to be important: we found that removing any convolutional layer (each of which contains no more than 1% of the model's parameters) resulted in inferior performance.

In the end, the network's size is limited mainly by the amount of memory available on current GPUs and by the amount of training time that we are willing to tolerate. Our network takes between 5 and 6 days to train on two GTX 580 3GB GPUs. All of our experiments suggest that our results can be improved simply by waiting for faster GPUs and bigger datasets to become available.

3. THE DATASET

ImageNet is a dataset of over 15 million labeled high-resolution images belonging to roughly 22,000 categories. The images were collected from the web and labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool. Starting in 2010, as part of the Pascal Visual Object Challenge, an annual competition called the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) has been held. ILSVRC uses a subset of ImageNet with roughly 1000 images in each of 1000 categories. In all, there are roughly 1.2 million training images, 50,000 validation images, and 150,000 testing images.

ILSVRC-2010 is the only version of ILSVRC for which the test set labels are available, so this is the version on which we performed most of our experiments. Since we also entered our model in the ILSVRC-2012 competition, in Section 7 we report our results on this version of the dataset as well, for which test set labels are unavailable. On ImageNet, it is customary to report two error rates: top-1 and top-5, where the top-5 error rate is the fraction of test images for which the correct label is not among the five labels considered most probable by the model.

ImageNet consists of variable-resolution images, while our system requires a constant input dimensionality. Therefore, we down-sampled the images to a fixed resolution of 256×256 . Given a rectangular image, we first rescaled the image such that the shorter side was of length 256, and then cropped out the central 256×256 patch from the resulting image. We did not pre process the images in any other way, except for subtracting the mean activity over the training set from each pixel. So we trained our network on the (centered) raw RGB values of the pixels.

4. THE ARCHITECTURE

The architecture of our network is summarized in Figure 2. It contains eight learned layers—five convolutional and three fully connected. Below, we describe some of the novel or unusual features of our network's architecture. Sections 4.1–4.4 are sorted according to our estimation of their importance, with the most important first.

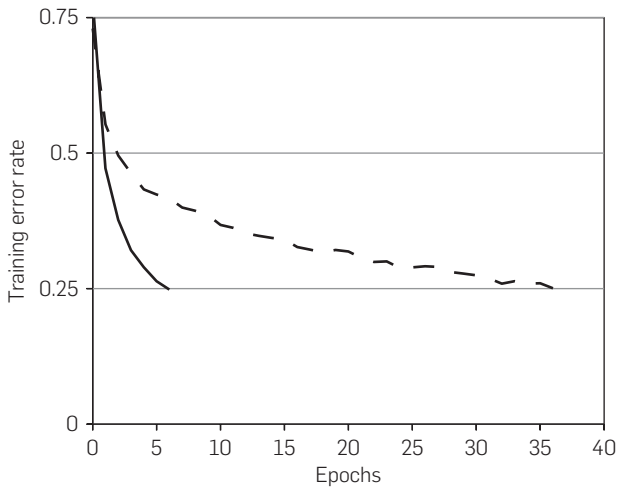
4.1. Rectified Linear Unit nonlinearity

The standard way to model a neuron's output f as a function of its input x is with $f(x) = \tanh(x)$ or $f(x) = (1 + e^{-x})^{-1}$. In terms of training time with gradient descent, these saturating nonlinearities are much slower than the non-saturating nonlinearity $f(x) = \max(0, x)$. Following Nair and Hinton,²⁴ we refer to neurons with this non linearity as Rectified Linear Units (ReLU). Deep CNNs with ReLUs train several times faster than their equivalents with tanh units. This is demonstrated in Figure 1, which shows the number of iterations required to reach 25% training error on the CIFAR-10 dataset for a particular four-layer convolutional network. This plot shows that we would not have been able to experiment with such large neural networks for this work if we had used traditional saturating neuron models.

We are not the first to consider alternatives to traditional neuron models in CNNs. For example, Jarrett et al.¹³ claim that the nonlinearity $f(x) = |\tanh(x)|$ works particularly well with their type of contrast normalization followed by local

^a <http://code.google.com/p/cuda-convnet/>.

Figure 1. A four-layer convolutional neural network with ReLUs (solid line) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons (dashed line). The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.



average pooling on the Caltech-101 dataset. However, on this dataset the primary concern is preventing overfitting, so the effect they are observing is different from the accelerated ability to fit the training set which we report when using ReLUs. Faster learning has a great influence on the performance of large models trained on large datasets.

4.2. Training on multiple GPUs

A single GTX 580 GPU has only 3GB of memory, which limits the maximum size of the networks that can be trained on it. It turns out that 1.2 million training examples are enough to train networks which are too big to fit on one GPU. Therefore we spread the net across two GPUs. Current GPUs are particularly well-suited to cross-GPU parallelization, as they are able to read from and write to one another's memory directly, without going through host machine memory. The parallelization scheme that we employ essentially puts half of the kernels (or neurons) on each GPU, with one additional trick: the GPUs communicate only in certain layers. This means that, for example, the kernels of layer 3 take input from all kernel maps in layer 2. However, kernels in layer 4 take input only from those kernel maps in layer 3 which reside on the same GPU. Choosing the pattern of connectivity is a problem for cross-validation, but this allows us to precisely tune the amount of communication until it is an acceptable fraction of the amount of computation.

The resultant architecture is somewhat similar to that of the "columnar" CNN employed by Cireřan et al.,⁴ except that our columns are not independent (see Figure 2). This scheme reduces our top-1 and top-5 error rates by 1.7% and 1.2%, respectively, as compared with a net with half as many kernels in each convolutional layer trained on one GPU. The two-GPU

net takes slightly less time to train than the one-GPU net.^b

4.3. Local response normalization

ReLUs have the desirable property that they do not require input normalization to prevent them from saturating. If at least some training examples produce a positive input to a ReLU, learning will happen in that neuron. However, we still find that the following local normalization scheme aids generalization. Denoting by $a_{x,y}^i$ the activity of a neuron computed by applying kernel i at position (x,y) and then applying the ReLU nonlinearity, the response-normalized activity $b_{x,y}^i$ is given by the expression

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0, i-n/2)}^{\min(N-1, i+n/2)} (a_{x,y}^j)^2 \right)^\beta,$$

where the sum runs over n "adjacent" kernel maps at the same spatial position, and N is the total number of kernels in the layer. The ordering of the kernel maps is of course arbitrary and determined before training begins. This sort of response normalization implements a form of lateral inhibition inspired by the type found in real neurons, creating competition for big activities among neuron outputs computed using different kernels. The constants k, n, α , and β are hyperparameters whose values are determined using a validation set; we used $k = 2, n = 5, \alpha = 10^{-4}$, and $\beta = 0.75$. We applied this normalization after applying the ReLU nonlinearity in certain layers (see Section 4.5).

This scheme bears some resemblance to the local contrast normalization scheme of Jarrett et al.,¹³ but ours would be more correctly termed "brightness normalization," since we do not subtract the mean activity. Response normalization reduces our top-1 and top-5 error rates by 1.4% and 1.2%, respectively. We also verified the effectiveness of this scheme on the CIFAR-10 dataset: a four-layer CNN achieved a 13% test error rate without normalization and 11% with normalization.^c

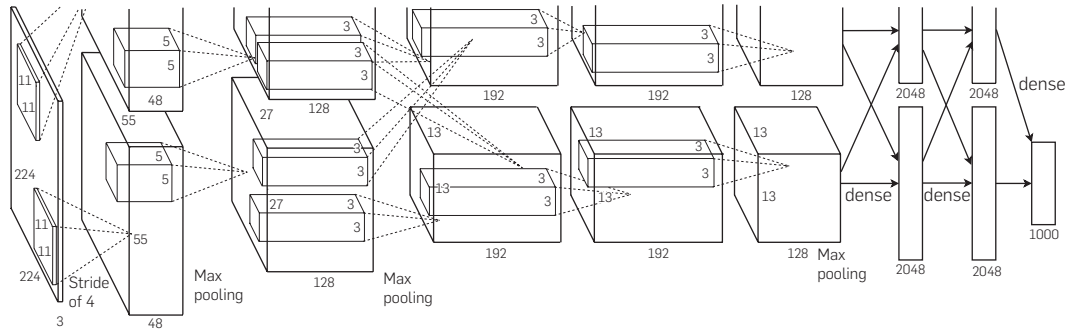
4.4. Overlapping pooling

Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map. Traditionally, the neighborhoods summarized by adjacent pooling units do not overlap (e.g., Refs.^{5,13,20}). To be more precise, a pooling layer can be thought of as consisting of a grid of pooling units spaced s pixels apart, each summarizing a neighborhood of size $z \times z$ centered at the location of the pooling unit. If we set $s = z$, we obtain traditional local pooling as commonly employed in CNNs. If we set $s < z$, we obtain overlapping

^b The one-GPU net actually has the same number of kernels as the two-GPU net in the final convolutional layer. This is because most of the net's parameters are in the first fully connected layer, which takes the last convolutional layer as input. So to make the two nets have approximately the same number of parameters, we did not halve the size of the final convolutional layer (nor the fully connected layers which follow). Therefore this comparison is biased in favor of the one-GPU net, since it is bigger than "half the size" of the two-GPU net.

^c We cannot describe this network in detail due to space constraints, but it is specified precisely by the code and parameter files provided here: <http://code.google.com/p/cuda-convnet/>.

Figure 2. An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 290,400–186,624–64,896–64,896–43,264–4096–4096–1000.



pooling. This is what we use throughout our network, with $s = 2$ and $z = 3$. This scheme reduces the top-1 and top-5 error rates by 0.4% and 0.3%, respectively, as compared with the non overlapping scheme $s = 2, z = 2$, which produces output of equivalent dimensions. We generally observe during training that models with overlapping pooling find it slightly more difficult to overfit.

4.5. Overall architecture

Now we are ready to describe the overall architecture of our CNN. As depicted in Figure 2, the net contains eight layers with weights; the first five are convolutional and the remaining three are fully connected. The output of the last fully connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels. Our network maximizes the multinomial logistic regression objective, which is equivalent to maximizing the average across training cases of the log-probability of the correct label under the prediction distribution.

The kernels of the second, fourth, and fifth convolutional layers are connected only to those kernel maps in the previous layer which reside on the same GPU (see Figure 2). The kernels of the third convolutional layer are connected to all kernel maps in the second layer. The neurons in the fully-connected layers are connected to all neurons in the previous layer. Response-normalization layers follow the first and second convolutional layers. Max-pooling layers, of the kind described in Section 4.4, follow both response-normalization layers as well as the fifth convolutional layer. The ReLU non linearity is applied to the output of every convolutional and fully connected layer.

The first convolutional layer filters the $224 \times 224 \times 3$ input image with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels (this is the distance between the receptive field centers of neighboring neurons in a kernel map). The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs

of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully connected layers have 4096 neurons each.

5. REDUCING OVERFITTING

Our neural network architecture has 60 million parameters. Although the 1000 classes of ILSVRC make each training example impose 10 bits of constraint on the mapping from image to label, this turns out to be insufficient to learn so many parameters without considerable overfitting. Below, we describe the two primary ways in which we combat overfitting.

5.1. Data augmentation

The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations (e.g., Refs.^{4,5,30}). We employ two distinct forms of data augmentation, both of which allow transformed images to be produced from the original images with very little computation, so the transformed images do not need to be stored on disk. In our implementation, the transformed images are generated in Python code on the CPU while the GPU is training on the previous batch of images. So these data augmentation schemes are, in effect, computationally free.

The first form of data augmentation consists of generating image translations and horizontal reflections. We do this by extracting random 224×224 patches (and their horizontal reflections) from the 256×256 images and training our network on these extracted patches.^d This increases the size of our training set by a factor of 2048, though the resulting training examples are, of course, highly inter dependent. Without this scheme, our network suffers from substantial overfitting, which would have forced us to use much smaller networks. At test time, the network makes a prediction by extracting five 224×224 patches (the four corner patches and the center patch) as well as their horizontal reflections (hence 10 patches in all), and averaging the predictions made by the network's softmax layer on the ten patches.

^d This is the reason why the input images in Figure 2 are $224 \times 224 \times 3$ dimensional.

The second form of data augmentation consists of altering the intensities of the RGB channels in training images. Specifically, we perform PCA on the set of RGB pixel values throughout the ImageNet training set. To each training image, we add multiples of the found principal components, with magnitudes proportional to the corresponding eigen values times a random variable drawn from a Gaussian with mean 0 and standard deviation 0.1. Therefore to each RGB image pixel $I_{xy} = [I_{xy}^R, I_{xy}^G, I_{xy}^B]^T$ we add the following quantity:

$$[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1 \lambda_1, \alpha_2 \lambda_2, \alpha_3 \lambda_3]^T,$$

where \mathbf{p}_i and λ_i are i th eigenvector and eigenvalue of the 3×3 covariance matrix of RGB pixel values, respectively, and α_i is the aforementioned random variable. Each α_i is drawn only once for all the pixels of a particular training image until that image is used for training again, at which point it is re drawn. This scheme approximately captures an important property of natural images, namely, that object identity is invariant to changes in the intensity and color of the illumination. This scheme reduces the top-1 error rate by over 1%.

