

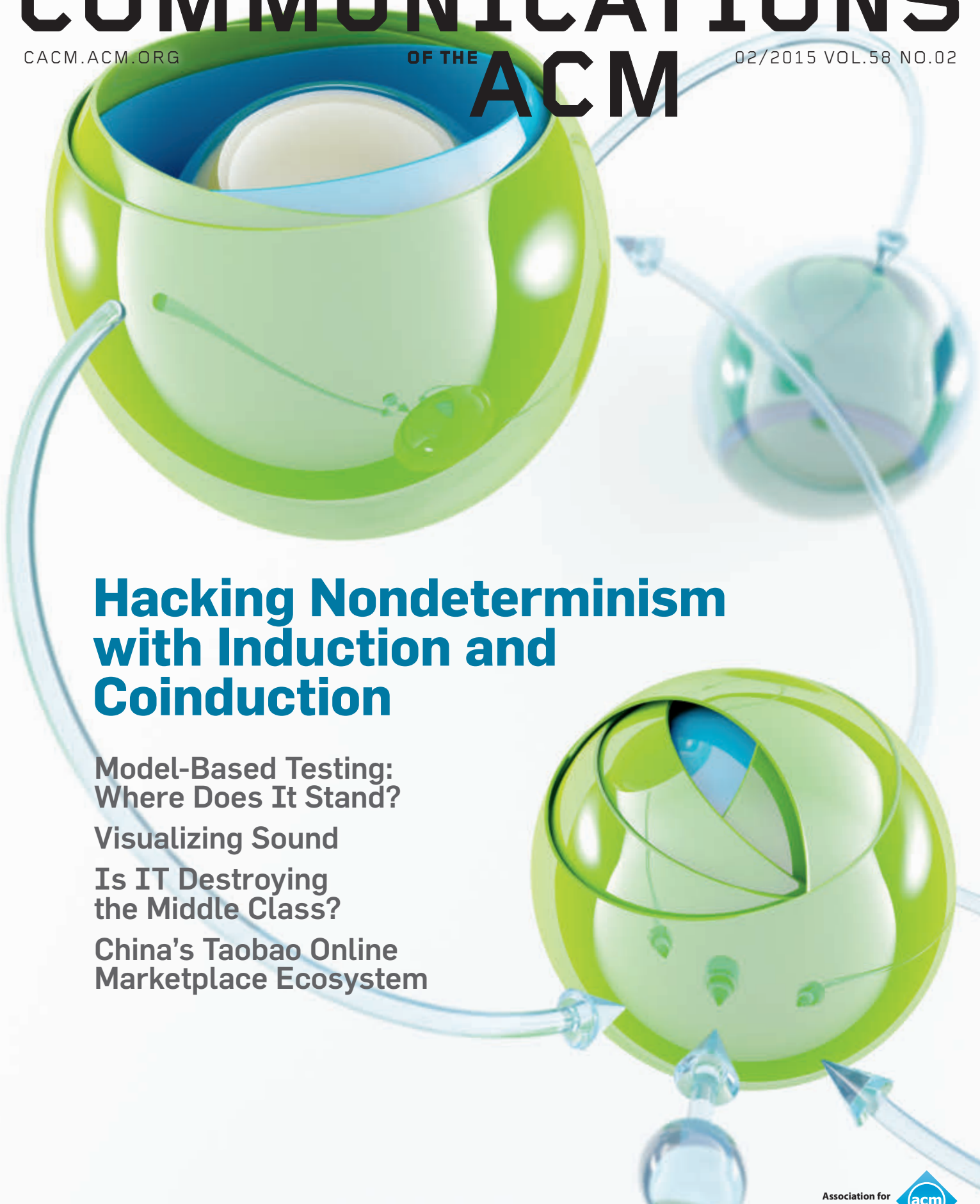
COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

02/2015 VOL.58 NO.02



Hacking Nondeterminism with Induction and Coinduction

Model-Based Testing:
Where Does It Stand?