5.2. Dropout

Combining the predictions of many different models is a very successful way to reduce test errors,^{1,3} but it appears to be too expensive for big neural networks that already take several days to train. There is, however, a very efficient version of model combination that only costs about a factor of two during training. The recently introduced technique, called “dropout”,¹² consists of setting to zero the output of each hidden neuron with probability 0.5. The neurons which are “dropped out” in this way do not contribute to the forward pass and do not participate in back propagation. So every time an input is presented, the neural network samples a different architecture, but all these architectures share weights. This technique reduces complex co adaptations of neurons, since a neuron cannot rely on the presence of particular other neurons. It is, therefore, forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. At test time, we use all the neurons but multiply their outputs by 0.5, which is a reasonable approximation to taking the geometric mean of the predictive distributions produced by the exponentially-many dropout networks.

We use dropout in the first two fully connected layers of Figure 2. Without dropout, our network exhibits substantial overfitting. Dropout roughly doubles the number of iterations required to converge.

6. DETAILS OF LEARNING

We trained our models using stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, and weight decay of 0.0005. We found that this small amount of weight decay was important for the model to learn. In other words, weight decay here is not merely a regularizer: it reduces the model’s training error. The update rule for weight w was

$$v_{i+1} := 0.9 \cdot v_i - 0.0005 \cdot \epsilon \cdot w_i - \epsilon \cdot \left\langle \frac{\partial L}{\partial w} \Big|_{w_i} \right\rangle_{D_i},$$

$$w_{i+1} := w_i + v_{i+1},$$

where i is the iteration index, v is the momentum variable, ϵ is the learning rate, and $\langle \frac{\partial L}{\partial w} \Big|_{w_i} \rangle_{D_i}$ is the average over the i th batch D_i of the derivative of the objective with respect to w , evaluated at w_i .

We initialized the weights in each layer from a zero-mean Gaussian distribution with standard deviation 0.01. We initialized the neuron biases in the second, fourth, and fifth convolutional layers, as well as in the fully connected hidden layers, with the constant 1. This initialization accelerates the early stages of learning by providing the ReLUs with positive inputs. We initialized the neuron biases in the remaining layers with the constant 0.

We used an equal learning rate for all layers, which we adjusted manually throughout training. The heuristic which we followed was to divide the learning rate by 10 when the validation error rate stopped improving with the current learning rate. The learning rate was initialized at 0.01 and reduced three times prior to termination. We trained the network for roughly 90 cycles through the training set of 1.2 million images, which took 5–6 days on two NVIDIA GTX 580 3GB GPUs.

7. RESULTS

Our results on ILSVRC-2010 are summarized in Table 1. Our network achieves top-1 and top-5 test set error rates of 37.5% and 17.0%, respectively.^e The best performance achieved during the ILSVRC-2010 competition was 47.1% and 28.2% with an approach that averages the predictions produced from six sparse-coding models trained on different features,² and since then the best published results are 45.7% and 25.7% with an approach that averages the predictions of two classifiers trained on Fisher Vectors (FVs) computed from two types of densely sampled features.²⁹

We also entered our model in the ILSVRC-2012 competition and report our results in Table 2. Since the ILSVRC-2012 test set labels are not publicly available, we cannot report test

^e The error rates without averaging predictions over 10 patches as described in Section 5.1 are 39.0% and 18.3%.

Table 1. Comparison of results on ILSVRC-2010 test set.

| Model | Top-1 (%) | Top-5 (%) |
|-----------------------------------|-------------|-------------|
| <i>Sparse coding</i> ² | 47.1 | 28.2 |
| <i>SIFT + FVs</i> ²⁹ | 45.7 | 25.7 |
| CNN | 37.5 | 17.0 |

In *italics* are best results achieved by others.

Table 2. Comparison of error rates on ILSVRC-2012 validation and test sets.

| Model | Top-1 (val, %) | Top-5 (val, %) | Top-5 (test, %) |
|--------------------------------|----------------|----------------|-----------------|
| <i>SIFT + FVs</i> ⁶ | – | – | 26.2 |
| 1 CNN | 40.7 | 18.2 | – |
| 5 CNNs | 38.1 | 16.4 | 16.4 |
| 1 CNN* | 39.0 | 16.6 | – |
| 7 CNNs* | 36.7 | 15.4 | 15.3 |

In *italics* are best results achieved by others. Models with an “*” were “pre-trained” to classify the entire ImageNet 2011 Fall release (see Section 7 for details).

error rates for all the models that we tried. In the remainder of this paragraph, we use validation and test error rates interchangeably because in our experience they do not differ by more than 0.1% (see Table 2). The CNN described in this paper achieves a top-5 error rate of 18.2%. Averaging the predictions of five similar CNNs gives an error rate of 16.4%. Training one CNN, with an extra sixth convolutional layer over the last pooling layer, to classify the entire ImageNet Fall 2011 release (15M images, 22K categories), and then “fine-tuning” it on ILSVRC-2012 gives an error rate of 16.6%. Averaging the predictions of two CNNs that were pre-trained on the entire Fall 2011 release with the afore mentioned five CNNs gives an error rate of 15.3%. The second-best contest entry achieved an error rate of 26.2% with an approach that averages the predictions of several classifiers trained on FVs computed from different types of densely sampled features.⁶

Finally, we also report our error rates on the Fall 2009 version of ImageNet with 10,184 categories and 8.9 million images. On this dataset we follow the convention in the literature of using half of the images for training and half for testing. Since there is no established test set, our split necessarily differs from the splits used by previous authors, but this does not affect the results appreciably. Our top-1 and top-5 error rates on this dataset are 67.4% and 40.9%, attained by the net described above but with an additional, sixth convolutional layer over the last pooling layer. The best published results on this dataset are 78.1% and 60.9%.²³

7.1. Qualitative evaluations

Figure 3 shows the convolutional kernels learned by the network’s two data-connected layers. The network has learned a variety of frequency- and orientation-selective kernels, as well as various colored blobs. Notice the specialization exhibited by the two GPUs, a result of the restricted connectivity described in Section 4.5. The kernels on GPU 1 are largely color-agnostic, while the kernels on GPU 2 are largely

color-specific. This kind of specialization occurs during every run and is independent of any particular random weight initialization (modulo a renumbering of the GPUs).

In the left panel of Figure 4 we qualitatively assess what the network has learned by computing its top-5 predictions on eight test images. Notice that even off-center objects, such as the mite in the top-left, can be recognized by the net. Most of the top-5 labels appear reasonable. For example, only other types of cat are considered plausible labels for the leopard. In some cases (grille, cherry) there is genuine ambiguity about the intended focus of the photograph.

Another way to probe the network’s visual knowledge is to consider the feature activations induced by an image at the last, 4096-dimensional hidden layer. If two images produce feature activation vectors with a small Euclidean separation, we can say that the higher levels of the neural network consider them to be similar. Figure 4 shows five images from the test set and the six images from the training set that are most similar to each of them according to this measure. Notice that at the pixel level, the retrieved training images are generally not close in L2 to the query images in the first column. For example, the retrieved dogs and elephants appear in a variety

Figure 3. Ninety-six convolutional kernels of size $11 \times 11 \times 3$ learned by the first convolutional layer on the $224 \times 224 \times 3$ input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2 (see Section 7.1 for details).

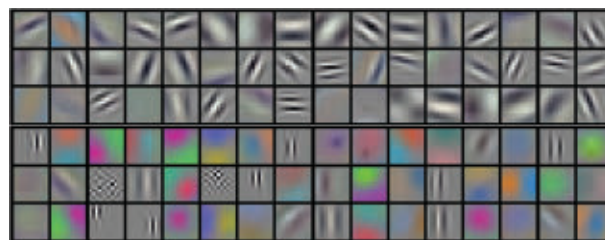
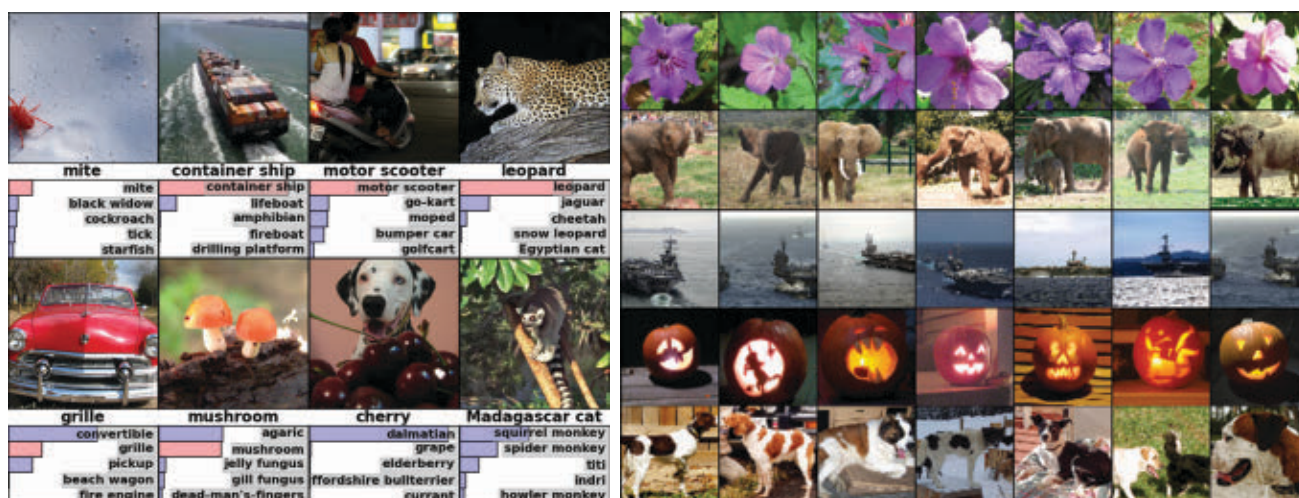


Figure 4. (Left) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (Right) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.



of poses. We present the results for many more test images in the supplementary material.


Computing similarity by using Euclidean distance between two 4096-dimensional, real-valued vectors is inefficient, but it could be made efficient by training an auto encoder to compress these vectors to short binary codes. This should produce a much better image retrieval method than applying auto encoders to the raw pixels,¹⁶ which does not make use of image labels and hence has a tendency to retrieve images with similar patterns of edges, whether or not they are semantically similar.

8. DISCUSSION

Our results show that a large, deep CNN is capable of achieving record-breaking results on a highly challenging dataset using purely supervised learning. It is notable that our network's performance degrades if a single convolutional layer is removed. For example, removing any of the middle layers results in a loss of about 2% for the top-1 performance of the network. So the depth really is important for achieving our results.

To simplify our experiments, we did not use any unsupervised pre-training even though we expect that it will help, especially if we obtain enough computational power to significantly increase the size of the network without obtaining a corresponding increase in the amount of labeled data. Thus far, our results have improved as we have made our network larger and trained it longer but we still have many orders of magnitude to go in order to match the infero temporal pathway of the human visual system. Ultimately we would like to use very large and deep convolutional nets on video sequences where the temporal structure provides very helpful information, that is, missing or far less obvious in static images.

9. EPILOGUE

The response of the computer vision community to the success of SuperVision was impressive. Over the next year or two, they switched to using deep neural networks and these are now widely deployed by Google, Facebook, Microsoft, Baidu and many other companies. By 2015, better hardware, more hidden layers, and a host of technical advances reduced the error rate of deep convolutional neural nets by a further factor of three so that they are now quite close to human performance for static images.^{11,31} Much of the credit for this revolution should go to the pioneers who spent many years developing the technology of CNNs, but the essential missing ingredient was supplied by FeiFei et al.⁷ who put a huge effort into producing a labeled dataset that was finally large enough to show what neural networks could really do. 

References

- Bell, R., Koren, Y. Lessons from the netflix prize challenge. *ACM SIGKDD Explor. NewsL.* 9, 2 (2007), 75–79.
- Berg, A., Deng, J., Fei-Fei, L. Large scale visual recognition challenge 2010. www.image-net.org/challenges/2010.
- Breiman, L. Random forests. *Mach. Learn.* 45, 1 (2001), 5–32.
- Ciresan, D., Meier, U., Masci, J., Gambardella, L., Schmidhuber, J. High-performance neural networks for visual object classification. *Arxiv preprint arXiv:1102.0183*, 2011.
- Ciresan, D., Meier, U., Schmidhuber, J. Multi-column deep neural networks for image classification. *Arxiv preprint arXiv:1202.2745*, 2012.
- Deng, J., Berg, A., Sathesh, S., Su, H., Khosla, A., Fei-Fei, L. In *ILSVRC-2012* (2012).
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *CVPR09* (2009).

- Fei-Fei, L., Fergus, R., Perona, P. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Comput. Vision Image Understanding* 106, 1 (2007), 59–70.
- Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.* 36, 4 (1980), 193–202.
- Griffin, G., Holub, A., Perona, P. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.
- He, K., Zhang, X., Ren, S., Sun, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
- Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).
- Jarrett, K., Kavukcuoglu, K., Ranzato, M.A., LeCun, Y. What is the best multi-stage architecture for object recognition? In *International Conference on Computer Vision* (2009). IEEE, 2146–2153.
- Krizhevsky, A. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.
- Krizhevsky, A. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 2010.
- Krizhevsky, A., Hinton, G. Using very deep autoencoders for content-based image retrieval. In *ESANN* (2011).
- LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., et al. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems* (1990).
- LeCun, Y. Une procedure d'apprentissage pour reseau a seuil asymmetrique (a learning scheme for asymmetric threshold networks). 1985.
- LeCun, Y., Huang, F., Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. Volume 2* (2004). IEEE, II–97.
- LeCun, Y., Kavukcuoglu, K., Farabet, C. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)* (2010). IEEE, 253–256.
- Lee, H., Grosse, R., Ranganath, R., Ng, A. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009). ACM, 609–616.
- Linnainmaa, S. Taylor expansion of the accumulated rounding error. *BIT Numer. Math.* 16, 2 (1976), 146–160.
- Mensink, T., Verbeek, J., Perronnin, F., Csurka, G. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *ECCV – European Conference on Computer Vision* (Florence, Italy, Oct. 2012).
- Nair, V., Hinton, G.E. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning* (2010).
- Pinto, N., Cox, D., DiCarlo, J. Why is real-world visual object recognition hard? *PLoS Comput. Biol.* 4, 1 (2008), e27.
- Pinto, N., Doukhan, D., DiCarlo, J., Cox, D. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput. Biol.* 5, 11 (2009), e1000579.
- Rumelhart, D.E., Hinton, G.E., Williams, R.J. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.
- Russell, BC, Torralba, A., Murphy, K., Freeman, W. Labelme: A database and web-based tool for image annotation. *Int. J. Comput. Vis.* 77, 1 (2008), e157–173.
- Sánchez, J., Perronnin, F. High-dimensional signature compression for large-scale image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011* (2011). IEEE, 1665–1672.
- Simard, P., Steinkraus, D., Platt, J. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*. Volume 2 (2003), 958–962.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), 1–9.
- Turaga, S., Murray, J., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W., Seung, H. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Comput.* 22, 2 (2010), 511–538.
- Werbos, P. Beyond regression: New tools for prediction and analysis in the behavioral sciences, 1974.

Alex Krizhevsky and Geoffrey E. Hinton
{akrizhevsky, geoffhinton}@google.com,
Google Inc.

Ilya Sutskever (ilyasu@openai.com),
OpenAI.

Technical Perspective

Low-Depth Arithmetic Circuits

By Avi Wigderson

THE COMPUTATIONS OF polynomials (over a field, which we shall throughout assume is of zero or large enough characteristic) using arithmetic operations of addition and multiplication (and possibly division) are of course as natural as the computation of Boolean functions via logical gates, and capture many natural important tasks including Fourier transforms, linear algebra, matrix computations and more generally symbolic algebraic computations arising in many settings. Arithmetic circuits are the natural computational model for understanding the computational complexity of such tasks just like Boolean circuits are for Boolean functions. The presence of algebraic structure and mathematical tools supplied by centuries of work in algebra were a source of hope that understanding arithmetic circuits will be much faster and easier than their Boolean siblings. And while we generally know more about arithmetic circuits, their power is far from understood, and in particular, the arithmetic analog VP vs. VNP of the Boolean P vs. NP problem as formulated by Valiant⁸ is wide open.

The past few years have seen a revolution in our understanding of arithmetic circuits. Surprising new upper bounds, combined with new powerful techniques of proving lower bounds, have brought us to recognize the mysterious importance of very shallow circuits to capture the long-term goals of the field, and to pinpointing with uncommon precision the complexity of natural problems in this model. These developments have rejuvenated the hope, and give a concrete program, to the possible separation of Valiant's classes VP and VNP. The following paper of Gupta et. al. on the "chasm at depth 3" is one of the culminations of this new understanding. I will now briefly explain the "chasm" phenomenon, and some of these developments, on which the authors of the paper elaborate.

A series of important works in the 1980s on constant-depth Boolean circuits gives a very good picture of their limitations, including tight exponential lower bounds on extremely simple functions like the symmetric functions. The basic message is that constant-depth (and polynomial size) is an extremely weak class of algorithms. Strangely (and specifically over large enough fields) this intuition fails completely for arithmetic circuits. In 1980, Ben-Or already made the important simple observation that the symmetric polynomials can be computed in depth-3 by a quadratic size formula! So, the challenge to prove exponential lower bounds for this simple model was on. Such bounds were proved under further restrictions (like homogeneity) by Nisan-Wigderson,⁶ who introduced important techniques of partial derivatives and random restrictions to the study arithmetic circuits. While the best general lower bound is quadratic (matching Ben-Or's result for symmetric polynomials), there was still a belief that constant depth is a weak model and we should easily prove much better bounds, for harder functions like permanent and even determinant. Still, no such progress followed for over a decade.

A surprising and influential paper

While we generally know more about arithmetic circuits, their power is far from understood.

by Agrawal and Vinay,¹ which they called a "chasm at depth 4" appeared in 2008. Its clear message: proving lower bounds for (even homogeneous) depth-4 circuits is as hard (or, for optimists, as useful) as proving lower bounds on general circuits. Extending the celebrated depth reduction technique of Valiant, Skyum, Berkowitz and Rackoff,⁹ this paper (with subsequent improvements of Tavenas and Koiran) shows that any arithmetic circuit of size s computing a degree d polynomial can also be computed by a homogeneous depth-4 circuit of size $s^{O(\sqrt{d})}$. For example, proving a subexponential $n^{\omega(\sqrt{n})}$ lower bound for computing the permanent on $n \times n$ matrices, on such a weak constant-depth circuit would separate VP from VNP! But recall, even for depth-3 we were stuck.

Next, a series of papers, mainly by subsets of the present authors and a few others got us extremely close to this goal! Using deep ideas and results from algebraic geometry originating from the work of Hilbert in commutative algebra, they have extended the method of partial derivatives to the much stronger "shifted partial derivatives" and combined with other ideas including very fine combinatorial analysis were able to reach the chasm, but not cross it. More precisely they proved lower bounds of $n^{\Omega(\sqrt{n})}$ or both determinant and permanent. Note that for the determinant this lower bound is tight, and changing the Ω to ω in this expression for the permanent would thus separate the two and hence separate VP from VNP.

So far for depth 4. The following paper proves that the very same chasm actually exists in depth 3, at least over fields of characteristic 0 like the rational numbers. More precisely, as above, every size s arithmetic circuit computing a polynomial of degree d can be computed by a depth-3 circuit of size $s^{O(\sqrt{d})}$ (which un-



Association for
Computing Machinery

ACM Conference Proceedings Now Available via Print-on-Demand!

*Did you know that you can
now order many popular
ACM conference proceedings
via print-on-demand?*

Institutions, libraries and individuals can choose from more than 100 titles on a continually updated list through Amazon, Barnes & Noble, Baker & Taylor, Ingram and NACSCORP: CHI, KDD, Multimedia, SIGIR, SIGCOMM, SIGCSE, SIGMOD/PODS, and many more.

**For available titles and
ordering info, visit:
librarians.acm.org/pod**



fortunately are not homogeneous anymore and actually have very large degrees). It is difficult to explain to non-experts why this is so unexpected, but as before it carries the same message: proving $n^{-(\sqrt{n})}$ lower bound for computing the permanent even on depth-3 arithmetic circuits would separate VP from VNP.

Again, for optimists, this means that we are extremely close to resolving a major goal of the field. For pessimists, this may be simply an indication of how strong are depth-3 circuits and how hopeless proving lower bounds is even for them. I am on the optimists' side. As far as we know, in the arithmetic setting we have no barriers (aka excuses) of the "natural proofs" or "relativization" varieties, which seem to explain our failure so far to prove lower bounds in the Boolean setting. Moreover, the arithmetic setting is making good on the promise of providing mathematical techniques, which may help resolve its major problems, as the line of work here indicates. Another indication, of course, the Geometric Complexity Theory (GCT) approach of Mulmuley and Sohoni,⁵ which approaches the VP vs. VNP question from a different direction, based on invariant theory and representation theory. Indeed, some relations and connections between the two approaches were discovered, and the growing interests of pure mathematicians in these computational complexity questions is encouraging.

As we have grown to expect of computational complexity, there are myriad connections of this research in arithmetic complexity to other seemingly distinct subareas of the field. One important connection to pseudorandomness, in particular the intimate relationship between derandomizing the probabilistic algorithm for Polynomial Identity Testing (PIT) and arithmetic lower bounds discovered by Kabanets and Impagliazzo.⁴ Subsequently, Dvir and Shpilka² initiated a program to understand this question for very shallow circuits, making connections to locally decodable and correctable codes on the one hand and to combinatorial geometry (mainly incidence theory) on the other. This

has lead to a long sequence of papers obtaining both new lower bounds and new identity testing algorithms for larger and larger classes of low-depth and other circuit models. Yet another exciting connection was recently made by Grochow and Pitassi³ between arithmetic complexity, PIT and proof complexity, in what will surely develop and possibly lead to lower bounds in that field.

Let me conclude with a riddle: Can depth-3 lower bounds on Boolean circuits lead to "real" lower bounds, as they do in the arithmetic setting? Yes! The positive answer in Valiant's⁷ predates all of these developments. In this paper Valiant shows that any function requiring $\exp(n)$ size depth-3 Boolean circuits cannot be computed by a linear-size, logarithmic depth circuit. The state-of-art in Boolean circuit lower bounds is so pathetic that the latter remained one of its major goals since Valiant's paper, and can be achieved via strong enough depth-3 lower bounds.

On to proving real lower bounds! ☑

References

1. Agrawal, M. and Vinay, V. Arithmetic circuits: A chasm at depth four. In *Proceedings of the Annual Symposium on Foundations of Computer Science*. IEEE, 2008, 67–75.
2. Dvir, Z. and Shpilka, A. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM Journal on Computing* 36, 5 (2007), 1404–1434.
3. Grochow, J. et al. Circuit complexity, proof complexity, and polynomial identity testing. In *Proceedings of IEEE 55th Annual Symposium Foundations of Computer Science*. IEEE, 2014, 110–119.
4. Kabanets, V. and Impagliazzo, R. Identity tests means proving circuit lower bounds. *Comput. Complexity* 13, 1–2 (2004), 1–46.
5. Mulmuley, K. and Sohoni, M. Geometric complexity theory I: An approach to the P vs. NP and related problems. *SIAM J. Comput.* 31, 2 (201), 496–526.
6. Nisan, N. and Wigderson, A. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity* 6, 3 (1996), 217–234.
7. Valiant, L.G. *Graph-theoretic arguments in low-level complexity*. Springer, 1977.
8. Valiant L.G. Completeness classes in algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, (1979), 249–261.
9. Valiant, L.G., Skyum, S., Berkowitz, S., and Rackoff, C. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing* 12, 4 (1983), 641–644.

Avi Wigderson is a professor in the School of Mathematics, Institute for Advanced Study, at Princeton University, Princeton, NJ.

Copyright held by author.

Unexpected Power of Low-Depth Arithmetic Circuits

By Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi

Abstract

Complexity theory aims at understanding the “hardness” of certain tasks with respect to the number of “basic operations” required to perform it. In the case of arithmetic circuit complexity, the goal is to understand how hard it is to compute a formal polynomial in terms of the number of additions and multiplications required. Several earlier results have shown that it is possible to rearrange basic computational elements in surprising ways to give more efficient algorithms. The main result of this article is along a similar vein.

We present a simulation of any formal polynomial computed by an arithmetic circuit by a *shallow* circuit of not-much larger size. Roughly, depth corresponds to the time required in a massively parallel computation. This result shows that efficient computations can be speedup to run in depth three, while requiring surprisingly low size.

In addition to the possible usefulness of the shallow simulations, this theorem has implications in computational complexity lower bounds, since this implies that any small improvement in current lower bound approaches would lead to dramatic advances in lower bounds research.