Visualizing Sound

Is IT Destroying
the Middle Class?

China's Taobao Online
Marketplace Ecosystem

Applicative 2015

February 26-27 2015
New York City

APPLICATIVE 2015 is ACM's first conference designed specifically for practitioners interested in the latest emerging technologies and techniques.

The conference consists of two tracks:

SYSTEMS will explore topics that enable systems-level practitioners to build better software for the modern world. The speakers participating in this track are involved in the design, implementation, and support of novel technologies and low-level software supporting some of today's most demanding workloads. Topics range from memory allocation, to multicore synchronization, time, distributed systems, and more.

APPLICATIONS will cover topics such as reactive programming, single-page application frameworks, and other tools and approaches for building robust applications more quickly. The speakers slated for this track represent leading technology companies and will share how they are applying new technologies to the products they deliver.

For more information about the conference and how to register, please visit:

<http://applicative.acm.org>



Association for
Computing Machinery



ACM Books



MORGAN & CLAYPOOL
PUBLISHERS

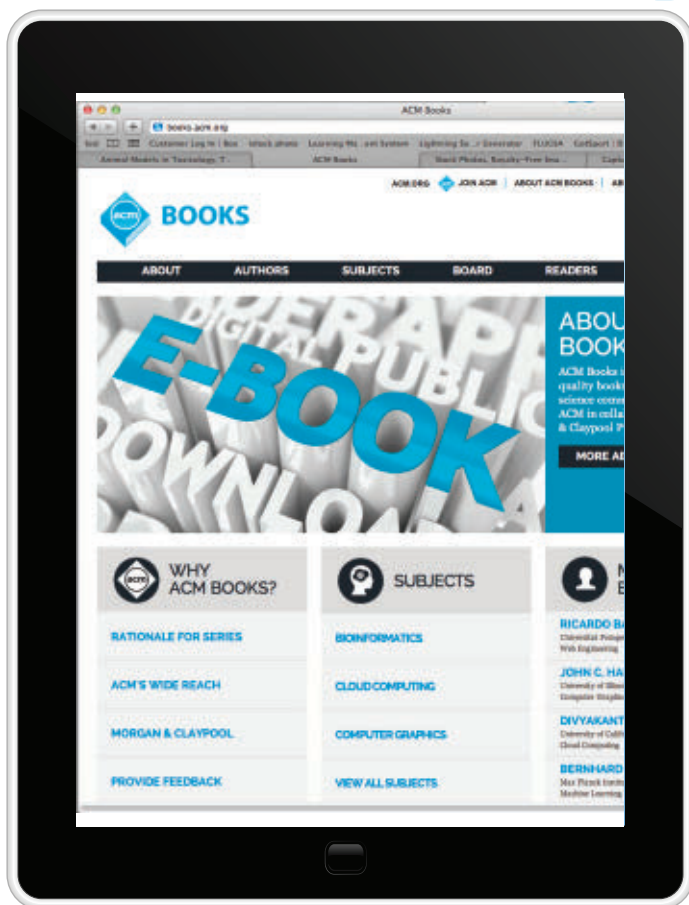
Publish your next book in the ACM Digital Library

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

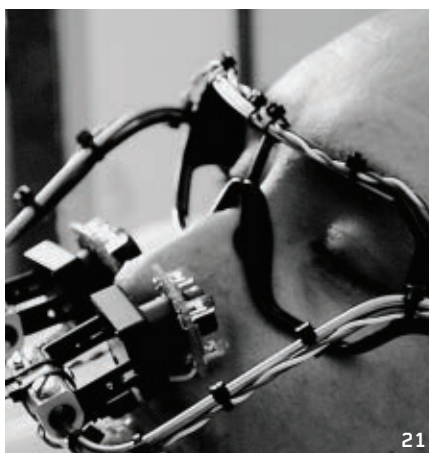
Departments

- 5 **Editor's Letter**
**Is Information Technology
Destroying the Middle Class?**
By Moshe Y. Vardi
-
- 7 **Cerf's Up**
There Is Nothing New under the Sun
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
**Software Engineering,
Like Electrical Engineering**
-
- 12 **BLOG@CACM**
**What's the Best Way to Teach
Computer Science to Beginners?**
Mark Guzdial questions the practice
of teaching programming to new CS
students by having them practice
programming largely on their own.
-
- 39 **Calendar**
-
- 97 **Careers**

Last Byte

- 104 **Upstart Puzzles**
Take Your Seats
By Dennis Shasha

News



- 15 **Visualizing Sound**
New techniques capture speech by
looking for the vibrations it causes.
By Neil Savage
-
- 18 **Online Privacy: Regional Differences**
How do the U.S., Europe, and
Japan differ in their approaches
to data protection — and what
are they doing about it?
By Logan Kugler
-
- 21 **Using Technology to Help People**
Companies are creating
technological solutions for
individuals, then generalizing
them to broader populations
that need similar assistance.
By Keith Kirkpatrick

Viewpoints

- 24 **Privacy and Security**
**We Need a Building Code
for Building Code**
A proposal for a framework for
code requirements addressing
primary sources of vulnerabilities
for building systems.
By Carl Landwehr
-
- 27 **Economic and Business Dimensions**
**Three Paradoxes
of Building Platforms**
Insights into creating China's Taobao
online marketplace ecosystem.
By Ming Zeng
-
- 30 **Inside Risks**
**Far-Sighted Thinking about
Deleterious Computer-Related Events**
Considerably more anticipation
is needed for what might
seriously go wrong.
By Peter G. Neumann
-
- 34 **Education**
**Putting the Computer Science in
Computing Education Research**
Investing in computing education
research to transform computer
science education.
By Diana Franklin
-
- 37 **Kode Vicious**
Too Big to Fail
Visibility leads to debuggability.
By George V. Neville-Neil
-
- 40 **Viewpoint**
Do-It-Yourself Textbook Publishing
Comparing experiences publishing
textbooks using traditional publishers
and do-it-yourself methods.
By Armando Fox and David Patterson
-
- 44 **Viewpoint**
**In Defense of Soundiness:
A Manifesto**
Soundy is the new sound.
By Benjamin Livshits et al.



Association for Computing Machinery
Advancing Computing as a Science & Profession

Practice



48

- 48 **Securing Network Time Protocol**
Crackers discover how to use NTP as a weapon for abuse.
By Harlan Stenn

- 52 **Model-Based Testing: Where Does It Stand?**
MBT has positive effects on efficiency and effectiveness, even if it only partially fulfills high expectations.
By Robert V. Binder, Bruno Legeard, and Anne Kramer



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

Contributed Articles



58

- 58 **To Govern IT, or Not to Govern IT?**
Business leaders may bemoan the burdens of governing IT, but the alternative could be much worse.
By Carlos Juiz and Mark Toomey

- 65 **Automated Support for Diagnosis and Repair**
Model checking and logic-based learning together deliver automated support, especially in adaptive and autonomous systems.
By Dalal Alrajeh, Jeff Kramer, Alessandra Russo, and Sebastian Uchitel

**About the Cover:**

This month's cover story, by Filippo Bonchi and Damien Pous, introduces an elegant technique for proving language equivalence of nondeterministic finite automata. Cover illustration by Zeitguised.

Review Articles



74

- 74 **Verifying Computations without Reexecuting Them**
From theoretical possibility to near practicality.
By Michael Walfish and Andrew J. Blumberg

Research Highlights

- 86 **Technical Perspective**
The Equivalence Problem for Finite Automata
By Thomas A. Henzinger and Jean-François Raskin

- 87 **Hacking Nondeterminism with Induction and Coinduction**
By Filippo Bonchi and Damien Pous



Watch the authors discuss this work in this exclusive *Communications* video.