1. INTRODUCTION

The field of computational complexity broadly attempts to understand the limits of computation under certain resource bounds. The main goal of this field is to understand which problems have *efficient solutions* and which problems do not. The resources that are often studied are running time of the algorithm, the space used, the randomness used, number of I/O operations, etc.

In the context of *time complexity*, recall the Boolean class P , containing all decision problems that can be solved by an algorithm that takes polynomial-time in the size of the input. The class P is the class of all problems that we deem *efficiently computable* with respect to time, and we would like to know if certain algorithmic tasks are in P or not. Some examples are the *traveling salesman problem* (TSP) (given a weighted graph G and a parameter k , check if there is a tour to visit all vertices of the graph of total cost at most k) or *satisfiability* (SAT) (where given a Boolean formula, check if there is an assignment to the variables that satisfies the formula). This question is precisely the well-known “ P versus NP ” question and it is widely believed that there are no efficient algorithms for TSP or SAT. Such conjectures can be interpreted as saying that these computational tasks require many “basic elementary operations.” For example, in Boolean circuit complexity, the goal is to understand the minimum number of AND, OR, and NOT gates required to compute a given

Boolean function.

The classes of problems of interest in this paper are inherently algebraic such as integer multiplication, matrix multiplication, etc. For many of these problems, many century/millennia old algorithms have been superseded by faster modern algorithms. These faster algorithms show that it is possible to rearrange basic computational elements in surprising ways to give more efficient solutions. Some excellent examples of such algebraic algorithms are the classic Strassen’s²² faster matrix multiplication algorithm, and Karatsuba’s algorithm¹¹ integer multiplication algorithm. The study of how and when this is possible is the field of *arithmetic complexity* and it can be seen as an algebraic analog of computational complexity theory.

In this field, which is the focus of this article, the main objects of study are formal polynomials over several variables. Specifically, we wish to understand “how hard” certain polynomials are to compute. The most natural way to compute a formal polynomial from the variables x_1, \dots, x_n is via a sequence of operations consisting of additions, subtractions, and multiplications. Such computations can be visualized via *arithmetic circuits*. As seen in Figure 1, we shall allow input wires to a (+) gate to be labeled by scalars to enable computation of arbitrary linear combinations of the inputs. In particular, subtractions may be simulated by labeling the wire with (-1) .

We would like to say that “hard” formal polynomials require “many” such operations to compute them. In this regard, two relevant measures of hardness are the circuit size (the total number of arithmetic operations involved) and depth (the maximum length of a path from the input to the output). Our goal is to understand the optimal complexity, in terms of size and depth, of a given formal polynomial family. Note that a circuit could conceivably be implemented in hardware and the depth of the circuit would correspond to the *latency* or the time needed for the output to appear.

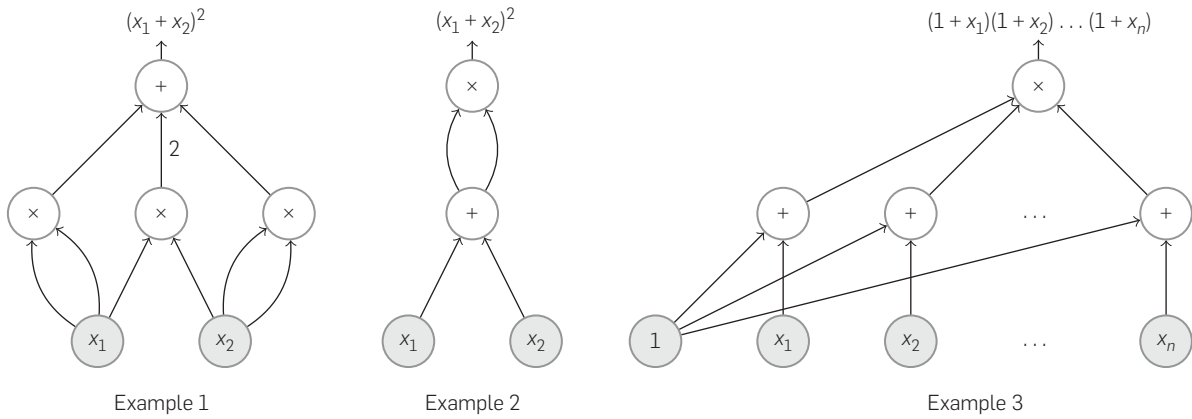
An example of a well-studied formal polynomial family is the *determinant*:

$$\begin{aligned} \text{Det}_d &= \det \begin{bmatrix} x_{11} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{d1} & \cdots & x_{dd} \end{bmatrix} \\ &= \sum_{\sigma \in S_d} \text{sign}(\sigma) \cdot x_{1\sigma(1)} \cdots x_{d\sigma(d)} \end{aligned}$$

where S_d refers to the set of all permutations of d symbols.

The original version of this paper was published in *The 54th Annual Symposium on Foundations of Computer Science (FOCS 2013)*, and would also appear in the *SICOMP Special Issue for FOCS 2013*.

Figure 1. Examples of arithmetic circuits.



Using the well-known (high school) method for computing the determinant, it can be shown Det_d can be computed by $\text{poly}(d)$ -sized arithmetic circuits. A very closely related family of formal polynomials is the *permanent*:

$$\begin{aligned} \text{Perm}_d &= \text{perm} \begin{bmatrix} x_{11} & \dots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{d1} & \dots & x_{dd} \end{bmatrix} \\ &= \sum_{\sigma \in S_d} x_{1\sigma(1)} \dots x_{d\sigma(d)}. \end{aligned}$$

Computing the permanent of a 0/1 matrix is not just NP-hard but can in fact be used to *count* the number of satisfying assignment for any Boolean formula, and count perfect matching in a bipartite graph, and in some other problems involving counting and estimation in statistics.

Although this family looks very similar to the determinant, it is widely believed that any arithmetic circuit computing Perm_d must be of size exponential in d . Showing that the permanent cannot be computed by polynomial-size arithmetic circuits is the biggest challenge in the field of arithmetic circuit complexity.

1.1. Importance of arithmetic circuits

To understand hardness of formal polynomial functions, we first need to define a suitable measure of hardness. Intuitively, a polynomial $f = x_1 \dots x_n$ should be “easy” as this is just a monomial. On the other hand, a polynomial such as $f' = (x_1 + 1) \dots (x_n + 1) = \sum_{S \subseteq [n]} \prod_{i \in S} x_i$ should also be “easy,” despite having exponentially many monomials, as f' is not much different from f . A measure of hardness, that is, robust with respect to such minor transformations is provided by arithmetic circuits. This model is particularly useful if we would like to understand functions like the determinant and permanent, which are very algebraic in nature. Valiant²⁴ introduced two classes of formal polynomials as algebraic analogs of the Boolean classes P and NP. Analogous to the class P that consists of efficiently computable Boolean functions, the class VP is defined as the class of formal polynomials $f(x_1, \dots, x_n)$ with $\text{deg}(f) = \text{poly}(n)$ that can be computed by arithmetic circuits of $\text{poly}(n)$ size.

Analogous to the class NP that consists of Boolean

functions $F(x_1, \dots, x_n)$ that can be expressed as

$$F(x_1, \dots, x_n) = \bigvee_{y \in \{0,1\}^m} G(x_1, \dots, x_n, y_1, \dots, y_m)$$

for some $G \in \text{P}$ and $m = \text{poly}(n)$, the class VNP consists of formal polynomials $f(x_1, \dots, x_n)$ that can be expressed as

$$f(x_1, \dots, x_n) = \sum_{y \in \{0,1\}^m} g(x_1, \dots, x_n, y_1, \dots, y_m)$$

for some formal polynomial $g \in \text{VP}$ and $m = \text{poly}(n)$.

Valiant²⁴ showed that Det_d and Perm_d are in some sense *complete* for the classes VP and VNP, respectively. Thus, showing that permanent cannot be computed by polynomial-size arithmetic circuits would separate the algebraic analog of NP from the algebraic analog of P. Further, if $\text{NP} \not\subseteq \text{P/poly}$, then it can be shown that $\text{VP} \neq \text{VNP}$ over finite fields. Hence, proving $\text{VP} \neq \text{VNP}$ could be thought of as a stepping stone toward proving $\text{P} \neq \text{NP}$.

Another connection to the Boolean world is via the problem of *polynomial identity testing (PIT)*, wherein we are given an arithmetic circuit as input and are required to check if the formal polynomial computed by the circuit is zero or not. Although this algorithmic question has a very natural randomized algorithm, constructing a deterministic algorithm would be major progress.¹⁰ Further, as in the Boolean world, derandomizing PIT has very deep connections to proving arithmetic circuit lower bounds via hardness-randomness tradeoffs.

Thus understanding arithmetic circuits could be a first step toward understanding Boolean circuits. Arithmetic circuits also have a lot of structure which, sometimes, have no analogs in the Boolean world. The most important example of this is the possibility of *depth reduction* for arithmetic circuits.

1.2. The power of shallow circuits

Circuits with low depth correspond to computations which are highly parallel, where edges indicate dependencies in computation. Therefore it is natural to try to minimize the depth of a circuit while allowing the size to increase somewhat. In the case of Boolean circuits, we know that very shallow Boolean circuits are not powerful at all. It is known that constant depth Boolean circuits consisting of \wedge and \vee

gates cannot even compute the parity of n -bits unless they are of exponential size. This important result is now a part of any course or textbook on complexity theory, such as Ref.² The analogous setting in arithmetic circuits to prove lower bounds for constant depth (even depth of three or four) have been challenging!

Valiant et al.²⁵ showed that if a formal polynomial f of degree d can be computed by a circuit of size s then it can in fact be computed by a circuit of depth $O(\log d \cdot \log s)$ and size $s^{O(1)}$. The contrapositive of this depth reduction is that if we can prove super-polynomial-size lower bounds for $O(\log^2 n)$ depth circuits, then we can prove super-polynomial-size lower bounds for general circuits.

Pushing this line of investigation of size-depth tradeoffs further, recent work has considered reduction to circuits of even smaller depth (with gates of unbounded fan-in) at the cost of a super-polynomial increase in size. In this direction, the work of Agrawal and Vinay¹ and a subsequent strengthening by Koiran¹⁴ and Tavenas²³ showed that if an n -variate degree d polynomial f has circuits of size $s = n^{O(1)}$ then f can in fact be computed by depth four circuits of size $n^{O(\sqrt{d})}$. It is worth noting that the trivial computation as a sum of monomials requires size $\binom{n+d}{d} = n^{O(d)}$, and a simple nonconstructive argument shows that most formal polynomials require $n^{O(d)}$ size. In this light, the increase in size by only $n^{O(\sqrt{d})}$ is

Notations. We recall that, $f(n) = \Omega(g(n))$ means that for some constant c , $f(n)$ is greater than $c \cdot g(n)$ for sufficiently large n . Furthermore, $f(n) = \omega(g(n))$ means that for every constant c , $f(n)$ is larger than $c \cdot g(n)$ for sufficiently large n .

still very useful in the context of lower bounds.

This implies that one *merely* needs to prove a (good enough) lower bound for depth four circuits in order to prove lower bounds for arbitrary circuits. For example, if we could show that Perm_d requires depth four circuits of size $2^{\Omega(d^{3/4})}$ then this would immediately imply that Perm_d requires general arithmetic circuits of size $2^{\Omega(d^{3/4})}$ to compute it. Indeed, motivated by this, recent lower bounds for depth four circuits have come very close to the threshold required to separate VP and VNP.

The following is a summary of the known depth reduction results:

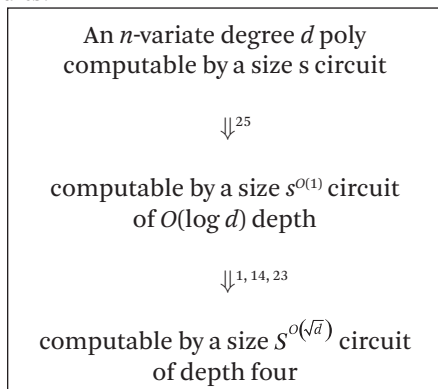
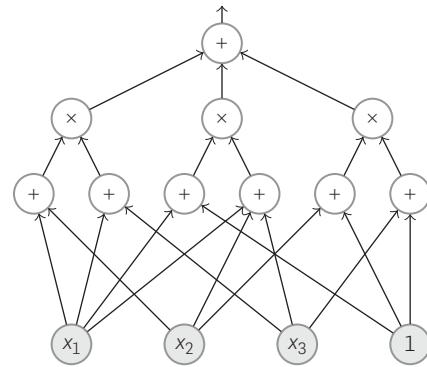


Figure 2.. A depth three circuit.