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

Executive Director and CEO

John White

Deputy Executive Director and COO

Patricia Ryan

Director, Office of Information Systems

Wayne Graves

Director, Office of Financial Services

Darren Ramdin

Director, Office of SIG Services

Donna Cappel

Director, Office of Publications

Bernard Rous

Director, Office of Group Publishing

Scott E. Delman

ACM COUNCIL

President

Alexander L. Wolf

Vice-President

Vicki L. Hanson

Secretary/Treasurer

Erik Altman

Past President

Vinton G. Cerf

Chair, SGB Board

Patrick Madden

Co-Chairs, Publications Board

Jack Davidson and Joseph Konstan

Members-at-Large

Eric Allman; Ricardo Baeza-Yates;

Cherri Pancake; Radia Perlman;

Mary Lou Soffa; Eugene Spafford;

Per Stenström

SGB Council Representatives

Paul Beame; Barbara Boucher Owens;

Andrew Sears

BOARD CHAIRS

Education Board

Mehran Sahami and Jane Chu Prey

Practitioners Board

George Neville-Neil

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Fabrizio Gagliardi

ACM India Council

Srinivas Padmanabhuni

ACM China Council

Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs

Jack Davidson; Joseph Konstan

Board Members

Ronald F. Boisvert; Marie-Paule Cani;

Nikil Dutt; Roch Guerrin; Carol Hutchins;

Patrick Madden; Catherine McGeoch;

M. Tamer Ozsu; Mary Lou Soffa

ACM U.S. Public Policy Office

Renee Dopplick, Director

1828 L Street, N.W., Suite 800

Washington, DC 20036 USA

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

Computer Science Teachers Association

Lissa Clayborn, Acting Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

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

STAFF

DIRECTOR OF GROUP PUBLISHING

Scott E. Delman

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

Editorial Assistant

Zarina Strakhan

Rights and Permissions

Deborah Cotton

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquiot

Designer

Iwona Usakiewicz

Production Manager

Lynn D'Addesio

Director of Media Sales

Jennifer Ruzicka

Public Relations Coordinator

Virginia Gold

Publications Assistant

Juliet Chance

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@cacm.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmhlp@cacm.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

Director of Media Sales

Jennifer Ruzicka

jen.ruzicka@hq.acm.org

Media Kit acmm mediasales@acm.org

Association for Computing Machinery (ACM)

2 Penn Plaza, Suite 701

New York, NY 10121-0701 USA

T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF

Moshe Y. Vardi

eic@cacm.acm.org

NEWS

Co-Chairs

William Pulleyblank and Marc Snir

Board Members

Mei Kobayashi; Kurt Mehloth;

Michael Mitzenmacher; Rajeev Rastogi

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;

John Leslie King

Board Members

William Aspray; Stefan Bechtold;

Michael L. Best; Judith Bishop;

Stuart I. Feldman; Peter Freeman;

Mark Guzdial; Rachelle Hollander;

Richard Ladner; Carl Landwehr;

Carlos Jose Pereira de Lucena;

Beng Chin Ooi; Loren Terveen;

Marshall Van Alstyne; Jeannette Wing



PRACTICE

Co-Chairs

Stephen Bourne

Board Members

Eric Allman; Charles Beeler; Bryan Cantrill;

Terry Coatta; Stuart Feldman; Benjamin Fried;

Pat Hanrahan; Tom Limoncelli;

Kate Matsudaira; Marshall Kirk McKusick;

Erik Meijer; George Neville-Neil;

Theo Schlossnagle; Jim Waldo

The Practice section of the CACM

Editorial Board also serves as

the Editorial Board of *queue*.