1.3. Lower bounds for shallow circuits

Depth three circuits. Progress in proving lower bounds for arithmetic circuits has been admittedly slow. This is generally considered to be one of the most challenging problems in computer science. The difficulty of the problem has led researchers to focus on natural subclasses of arithmetic circuits. Bounded depth circuits being one such natural subclass has received a lot of attention. The class of depth three arithmetic circuits, also denoted as $\Sigma\Pi\Sigma$ circuits, have been intensely investigated as it is the shallowest nontrivial subclass. Such a circuit C , as shown in Figure 2, computes a formal polynomial of the form

$$C(x_1, \dots, x_n) = \sum_{i=1}^s \prod_{j=1}^{d_i} \ell_{ij}(x_1, \dots, x_n),$$

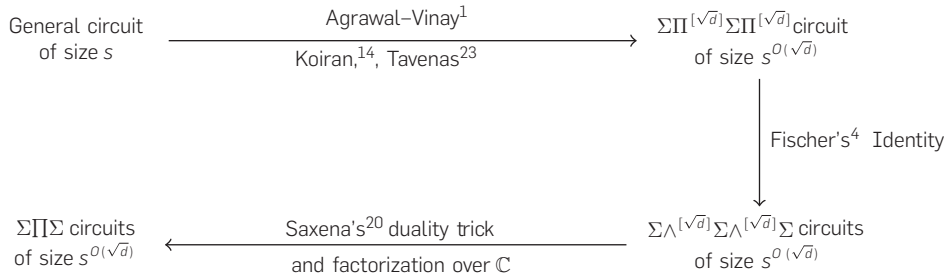
where each $\ell_{ij}(\mathbf{x})$ is a linear polynomial over the input variables. $\Sigma\Pi\Sigma$ circuits arise naturally in the investigation of the complexity of polynomial multiplication and matrix multiplication. Moreover, the optimal formula/circuit for some well-known families of formal polynomials are in fact depth three circuits. In particular, the best known circuit for computing Perm_d is known as Ryser's formula¹⁹:

$$\text{Perm}_d = \sum_{S \subseteq [d]} (-1)^{d-|S|} \prod_{i=1}^d \left(\sum_{j \in S} x_{ij} \right),$$

which is a depth three circuit of size $O(d^2 \cdot 2^d)$. Ryser's formula is another example of a nontrivial speedup of algebraic computation.

Quite a few lower bounds have been obtained for restrictions of depth three circuits. Nisan and Wigderson¹⁸ studied depth three circuits under the restriction of *homogeneity*. A homogeneous circuit is one where the degree of all intermediate computations are bounded by the degree of the output polynomial. Nisan and Wigderson¹⁸ showed that over any field \mathbb{F} , any homogeneous $\Sigma\Pi\Sigma$ circuit computing the determinant Det_d must be of size $2^{\Omega(d)}$. They also gave a concrete example to show that nonhomogeneous depth three circuits can be exponentially more powerful than homogeneous depth three circuits. For general circuits, however, homogeneity may be assumed without loss of generality. But the same cannot be said about very shallow circuits.

Figure 3. Reduction to depth three.



Grigoriev and Karpinski,⁶ and Grigoriev and Razborov⁷ showed that any $\Sigma\Pi\Pi$ arithmetic circuit over any *fixed finite field* computing Det_d must be of size at least $2^{\Omega(d)}$, thus also implying that any $\Sigma\Pi\Pi$ arithmetic circuit *over integers* computing Det_d must be of size at least $2^{\Omega(d)}$.

Other restrictions of $\Sigma\Pi\Pi$ circuits have been studied but till date, the best known lower bound in the general $\Sigma\Pi\Pi$ case is the quadratic lower bound due to Shpilka and Wigderson.²¹ Wigderson²⁶ highlighted this frontier in arithmetic complexity by concluding his plenary talk on “P, NP, and mathematics” at ICM 2006 with the problem of proving super-polynomial-size lower bounds for $\Sigma\Pi\Pi$ circuits computing Det_d . In fact, prior to our work, it was believed that any depth three circuit for Det_d should be of size $2^{\Omega(d)}$.

Depth four circuits. For depth four circuits (also called $\Sigma\Pi\Pi\Pi$ circuits), the depth-reduction results of Agrawal and Vinay,¹ with subsequent strengthening by Koiran¹⁴ and Tavenas²³ show that an n -variate degree d formal polynomial computed by a size s arithmetic circuit can be equivalently simulated by a depth four circuit of size $s^{O(\sqrt{d})}$. In fact, the resulting depth four circuit has the property that all multiplication gates have fan-in about \sqrt{d} . This implies that if we obtain an $n^{\omega(\sqrt{d})}$ lower bound for the size of such fan-in restricted depth four circuits computing an explicit n -variate degree d formal polynomial, then we obtain super-polynomial-size lower bounds for general circuits! Thus, in essence, such depth four circuits are almost as hard as general circuits. Agrawal and Vinay¹ referred to this phenomenon as “the chasm at depth four.”

The first breakthrough was obtained⁸ by presenting a lower bound of $2^{\Omega(\sqrt{d})}$ for Det_d and Perm_d using a technique called “shifted partial derivatives.” This is a technique built over a known technique (introduced by Nisan and Wigderson¹⁸) called the *partial derivative method*, and enhanced with some intuition from algebraic-geometry.

Using shifted partial derivatives again, this bound was subsequently improved^{5, 13, 15} to $n^{\Omega(\sqrt{d})}$, and a flurry of results gave more lower bounds for depth four circuits using very similar techniques with matching upper bounds! In fact, recent results^{12, 16} have obtained an $n^{\Omega(\sqrt{d})}$ for depth four circuits even when there is no restriction on the fan-in, and once again with matching upper bounds!

To separate VP and VNP, one “merely” needs to study depth four circuits and improve the known lower bounds from $n^{\Omega(\sqrt{d})}$ to $n^{\omega(\sqrt{d})}$.

How could one not be optimistic about this!

Some remarks. It is worth pointing out that all the lower bounds recently obtained via the *shifted partial derivatives* require an extremely delicate setting of parameters, with very much involved calculations on binomial coefficients (often working with the second-order terms in the Stirling’s approximation of $n!$). In fact, some of these proofs have forced researchers to *design*^a formal polynomials (based on combinatorial objects called Nisan–Wigderson¹⁷ designs) with the sole purpose of making these calculations amenable.

1.4. A new depth reduction

We present a surprising structural result about arithmetic circuits over \mathbb{Q} that was obtained as a consequence of attempting to break the $n^{\Omega(\sqrt{d})}$ barrier.

THEOREM 1. *Any n -variate degree d formal polynomial f that can be computed by size s arithmetic circuits over \mathbb{Q} can be equivalently computed by a depth three circuit of size $s^{O(\sqrt{d})}$.*

This effectively brings the *chasm* down to depth three. It is worth noting that the blow-up from s to $s^{O(\sqrt{d})}$ is the same asymptotics we get in the earlier reduction to depth four circuits. To give a better sense of why such a result was surprising (at least to us), we mention a few properties of the resulting depth three circuits.

- Although Det_d has a polynomial-sized circuit, albeit of large depth, the lower bound of Grigoriev and Karpinski’s⁶ show that depth three circuits computing Det_d *that only uses integer constants in them* must be of size $2^{\Omega(d)}$. This means that the simulation in Theorem 1 must involve fractions in the circuit. Thus, the proof of the theorem must be very algebraic in nature and depend very strongly on the underlying field of constants. All prior depth reductions were field independent, as they were very syntactic in nature. This depth reduction must therefore be very different.
- As mentioned earlier, we knew of a $2^{\Omega(d)}$ lower bound for depth three circuits over the integers for Det_d . On the algorithmic side, nothing significantly better than writing Det_d as a sum of $d! = d^{\Theta(d)}$ monomials was known. It was strange that the best known $\Sigma\Pi\Pi$ circuit for Perm_d was smaller than the best known $\Sigma\Pi\Pi$ circuit for Det_d over \mathbb{Q} ! Fortunately, our depth reduction yields a

^a Pun intended.

depth three circuit of size $d^{o(\sqrt{d})}$ computing Det_d over \mathbb{Q} .

- Lower bounds for homogeneous depth three circuits already imply that the circuit obtained from such a depth reduction must necessarily be nonhomogeneous. Indeed, the resulting depth three circuits in our depth reduction are *heavily nonhomogeneous* in the sense that although the output polynomial is of degree d , intermediate gates compute formal polynomials of degree $n^{o(\sqrt{d})}$ with the higher degree terms canceling out eventually!

The proof of Theorem 1 is rather quite short, and is obtained by using two known transformations (see Figure 3). But more than the proof, we believe that the train of thought that led to this theorem is more insightful. In this article, we shall present the result from the perspective of attempting to obtain a lower bound better than $n^{o(\sqrt{d})}$ for depth four circuits.

2. TOWARD BETTER LOWER BOUNDS FOR DEPTH FOUR

We now begin to describe the depth reduction. The road map we shall be following is given in Figure 3, and this section would handle the first two arrows of Figure 3. From the depth reduction results of Agrawal and Vinay,¹ Koiran,¹⁴ and Tavenas,²³ we have the first arrow of Figure 3, that is, any polynomial-size computation can be simulated by a depth four computation of not-much-larger size. This implies that the problem of prov-

Problem 1. Find an explicit n -variate degree d formal polynomial f such that any expression of the form

$$f = \sum_{i=1}^s Q_{i1} \cdots Q_{i\sqrt{d}}$$

with $\deg Q_{ij} \leq \sqrt{d}$ must have $s = n^{\omega(\sqrt{d})}$.

ing explicit super-polynomial-size lower bounds for arithmetic circuits reduces to the following question^b:

The expression on the RHS is just a depth four circuit with all multiplication fan-ins bounded by \sqrt{d} . We shall call such circuits $\Sigma\Pi^{[\sqrt{d}]}_{\Sigma\Pi^{[\sqrt{d}]}}$ circuits.

Problem 2. Find an explicit n -variate degree d formal polynomial f such that any expression of the form

$$f = \sum_{i=1}^s Q_i^{\wedge r}$$

with $\deg Q_i \leq \sqrt{d}$ must have $s = n^{\omega(\sqrt{d})}$.

We could first attempt solving the following “simpler” question:

We shall call expressions like the above RHS as $\Sigma\wedge^{[\sqrt{d}]}_{\Sigma\Pi^{[\sqrt{d}]}}$ circuits, where \wedge refers to the exponentiation.

It turns out that a solution to Problem 2 directly implies

^b Technically, the multiplications fan-ins ought to be at most $15\sqrt{d}$ but we shall take this liberty for the sake of better presentation.

a solution for Problem 1. The reason for this is that any product can be rewritten as a sum of suitable powers, just like $xy = ((x+y)^2 - x^2 - y^2)/2$. The following identity can be obtained by a straightforward application of the inclusion-exclusion principle.

$$r! \cdot Q_1 \cdots Q_r = \sum_{s \subseteq [r]} \left(\sum_{i \in s} Q_i \right)^r \cdot (-1)^{r-|s|}.$$

As one might suspect from the similarity, the above equation is a special case of Ryser’s formula. We shall call this Fischer’s identity,⁴ although he used a different identity to achieve the same transformation. Fischer’s identity can be thought of as converting a \times gate to a $\Sigma \wedge \Sigma$ circuit (Figure 4). Note that this identity is useful only as long as we can divide by $r!$ on both sides. Thus, if we are working only over integers, or over strange algebraic structures (such as fields of small characteristic) where $r! = 0$, we cannot apply the above transformation. It is very important that this division by $r!$ is allowed, and this is the also why this transformation cannot work if the resulting circuit must only involve integer constants.

Thus, if f is a formal polynomial admitting an expression of the form

$$f = \sum_{i=1}^s Q_{i1} \cdots Q_{i\sqrt{d}}$$

with $\deg Q_{ij} \leq \sqrt{d}$, then f also admits an expression of the form

$$f = \sum_{i=1}^{s'} Q_i^{\wedge r},$$

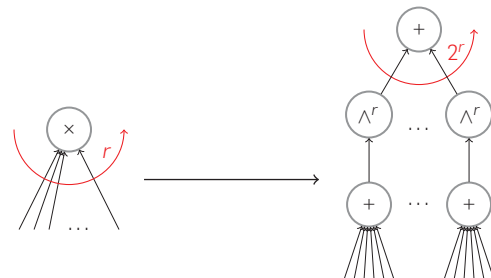
where $Q_i \leq \sqrt{d}$ and $s' \leq s \cdot 2^{\sqrt{d}}$. Hence, showing $s' = n^{\omega(\sqrt{d})}$ would immediately imply that $s = n^{\omega(\sqrt{d})}$. In other words, solving Problem 2 immediately yields a solution to Problem 1.

Just like we converted the top Π layer to exponentiations, we could also apply Fischer’s identity to the bottom multiplication gates also. Since each Q_i is a formal polynomial of degree at most \sqrt{d} , the total number of monomials in any Q_i is at most $n^{\sqrt{d}}$. Applying Fischer’s identity to those monomials as well yields an expression of the form

$$f = \sum_{i=1}^{s'} (\ell_{i1}^{\wedge r} + \cdots + \ell_{ir}^{\wedge r})^{\wedge \sqrt{d}},$$

where each ℓ_{ij} is a linear polynomial and $r \leq n^{o(\sqrt{d})}$. We shall call such expressions analogously as $\Sigma\wedge^{[\sqrt{d}]}_{\Sigma\wedge^{[\sqrt{d}]}}_{\Sigma}$ circuits. The above discussion can therefore be summarized as follows.

Figure 4. Transformation via Fischer’s identity.



LEMMA 2. *Let f be an n -variate degree d formal polynomial that can be computed by size s arithmetic circuits over \mathbb{Q} . Then, f can be equivalently computed by a $\Sigma \wedge^{[\sqrt{d}]} \Sigma \wedge^{[\sqrt{d}]} \Sigma$ circuit over \mathbb{Q} of size $s^{o(\sqrt{d})}$.*

Conversely, if f requires $\Sigma \wedge^{[\sqrt{d}]} \Sigma \wedge^{[\sqrt{d}]} \Sigma$ circuits of size $n^{\omega(\sqrt{d})}$, then f cannot be computed by polynomial-sized arithmetic circuits.

The only thing left to complete in our road map Figure 3 is the last arrow. Before we describe the last step, we shall present some intuition about the mysterious intermediate model of $\Sigma \wedge \Sigma \wedge \Sigma$ circuits that we went to. However, readers who are curious to know how the last step of the transformation works may jump to Section 4.1. For more intuition on $\Sigma \wedge \Sigma \wedge \Sigma$ circuits, we shall keep this on hold and address another lower bound question namely lower bound for depth three circuits.

3. TOWARD LOWER BOUNDS FOR DEPTH THREE CIRCUITS

A depth three circuit computes a formal polynomial of the form

$$f = \sum_{i=1}^s \ell_{i1} \cdots \ell_{iD},$$

where each ℓ_{ij} is a linear polynomial. If the circuit is homogeneous, then $D \leq \deg f$ but we shall be interested in nonhomogeneous depth three circuits and hence D could potentially be much larger than $\deg f$. Using standard tricks, we may assume that f is a homogeneous degree d formal polynomial and we have an expression of the form

$$f = \sum_{i=1}^s \alpha_i (1 + \ell_{i1}) \cdots (1 + \ell_{iD}), \quad (1)$$

where each $\alpha_i \in \mathbb{F}$; and each ℓ_{ij} is a homogeneous linear polynomial. By collecting degree d terms from the RHS, therefore follows that f can be written as

$$f = \sum_{i=1}^s \alpha_i \cdot \text{SYM}_d(\ell_{i1}, \dots, \ell_{iD}),$$

where $\text{SYM}_d(y_1, \dots, y_D)$ is the elementary symmetric polynomial of degree d defined as

$$\text{SYM}_d(y_1, \dots, y_D) = \sum_{1 \leq i_1 < \dots < i_d \leq D} y_{i_1} \cdots y_{i_d}.$$

These elementary symmetric polynomials can be expressed in terms of the *power symmetric polynomials*, defined as

$$P_i(y_1, \dots, y_D) = y_1^i + \dots + y_D^i,$$

via the well-known Newton identities:

$$\begin{aligned} \text{SYM}_d &= \frac{1}{d!} \begin{vmatrix} P_1 & 1 & & & \\ P_2 & P_1 & 2 & & \\ \vdots & \vdots & \vdots & \ddots & \\ P_{d-1} & P_{d-2} & \dots & P_1 & d-1 \\ P_d & P_{d-1} & \dots & P_2 & P_1 \end{vmatrix} \\ &= \sum_{\substack{a_1, \dots, a_d \\ \sum i \cdot a_i = d}} c_a \cdot P_1^{a_1} \cdots P_d^{a_d} \end{aligned}$$

where each c_a is in \mathbb{Q} . The number of summands on the RHS is bounded by the number of partitions of d , which is at most $2^{o(\sqrt{d})}$ by estimates of Hardy and Ramanujan.⁹ Thus, the above equation is a $\Sigma \Pi \Sigma \wedge$ circuit of size $2^{o(\sqrt{d})}$ computing SYM_d .

To convert this to a $\Sigma \wedge \Sigma \wedge \Sigma$ circuit, we could use Fischer's identity again. At first sight, it appears as though this would yield a blow up of 2^d as some of the product gates could have fan-in d . However, notice that the sum is over a_i 's satisfying $\sum i \cdot a_i = d$. Hence, there can be at most $O(\sqrt{d})$ of the a_i 's that are nonzero. By looking at Fischer's identity applied on $P_1^{a_1} \cdots P_d^{a_d}$ more carefully, we see that it uses at most $(1+a_1) \cdots (1+a_d) \leq d^{o(\sqrt{d})}$ distinct linear powers instead of the naïve bound of 2^d . This fact of expressing any degree d monomial over m variables as a $\Sigma \wedge \Sigma$ circuit of size $d^{o(m)}$ was also observed by Ellison.³

This results in an $\Sigma \wedge \Sigma \wedge \Sigma$ circuit for $\text{SYM}_d(y_1, \dots, y_D)$ of size $\text{poly}(D) \cdot d^{o(\sqrt{d})}$. Hence, any formal polynomial expressible as in (1) can be equivalently expressed as homogeneous $\Sigma \wedge \Sigma \wedge \Sigma$ circuit of size $d^{o(\sqrt{d})} \cdot \text{poly}(D, s, n)$. That is, if f admits a poly-sized depth three circuit, then f also admits a homogeneous $\Sigma \wedge \Sigma \wedge \Sigma$ circuit of size $d^{o(\sqrt{d})} \cdot \text{poly}(n)$. The following lemma summarizes this discussion.

LEMMA 3. *Let f be an n -variate degree d formal polynomial, that is, computable by depth three circuit of size s over \mathbb{Q} . Then, f is equivalently computable by a homogeneous $\Sigma \wedge \Sigma \wedge \Sigma$ circuit of size $d^{o(\sqrt{d})} \cdot \text{poly}(s)$.*

Conversely, if f requires $\Sigma \wedge \Sigma \wedge \Sigma$ circuits of size $n^{\omega(\sqrt{d})}$ over \mathbb{Q} to compute it, then f requires depth three circuits of size $n^{\omega(\sqrt{d})}$.

4. PUTTING THEM TOGETHER

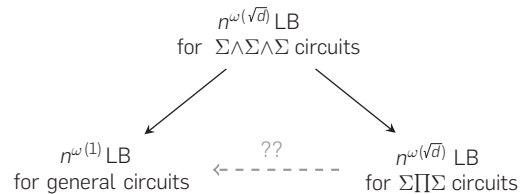
The two lemmas together weave an interesting picture (Figure 5). On one hand, Lemma 2 states that a lower bound of $n^{\omega(\sqrt{d})}$ for $\Sigma \wedge \Sigma \wedge \Sigma$ circuits would yield a super-polynomial-size lower bound for general arithmetic circuits. On the other, Lemma 3 states that an $n^{\omega(\sqrt{d})}$ lower bound for $\Sigma \wedge \Sigma \wedge \Sigma$ circuits would yield a lower bound of $n^{\omega(\sqrt{d})}$ for depth three circuits.

Could this just be a coincidence? Or, is it the case that any poly-sized arithmetic circuit can be equivalently expressed as a depth three circuit of size $n^{\omega(\sqrt{d})}$ over \mathbb{Q} ? As it turns out, there is indeed a depth reduction to convert any arithmetic circuit to a not-too-large depth three circuit over \mathbb{Q} .

4.1. The final piece

To complete the picture, it suffices to show how a $\wedge \Sigma \wedge$ term of the form $(y_1^a + \dots + y_s^a)^b$ can be expressed as a depth three circuit. This would immediately imply a way to express any $\Sigma \wedge \Sigma \wedge \Sigma$ circuit as a $\Sigma \Pi \Sigma$. The following result of

Figure 5. Power of $\Sigma \wedge \Sigma \wedge \Sigma$ circuits.



Saxena²⁰ turns out to be exactly what we need for this transformation. A similar transformation was also discovered earlier by Shpilka and Wigderson²¹ but statement below is from Ref.²⁰

LEMMA 4 (SAXENA'S²⁰ DUALITY TRICK). *There exist univariate formal polynomials f_{ij} 's of degree at most b such that*

$$(z_1 + \dots + z_s)^b = \sum_{i=1}^{sb+1} f_{i1}(z_1) \dots f_{is}(z_s).$$

It is worth noting that the degree of each term on the RHS is sb , whereas the LHS just has degree b . This is the place where nonhomogeneity is introduced. Applying Lemma 4 for $(y_1^a + \dots + y_s^a)^b$ gives

$$\begin{aligned} (y_1^a + \dots + y_s^a)^b &= \sum_{i=1}^{sb+1} \prod_{j=1}^s f_{ij}(y_j^a) \\ &= \sum_{i=1}^{sb+1} \prod_{j=1}^s \tilde{f}_{ij}(y_j), \end{aligned}$$

where $\tilde{f}_{ij}(y) = f_{ij}(y^a)$. Since each $\tilde{f}_{ij}(y)$ is a univariate polynomial, it can be factorized completely over the \mathbb{C} , the field of complex numbers. Hence, if $f_{ij}(y) = \prod_k (y - \zeta_{ijk})$, then we get

$$\begin{aligned} (y_1^a + \dots + y_s^a)^b &= \sum_{i=1}^{sb+1} \prod_{j=1}^s \tilde{f}_{ij}(y_j) \\ &= \sum_{i=1}^{sb+1} \prod_{j=1}^s \prod_{k=1}^b (y_j - \zeta_{ijk}), \end{aligned}$$

which is a depth three circuit! Thus, $(y_1^a + \dots + y_s^a)$ can be expressed as a depth three circuit of size $\text{poly}(s, a, b)$ over \mathbb{C} . Hence, any $\Sigma \wedge \Sigma \wedge \Sigma$ circuit of size s can also be expressed as a depth three circuit of size $\text{poly}(s, a, b)$ over \mathbb{C} . As mentioned earlier, the degree of intermediate computations in the resulting depth three circuit is $s \cdot ab$ although the formal polynomial computed is of degree just ab .

This already yields the depth reduction over complex numbers. Lemma 2 states that any formal polynomial of degree d over n variables that can be computed by an arithmetic circuit of size s can be computed by an $\Sigma \wedge \Sigma \wedge \Sigma$ circuit of size $s' = s^{O(\sqrt{d})}$. By what we just noted, this can in-turn be simulated by a depth three circuit over \mathbb{C} of size $\text{poly}(s') = s^{O(\sqrt{d})}$. The degree of intermediate computations in the resulting depth three circuit is $s' \cdot d = s^{O(\sqrt{d})}$, which makes this a heavily nonhomogeneous circuit.

Although this depth reduction was done over the field of complex numbers, a little more effort can yield a depth three circuit over \mathbb{Q} as well. This is a consequence of a more general transformation to convert any circuit C of size s computing a formal polynomial $f \in \mathbb{Q}[x_1, \dots, x_n]$ using constants from an extension field of *low degree* into an equivalent circuit C' of not-much-larger size, thus yielding Theorem 1.

Further, the transformation is also efficient in the sense that the time required for this transformation is polynomial-time in the output size.

5. SOME CONSEQUENCES

An immediate consequence of such a depth reduction is new constructions of depth three circuits for known efficiently computable formal polynomials. The permanent can be computed by a depth three circuit of size $2^{O(d)}$ via Ryser's formula.¹⁹ Surprisingly, we knew of no such circuit for the determinant polynomial. In fact, nothing significantly better than writing Det_d as a sum of $d! = d^{O(d)}$ terms was known. With this depth reduction, we now have a depth three circuit of size $d^{O(\sqrt{d})}$ for Det_d over \mathbb{Q} .

Another immediate consequence is that strong enough lower bounds (i.e., $n^{\omega(\sqrt{d})}$) for $\Sigma\Pi\Sigma$ circuits would yield super-polynomial-size lower bounds (i.e., $n^{\omega(1)}$) for general arithmetic circuits. It is, however, unclear if *nonhomogeneous* depth three circuit lower bounds are easier than homogenous depth four circuit lower bounds. In the past, homogeneity has played a crucial role in devising techniques for lower bounds.