CONTRIBUTED ARTICLES

Co-Chairs

Al Aho and Andrew Chien

Board Members

William Aiello; Robert Austin; Elisa Bertino;

Gilles Brassard; Kim Bruce; Alan Bundy;

Peter Buneman; Peter Druschel;

Carlo Ghezzi; Carl Gutwin; Gal A. Kaminka;

James Larus; Igor Markov; Gail C. Murphy;

Shree Nayar; Bernhard Nebel;

Lionel M. Ni; Kenton O'Hara;

Sriram Rajamani; Marie-Christine Rousset;

Avi Rubin; Krishan Sabnani;

Ron Shamir; Yoav Shoham; Larry Snyder;

Michael Vitale; Wolfgang Wahlster;

Hannes Werthner; Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Gregory Morrisett

Board Members

Martin Abadi; Amir El Abbadi; Sanjeev Arora;

Dan Boneh; Andrei Broder; Stuart K. Card;

Jeff Chase; Jon Crowcroft; Matt Dwyer;

Alon Halevy; Maurice Herlihy; Norm Jouppi;

Andrew B. Kahng; Xavier Leroy; Kobbi Nissim;

Mendel Rosenblum; David Salesin;

Steve Seitz; Guy Steele, Jr.; David Wagner;

Margaret H. Wright

WEB

Chair

James Landay

Board Members

Marti Hearst; Jason I. Hong;

Jeff Johnson; Wendy E. MacKay

ACM Copyright Notice

Copyright © 2015 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@acm.org or fax (212) 869-0481.

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

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$100.

ACM Media Advertising Policy

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

Single Copies

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

COMMUNICATIONS OF THE ACM

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

POSTMASTER

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

Printed in the U.S.A.



Association for
Computing Machinery





DOI:10.1145/2666241

Moshe Y. Vardi

Is Information Technology Destroying the Middle Class?

The Kansas City Federal Reserve Bank's symposium in Jackson Hole, WY, is one of the world's most watched economic events. Focusing on important economic issues

that face the global economy, the symposium brings together most of the world's central bankers. The symposium attracts significant media attention and has been known for its ability to move markets.

While the most anticipated speakers at the 2014 meeting were Janet Yellen, chair of the Board of Governors of the Federal Reserve System, and Mario Draghi, president of the European Central Bank, it was a talk by David Autor, an MIT labor economist that attracted a significant level of attention. Autor presented his paper, "Polanyi's Paradox and the Shape of Employment Growth." The background for the paper was the question discussed in the July 2013 *Communications* editorial: Does automation destroy more jobs than it creates? While the optimists argue that though technology always destroys jobs, it also creates new jobs, the pessimists argue that the speed in which information technology is currently destroying jobs is unparalleled.

Based on his analysis of recent labor trends as well as recent advances in artificial intelligence (AI), Autor concluded, "Journalists and expert commentators overstate the extent of machine substitution for human labor. The challenges to substituting machines for workers in tasks requiring adaptability, common sense, and creativity remain immense," he argued. The general media welcomed Autor's talk with a palpable sense of relief and headlines such as "Everybody Relax: An MIT Economist Explains Why Robots Won't Steal Our Jobs." But a care-

ful reading of Autor's paper suggests that such optimism may be premature.

Autor's main point in the paper is that "our tacit knowledge of how the world works often exceeds our explicit understanding," which poses a significant barrier to automation. This barrier, known as "Polanyi's Paradox," is well recognized as the major barrier for AI. It is unlikely, therefore, that in the near term, say, the next 10 years, we will see a major displacement of human labor by machines. But Autor himself points out that contemporary computer science seeks to overcome the barrier by "building machines that learn from human examples, thus inferring the rules we tacitly apply but do not explicitly understand." It is risky, therefore, to bet we will not make major advances against Polanyi's Paradox, say, in the next 50 years.

But another main point of Autor's paper, affirming a decade-old line of research in labor economics, is that while automation may not lead to broad destruction of jobs, at least not in the near term, automation is having a major impact on the economy by creating polarization of the labor market. Information technology, argues Autor, is destroying wide swaths of routine office and manufacturing jobs. At the same time, we are far from being able to automate low-skill jobs, often requiring both human interaction and unstructured physical movement. Furthermore, information technology creates new high-skill jobs, which require cognitive skills that computers cannot

match. Projections by the U.S. Bureau of Labor Statistics show continued significant demand for information-technology workers for years to come.