The third consequence of such a depth reduction is implications to PIT. As mentioned earlier, PIT is the problem of checking if the formal polynomial computed by a given circuit C of size s is zero or not. The goal is to do this deterministically in $\text{poly}(s)$ time, which is like an algebraic analog of showing $P = \text{co-RP}$. A stronger form of PIT is the problem of *Blackbox PIT* where we are only given oracle access to the circuit C . In the blackbox model, we wish to construct an explicit *hitting set* of $\text{poly}(s)$ size, that is, a list of evaluations such that any nonzero circuit of size s is guaranteed to evaluate to a nonzero value on one of the points.

A consequence of this depth reduction is that constructing an explicit $\text{poly}(s)$ -sized hitting set for depth three circuits of size s implies an explicit hitting set of size $s^{O(\log s)}$ for general arithmetic circuits of size s . This is exactly along the same lines as in the depth reduction of Agrawal and Vinay¹ for depth four circuits.

6. CONCLUSION

If the goal is to prove super-polynomial-size lower bounds for arithmetic circuits, we can focus our attention on improving the current lower bound (even slightly) for any of the following three models:

- $\Sigma\Pi^{[\sqrt{d}]} \Sigma\Pi^{[\sqrt{d}]}$ circuits
- $\Sigma \wedge^{[\sqrt{d}]} \Sigma\Pi^{[\sqrt{d}]}$ circuits
- $\Sigma\Pi\Sigma$ circuits

A natural question here is—which of the above three models should we attack?

Homogeneity has played a very important role in the past lower bounds and proving lower bounds for such heavily nonhomogeneous depth three circuits appears daunting. We believe the model to focus our attention on is the class of $\Sigma \wedge^{[\sqrt{d}]} \Sigma\Pi^{[\sqrt{d}]}$ circuits. We strongly believe that this is the simplest form for which to prove super-polynomial arithmetic circuit lower bounds. We state the problem again below to emphasize our point.

The Problem. Find an explicit n -variate degree d formal polynomial f such that any expression of the form

$$f = \sum_{i=1}^s Q_i^{\sqrt{d}}$$

with $\deg Q_i \leq \sqrt{d}$ must have $S = n^{\omega(\sqrt{d})}$.

We have already obtained lower bounds of $n^{\omega(\sqrt{d})}$. How hard can it be to achieve $n^{\omega(\sqrt{d})}$, right? □

References

1. Agrawal, M., Vinay, V. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)* (2008), 67–75.
2. Arora, S., Barak, B. *Computational Complexity: A Modern Approach*, 1st edn. Cambridge University Press, New York, NY, USA, 2009.
3. Ellison, W. A 'waring's problem' for homogeneous forms. *Proc. Camb. Philos. Soc.* 65 (1969), 663–672.
4. Fischer, I. Sums of like powers of multivariate linear forms. *Math. Mag.* 67, 1 (1994), 59–61.
5. Fournier, H., Limaye, N., Malod, G., Srinivasan, S. Lower bounds for depth 4 formulas computing iterated matrix multiplication. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)* (2014), 128–135.
6. Grigoriev, D., Karpinski, M. An exponential lower bound for depth 3 arithmetic circuits. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC 1998)* (1998), 577–582.
7. Grigoriev, D., Razborov, A.A. Exponential lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. *Appl. Algebra Eng. Commun. Comput.* 10, 6 (2000), 465–487.
8. Gupta, A., Kamath, P., Kayal, N., Saptharishi, R. Approaching the chasm at depth four. *J. ACM* 61, 6 (2014), 33:1–33:16. Preliminary version in the *28th Annual IEEE Conference on Computational Complexity (CCC 2013)*.
9. Hardy, G.H., Ramanujan, S. Asymptotic formulæ in combinatory analysis. *Proc. London Math. Soc.* s2–17, 1 (1918), 75–115.
10. Kabanets, V., Impagliazzo, R. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.* 13, 1–2 (2004), 1–46.
11. Karatsuba, A., Ofman, Y. Multiplication of multidigit numbers on automata. *Engl. Transl. Soviet Phys. Dokl.* 7 (1963), 595–596.
12. Kayal, N., Limaye, N., Saha, C., Srinivasan, S. An exponential lower bound for homogeneous depth four arithmetic circuits. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)* (2014).
13. Kayal, N., Saha, C., Saptharishi, R. A super-polynomial lower bound for regular arithmetic formulas. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)* (2014), 146–153.
14. Koiran, P. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.* 448 (2012), 56–65.
15. Kumar, M., Saraf, S. The limits of depth reduction for arithmetic formulas: It's all about the top

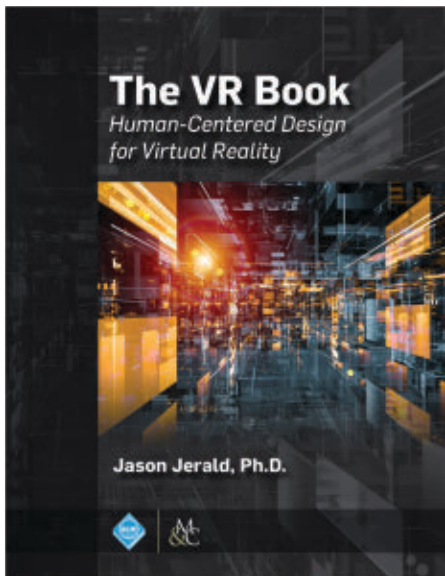
- fan-in. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)* (2014), 136–145.
16. Kumar, M., Saraf, S. On the power of homogeneous depth 4 arithmetic circuits. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)* (2014).
17. Nisan, N., Wigderson, A. Hardness vs randomness. *J. Comput. Syst. Sci.* 49, 2 (1994), 149–167.
18. Nisan, N., Wigderson, A. Lower bounds on arithmetic circuits via partial derivatives. *Comput. Complex.* 6, 3 (1997), 217–234.
19. Ryser, H.J. Combinatorial mathematics. *Math. Assoc. Am.* 14 (1963).
20. Saxena, N. Diagonal circuit identity testing and lower bounds. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP 2008)* (2008), 60–71.
21. Shpilka, A., Wigderson, A. Depth-3 arithmetic circuits over fields of characteristic zero. *Comput. Complex.* 10, 1 (2001), 1–27.
22. Strassen, V. Gaussian elimination is not optimal. *Numer. Math.* 13, 3 (1969), 354–356.
23. Tavenas, S. Improved bounds for reduction to depth 4 and 3. In *Proceedings of the 38th International Symposium on the Mathematical Foundations of Computer Science (MFCS 2013)* (2013).
24. Valiant, L.G. Completeness classes in algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)* (1979), 249–261.
25. Valiant, L.G., Skyum, S., Berkowitz, S., Rackoff, C. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.* 12, 4 (1983), 641–644.
26. Wigderson, A. P, NP and mathematics – A computational complexity perspective. In M. Sanz-Solé, J. Soria, J.L. Varona and J. Verdera, eds. *Proceedings of the ICM 06 (Madrid)*, Volume 1 (2007). EMS Publishing House, Zurich, 665–712.

Ankit Gupta (ankitgupta@cmi.ac.in), Chennai Mathematical Institute, Kelambakkam, India.

Prithish Kamath (prithish@mit.edu), Massachusetts Institute of Technology, Cambridge, MA.

Neeraj Kayal and Ramprasad Saptharishi (neeraka@microsoft.com, ramprasad@cmi.ac.in), Microsoft Research, Bangalore, India.

Copyright held by the authors.
Publication rights licensed to ACM. \$15.00



Without a clear understanding of the human side of virtual reality, the experience will always fail.

“Dr. Jerald has recognized a great need in our community and filled it. The VR Book is a scholarly and comprehensive treatment of the user interface dynamics surrounding the development and application of virtual reality. I have made it a required reading for my students and research colleagues. Well done!”

- Professor Tom Furness, University of Washington
VR Pioneer and Founder of HIT Lab International and the Virtual World Society



ISBN: 978-1-970001-12-9 DOI: 10.1145/2792790
<http://books.acm.org>
<http://www.morganclaypoolpublishers.com/vr>

Docomo Innovations, Inc. **Principal/Senior Software Engineer -** **Deep Learning**

DOCOMO Innovations, Inc. (DII) is a subsidiary of NTT DOCOMO, Inc. in Japan, a world leader in mobile operations and a growing provider of comprehensive mobile services.

Open Services Innovation Group (OSIG) develops services using state-of-the-art technologies. We are using approaches from Machine Learning, Artificial Intelligence to build next-generation intelligent systems and services for data analysis, text analytics, machine translation and other problems.

We are looking for a Principal or Senior Software Engineer to help us improve machine learning and deep learning capabilities. The position is based in our Palo Alto office.

To view complete job details and to apply, visit: <https://docomoinnovations.applicantstack.com/x/detail/a2z4hfhszcg/aaam>

The University Of Western Ontario **Tier I Canada Research Chair in Data Science**

The Departments of Computer Science and Statistical & Actuarial Sciences are pleased to announce a search for a Tier I Canada Research Chair in Data Science for a jointly appointed Tenured Appointment at the rank of Associate Professor or Full Professor. The rank will be commensurate with the successful applicant's qualifications and experience. The candidate must have a Ph.D. in Computer Science, Statistics, Mathematics, Engineering or a related field and a demonstrated record of independent research accomplishments. The starting date will be July 1, 2018, or as negotiated.

In accordance with the regulations set for Tier 1 Canada Research Chairs (<http://www.chairs-chaires.gc.ca>), the successful candidate will be an outstanding and innovative researcher whose accomplishments have made a major impact; be recognized internationally as a leader in their field; and have a superior record of attracting and supervising graduate students and postdoctoral fellows. The successful candidate will provide leadership in research, foster collaborations with industry, promote interdisciplinary scholarship, increase knowledge mobilization and societal benefits, and catalyze, in collaboration with faculty and staff, the development of interdisciplinary graduate and undergraduate programs in Data Science and Analytics. The preferred candidate will demonstrate leadership, collegiality and stra-

tegic vision through collaboration with existing Computer Science and Statistical & Actuarial Sciences faculty members to foster existing and build new links between the departments. The candidate must propose an innovative research program of the highest quality that will attract excellent trainees, students, and future researchers and will have displayed an interest in working on problems of industrial and/or societal relevance.

Data Science is a priority research and teaching focus at Western University. The preferred candidate will have an internationally peer-recognized record of research in data science/analytics. These areas might include one or more of the following: Machine learning, statistical analysis and calibration of big data sets and models, data-driven and data intensive systems, Bayesian modeling, stream data mining, spatio-temporal analysis, graph analytics,

text analytics, and visualization and analysis of high-dimensional data.

General information about the University can be found at <http://www.uwo.ca/> London boasts an international airport, galleries, theatre, music and sporting events (see <http://www.goodmovelondon.com/>)

Candidates should submit a curriculum vitae, one-page teaching statement, one-page statement listing experience or interest in professional and/or leadership programs, two-page research plan, and contact details of at least three referees to:

Dianne McFadzean Department of Computer Science, Western University, London, Ontario N6A 5B7 dmcfadze@uwo.ca

Applications will be considered starting September 30, 2017 and will continue until the position is filled. See the full advertisement at <http://www.uwo.ca/facultyrelations/faculty/crc-tier-1---data-science---advertisement-2.pdf>



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/ issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact:

acmm mediasales@acm.org

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://jobs.acm.org>

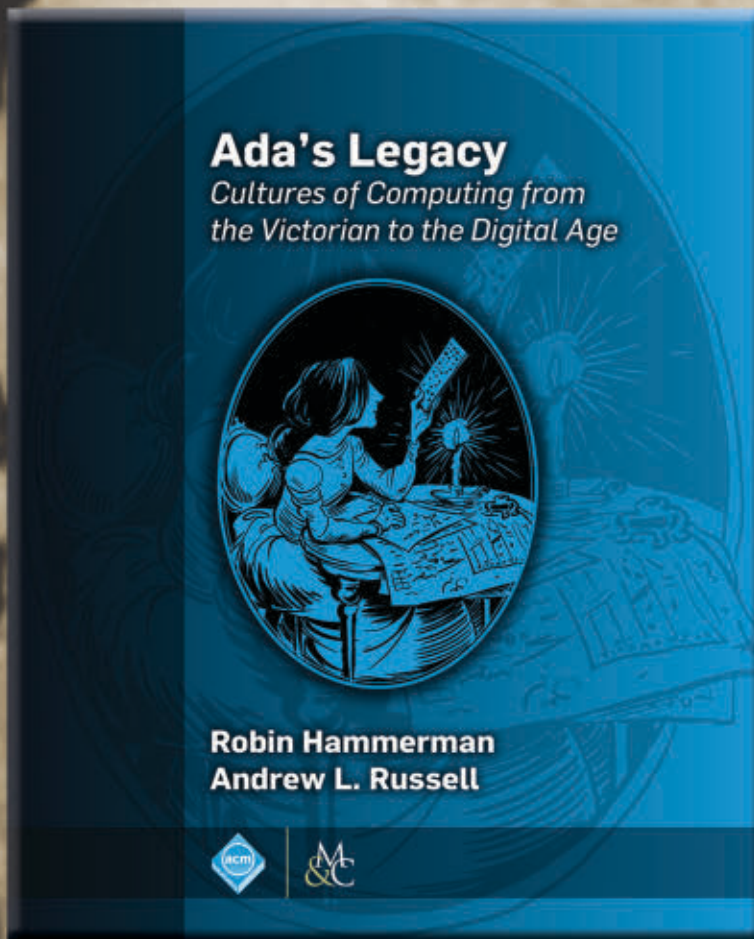
Ads are listed for a period of 30 days.