The result of this polarization is a shrinking middle class. In the U.S., middle-income jobs in sales, office work, and the like used to account for the majority of jobs. But that share of the labor market has shrunk over the past 20 years, while the share of high-end and low-end work expanded. Autor's data shows this pattern—shrinkage in the middle and growth at the high and low ends—occurred also in 16 EU countries.

The immediate outcome of this polarization is growing income and wealth disparity. "From 1979 through 2007, wages rose consistently across all three abstract task-intensive categories of professional, technical, and managerial occupations," noted Autor. Their work tends to be complemented by machines, he argued, making their services more valuable. In contrast, wages have stagnated for middle-income workers, and the destruction of middle-income jobs created downward pressure on low-income jobs. Indeed, growing inequality of income and wealth has recently emerged as a major political issue in the developed world.

Autor is a long-term optimist, arguing that in the long run the economy and workforce will adjust. But AI's progress over the past 50 years has been nothing short of dramatic. It is reasonable to predict that its progress over the next 50 years would be equally impressive. My own bet is on disruption rather than on equilibrium and adjustment.

Follow me on Facebook, Google+, and Twitter.

Moshe Y. Vardi, EDITOR-IN-CHIEF

Copyright held by author.



Association for Computing Machinery (ACM) Chief Executive Officer

ACM, the Association for Computing Machinery, invites applications for the position of Chief Executive Officer (CEO).

ACM is the oldest and largest educational and scientific computing society with 108,000 members worldwide. The association has an annual budget of \$60 million, 75 full-time staff in New York and Washington DC, a rich publications program that includes 50 periodicals in computing and hundreds of conference proceedings, a dynamic set of special interest groups (SIGs) that run nearly 200 conferences/symposia/workshops each year, initiatives in India, China, and Europe, and educational and public policy initiatives. ACM is the world's premier computing society.

The ACM CEO serves as the primary executive responsible for the formulation and implementation of ACM strategic direction, for representing ACM in the worldwide computing community, and for overall management of the affairs of the association. The successful candidate will have a high professional standing in the computing field, executive experience, leadership skills, and a vision of the future of professional societies and computing. The CEO reports to the ACM President. The CEO is not required to work from ACM's New York headquarters, but he or she must be able to travel frequently to headquarters and other ACM meetings and venues. The full job description can be found at: ceosearch.acm.org

Interested applicants should contact the ACM CEO Search Committee: ceosearch@acm.org

The ACM is an equal opportunity employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, national origin, age, protected veteran status or status as an individual with disability.



**Association for
Computing Machinery**



DOI:10.1145/2714559

Vinton G. Cerf

There Is Nothing New under the Sun

By chance, I was visiting the Folger Shakespeare Library^a last December where a unique manuscript was on display. It is called the Voynich Manuscript^b and all indications are it was written sometime

between 1410 and 1430. No one has succeeded in decoding it. Among the many who have tried was William Friedman, the chief cryptologist for the National Security Agency at the time of its founding. Friedman and his wife, Elizabeth, were great authorities on antiquities and together published books on this and many other topics. Of note is a book on Shakespearean Ciphers published in 1957^c exploring the use of ciphers in Shakespeare's works and contemporary writings.

I was interested to learn there are many books and manuscripts devoted to this mysterious codex. A brief Web search yielded a bibliography of many such works.^d Friedman ultimately concluded this was not a cipher but rather a language invented, *de novo*, whose structure and alphabet were unknown. In what I gather is typical of Friedman, he published his opinion on this manuscript as an anagram of the last paragraph in his article on acrostics and anagrams found in Chaucer's *Canterbury Tales*.^e Properly rearranged, the anagram reads:

"The Voynich MS was an early attempt to construct an artificial or universal language of the a priori type."

Friedman also drew one of our com-

puter science heroes into the fray, John Von Neumann. A photo of the two of them conferring on this topic was on display at the Folger. There is no indication that Von Neumann, a brilliant polymath in his own right, was any more able than Friedman to crack the code.

I was frankly astonished to learn that Francis Bacon devised a binary encoding scheme and wrote freely about it in a book published in 1623.^f In effect, Bacon proposed that one could hide secret messages in what appears to be ordinary text (or any other images) in which two distinct "characters" could be discerned, if you knew what to look for. He devised a five-bit binary method to encode the letters of the alphabet. For example, he would use two typefaces as the bits of the code, say, **W** and **W**. Bacon referred to each typeface as "A" and "B." He would encode the letter "a" as "AAAAA" and the letter "b" as "AAAAB," and "c" as AAABA, and so on through the alphabet. The actual image of the letter "a" could appear as **"theme"** since all five letters are in the bolder typeface (**AAAAA**). Of course, any five letters would do, and could be part of a word, all of a word, broken across two words. The letter "b" could be encoded as **"theme"** since this word is written as "AAAAB" in Bacon's "bilateral" code. Any pair of subtle differences could be used to hide the message—a form of steganography. Of course the encoding need not consist of five-letter words. "abc" could be encoded as: the hidden

message and would be read out as:

/the hi/ddenm/essag/e.../AAAA/AAAAB/AAABA/...

Examples at the Folger exhibit included a piece of sheet music in which the legs of the notes were either complete or slightly broken to represent the two "typefaces" of the binary code.

Showing my lack of knowledge of cryptography, I was quite surprised to realize that centuries before George Boole and Charles Babbage, the notion of binary encoding was well known and apparently even used!

Secret writing was devised in antiquity. Julius Caesar was known to use a simple rotational cipher (for example, "go back three letters" so that "def" would be written as "abc") so that this kind of writing is called Caesar Cipher. Of course, there are even older examples of secret writing. I need to re-read David Kahn's famous book^g on this subject.

Returning to binary for a moment, one is drawn to the possibility of using other systems than binary, not to encode, but to compute. As 2015 unfolds, I await further progress on quantum computing because there are increasing reports that the field is rapidly advancing. Between that and the neuromorphic chips that have been developed, one is expecting some very interesting research results for the rest of this year and, indeed, the decade. ■

g D. Kahn. *The Codebreakers—The Story of Secret Writing*. (1996), ISBN 0-684-83130-9.

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google.

Copyright held by author.

a <http://www.folger.edu/>

b <http://brbl-dl.library.yale.edu/vufind/Record/3519597>

c W. and E. Friedman. *The Shakespearean Ciphers Examined*. Cambridge Univ. Press, 1957.

d <http://www.ic.unicamp.br/~stolfi/voynich/mirror/reeds/bib.html>

e Friedman, W.F., and Friedman, E.S. *Acrostics, Anagrams, and Chaucer*. (1959), 1–20.

f F. Bacon (1561–1626). *De Dignitate & augmentis scientiarum*, John Havilland, 1623.

Software Engineering, Like Electrical Engineering

THOUGH I AGREE with the opening lines of Ivar Jacobson's and Ed Seidewitz's article "A New Software Engineering" (Dec. 2014) outlining the "promise of rigorous, disciplined, professional practices," we must also look at "craft" in software engineering if we hope to raise the profession to the status of, say, electrical or chemical engineering. My 34 years as a design engineer at a power utility, IT consultant, and software engineer shows me there is indeed a role for the software engineer in IT. Consider that electricity developed first as a science, then as electrical engineering when designing solutions. Likewise, early electrical lab technicians evolved into today's electrical fitters and licensed engineers.

The notion of software engineer has existed for less than 30 years and is still evolving from science to craft to engineering discipline. In my father's day (50 years ago) it was considered a professional necessity for all engineering students to spend time "on the tools," so they would gain an appreciation of practical limitations when designing solutions. Moving from craft to engineering science is likewise important for establishing software engineering as a professional discipline in its own right.