**For More Information Contact:
ACM Media Sales
at 212-626-0686 or
acmm mediasales@acm.org**

Robin Hammerman and Andrew L. Russell

Ada's Legacy

Cultures of Computing from the
Victorian to the Digital Age



INSPIRING MINDS FOR 200 YEARS

Ada's Legacy illustrates the depth and diversity of writers, things, and makers who have been inspired by Ada Lovelace, the English mathematician and writer.

The volume commemorates the bicentennial of Ada's birth in December 1815, celebrating her many achievements as well as the impact of her work which reverberated widely since the late 19th century. This is a unique contribution to a resurgence in Lovelace scholarship, thanks to the expanding influence of women in science, technology, engineering and mathematics.

ACM Books is a new series of high quality books for the computer science community, published by the Association for Computing Machinery with Morgan & Claypool Publishers.



ISBN: 978-1-97000-148-8 DOI: 10.1145/2809523

<http://books.acm.org>

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

[CONTINUED FROM P. 104] tems for the physics experiments, but he found an excuse to do it as a way of kicking the tires on a new computer, the NeXT machine, which Steve Jobs had made when he left Apple.

The comment Mike appended to your proposal was “vague but exciting.”

That particular document didn't come to light until he passed away, and his wife, Peggy, found it. So you could see that a lot of credit is due to him for going with “exciting” rather than “vague.”

You initial mission for the Web was quite collaborative: the first browser also functioned as an editor.

The idea was that the Web should be a read/write space, so if there's an idea that isn't there as I'm browsing on the Web, I could just put it on, and other people could immediately link in their own ideas. The intention was to capture both the text of a new idea, but also the realization that it linked to another idea, which would be a brilliant environment in which to develop, for example, the real-time software that CERN needed to run experiments in the accelerators.

But in fact, the original browser only ran on the NeXT, and the NeXT did not take over the planet. So more and more people saw the Web as a read-only medium. And then HTML got more complicated, so the job of building browsers became more difficult.

It was around this time that you came to MIT.

I was working at CERN, and people from the tech industry came to me and said, “We need to form a consortium. If you were to go to MIT and start a consortium for the Web, we would join.” So I went to MIT in 1994, and it was none too early, because Microsoft and Netscape were fiercely battling over HTML tags.

The Web's original read/write aspirations were revived, to some extent, when blogs came into being.

Hypertext Web is a very flexible medium, and obviously people have done all kinds of things with it. Blogs were an early one, and if you rewind to the culture at that time, there was a strong utopian flavor to it. People would link to you and you could link to other people, and you and your computer were part of this big

“Walled gardens can be very successful ... but the walled garden can never compete with the crazy diversity of the jungle outside the gates.”

web of interconnected humanity, which felt very decentralized.

In the nationless environment of the Web, people imagined we would break down barriers and live in peace and love.

But looking back, in a way, that lapsed goal isn't such an unreasonable thing to aim for. Even though we don't have it now, I think, for a lot of people, the Web feels very centralized. With social networks, it feels as though they're still logging onto this big mainframe computer.

Of course, social networks have co-opted some of the early Web's Utopian language.

Each one of them tries to do everything that the open Web is able to do, but within a closed, controlled environment. Walled gardens can be very successful—AOL got a huge number of people online to connect and communicate. But the walled garden can never compete with the crazy diversity of the jungle outside the gates.

Let's talk about your work with the World Wide Web Foundation.

Initially, the Web Foundation was founded with two goals. One was access.

When we started, 20% of the world's population was using the Web, which is actually a huge number. But then suddenly, because it's comparable with the number of people on the planet, it becomes a small number, because it begs the question of, “What about the other people?” So the Web Foundation is partly focused on what we can do to get people who are not using the Web on board as quickly as possible.

And the second goal?

The other goal is to maintain the Web's essential qualities: its neutrality, its confidentiality, its openness, its universality in the sense that it can be used by anybody for anything without discrimination. When we started, there was no one else really thinking about communication as a right; now, more organizations have gotten involved, and there's quite a network of us collaborating and pushing for a better Web.

Neutrality and access are no longer the only challenges faced by the Web.

We'd imagined that if we keep the Web neutral, get everybody equal access, and keep it non-discriminatory, then surely humanity will do the right thing. Last year, we realized that we can't just assume people will make the right choices to provide justice or truth or democracy. So now the Web Foundation and other organizations are making a conscious, strategic decision to think about the next layer.

Sounds like there's some overlap with your work with the Web Science Trust, an organization you founded in 2006 to formalize the study of how people behave on the Web.

We started the Web Science Trust because we realized that in order to really understand the way political and emotional ideas propagate across a network of people connected by technology, we need a very multidisciplinary mix of people—psychologists and economists, as well as computer scientists and mathematicians and physicists and so on.

2016 gives an important charge to anyone who calls themselves a Web scientist.

Yes. To say, “Use your skills of analysis to understand what goes on, understand really to what extent these things support truth and democracy.” And then you need to use your skills as an engineer to build things that are better. Just looking at the existing social networks and sighing about what happens isn't going to do anything. We have to start building stuff that's better. ■

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

© 2017 ACM 0001-0782/17/06 \$15.00

Q&A

This Is for Everyone

Sir Tim Berners-Lee on the formative years of the World Wide Web, and the challenges it now faces.

BUILDING A DECENTRALIZED platform like the World Wide Web is, in many ways, a crucial test of one's ability to let go, but ACM A.M. Turing Award recipient Sir Tim Berners-Lee is proud of the explosive creativity his invention has fostered. However, that does not mean he is done refining his creation: now a professor at the Massachusetts Institute of Technology (MIT) and Oxford University, Berners-Lee is still passionately involved in the fight to keep the Web open and available to all, protect people's personal data, and stop the spread of fake news.

Your parents were both programmers.

They were mathematicians, and they worked for Ferranti, which was

commercializing the Mark 1 computer for Manchester University. At the time, they thought they would have all kinds of things next week, translating a document or solving school timetable problems. That turned out to be more difficult. But they found it was great for accounts, and they worked on some of the first cathode-ray tube GUIs (graphical user interfaces). It was very exciting, and my parents and their colleagues were full of immense challenge and opportunity.

And then, in 1989, you invented the World Wide Web. What stands out for you now about that time?

A few things are worth picking out. For instance, even though the Internet

was well established back then, it was politically inappropriate to use Internet protocols in Europe, because Europe was trying to stick to ISO protocols. There were a few people who used the Internet anyway, like Ben Segal, who was one of my mentors at CERN and suggested it would be a good way to go, even though it wasn't officially the way to go.

Your manager at CERN, Mike Sendall, also took what seems, in retrospect, like an amazing institutional leap of faith in allowing you to build the World Wide Web.

Mike didn't have an official excuse for creating a global hypertext system, because we were supposed to be working on sys- [CONTINUED ON P. 103]



PHOTOGRAPH BY ALEXANDER BERG



ACM Books



MORGAN & CLAYPOOL
PUBLISHERS

Publish your next book in the ACM Digital Library

ACM Books is a new series of advanced level books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers.

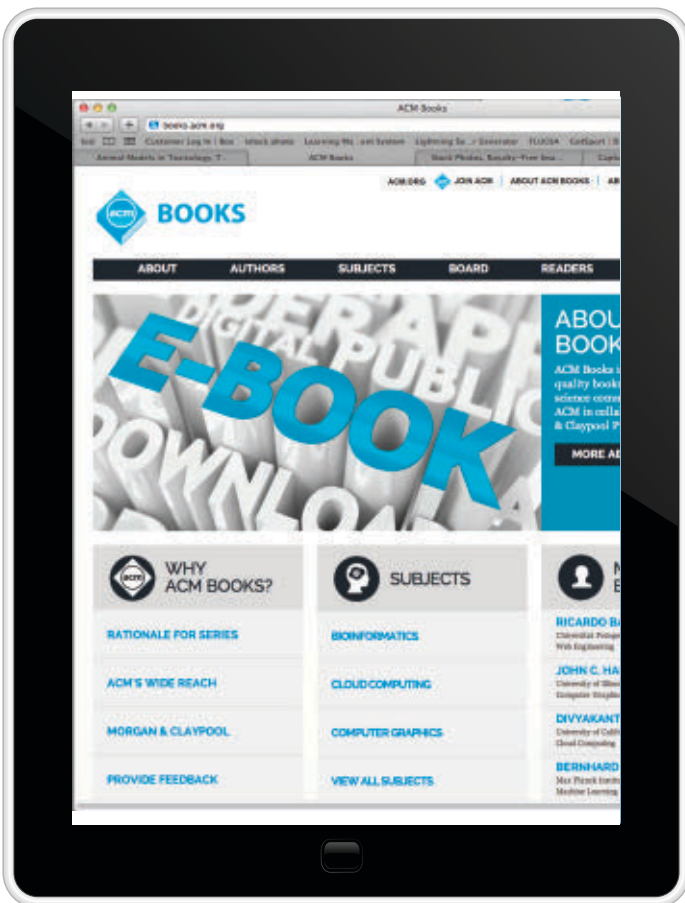
I'm pleased that ACM Books is directed by a volunteer organization headed by a dynamic, informed, energetic, visionary Editor-in-Chief (Tamer Özsu), working closely with a forward-looking publisher (Morgan and Claypool).

—Richard Snodgrass, University of Arizona

books.acm.org

ACM Books

- ◆ will include books from across the entire spectrum of computer science subject matter and will appeal to computing practitioners, researchers, educators, and students.
- ◆ will publish graduate level texts; research monographs/overviews of established and emerging fields; practitioner-level professional books; and books devoted to the history and social impact of computing.
- ◆ will be quickly and attractively published as ebooks and print volumes at affordable prices, and widely distributed in both print and digital formats through booksellers and to libraries and individual ACM members via the ACM Digital Library platform.
- ◆ is led by EIC M. Tamer Özsu, University of Waterloo, and a distinguished editorial board representing most areas of CS.



Proposals and inquiries welcome!

Contact: **M. Tamer Özsu**, Editor in Chief
booksubmissions@acm.org

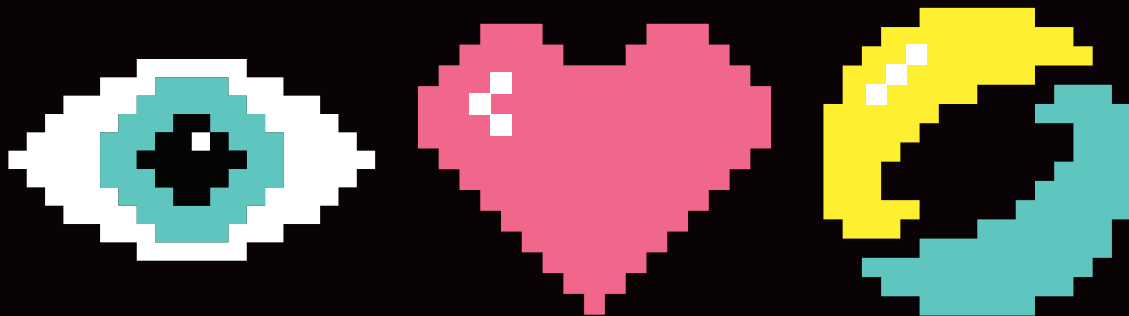


Association for
Computing Machinery

Advancing Computing as a Science & Profession

SIGGRAPH2017

AT THE  of **COMPUTER GRAPHICS &** **INTERACTIVE TECHNIQUES**



REGISTER TODAY *and* **SAVE**

FINAL OPPORTUNITY FOR SAVINGS: 7 JULY

30 JULY – 3 AUGUST

Los Angeles, California

[S2017.SIGGRAPH.ORG/REGISTRATION](https://s2017.siggraph.org/registration)



Sponsored by ACM SIGGRAPH