I disagree with Jacobson's and Seidewitz's notion that a "...new software engineering built on the experience of software craftsmen, capturing their understanding in a foundation that can then be used to educate and support a new generation of practitioners. Because craftsmanship is really all about the practitioner, and the whole point of an engineering theory is to support practitioners." When pursuing my master's of applied science in IT 15 years ago, I included a major in software engineering based on a software engineering course at Carnegie Mellon University covering state analysis

of safety-critical systems using three different techniques.

Modern craft methods like Agile software development help produce non-trivial software solutions. But I have encountered a number of such solutions that rely on the chosen framework to handle scalability, assuming that adding more computing power is able to overcome performance and user response-time limitations when scaling the software for a cloud environment with perhaps tens of thousands of concurrent users.

In the same way electrical engineers are not called in to design the wiring system for an individual residence, many software applications do not need the services of a software engineer. The main benefit of a software engineer is the engineer's ability to understand a complete computing platform and its interaction with infrastructure, users, and other systems, then design a software architecture to optimize the solution's performance in that environment or select an appropriate platform for developing such a solution.

Software engineers with appropriate tertiary qualifications deserve a place in IT. However, given the many tools available for developing software, the instances where a software engineer is able to add real benefit to a project may not be as numerous as in other more well-established engineering disciplines.

Ross Anderson, Melbourne, Australia

No Hacker Burnout Here

I disagree strongly with Erik Meijer's and Vikram Kapoor's article "The Responsive Enterprise: Embracing the Hacker Way" (Dec. 2014) saying developers "burn out by the time they reach their mid-30s." Maybe it is true that "some" or even perhaps "many" of us stop hacking at around that age. But the generalization is absolutely false as stated.

Some hackers do burn out and some do not. This means the proposition is erroneous, if not clearly offensive to the admitted minority still hacking away. I myself retired in 2013 at 75. And yes, I was the oldest hacker on my team and the only one born in the U.S., out of nine developers. Meijer himself is likely no spring chicken, given that he contributed to Visual Basic, yet he is likewise still hacking away. At the moment, I am just wrapping up a highly paid contract; a former client called me out of retirement. Granted, these are just two cases. Nonetheless, Meijer's and Kapoor's generalization is therefore false; it takes only one exception.

I do agree with them that we hackers (of any age) should be well-compensated. Should either of their companies require my services, my rate is \$950 per day. If I am needed in summer—August to September—I will gladly pay my own expenses to any location in continental Europe. I ride a motorcycle through the Alps every year and would be happy to take a short break from touring to roll out some code; just name the language/platform/objective.

As to the other ideas in the article—old (closed-loop system) and new (high pay for developers)—more research is in order. As we say at Wikipedia, "citation needed." Meanwhile, when we find one unsubstantiated pronouncement that is blatantly false in an article, what are we to think of those remaining?

Keith Davis, San Francisco, CA

What to Do About Our Broken Cyberspace

Cyberspace has become an instrument of universal mass surveillance and intrusion threatening everyone's creativity and freedom of expression. Intelligence services of the most powerful countries gobble up most of the world's long-distance communications traffic and are able to hack into almost any cellphone, personal computer, and

data center to seize information. Preparations are escalating for preemptive cyberwar because a massive attack could instantly shut down almost everything.¹ Failure to secure endpoints—cellphones, computers, data centers—and securely encrypt communications end-to-end has turned cyberspace into an active war zone with sporadic attacks.

Methods I describe here can, however, reduce the danger of preemptive cyberwar and make mass seizure of the content of citizens' private information practically infeasible, even for the most technically sophisticated intelligence agencies. Authentication businesses, incorporated in different countries, could publish independent directories of public keys that can then be cross-referenced with other personal and corporate directories. Moreover, hardware that can be verified by independent parties as operating according to formal specifications has been developed that can make mass break-ins using operating system vulnerabilities practically infeasible.² Security can be further enhanced through interactive biometrics (instead of passwords) for continuous authentication and through interactive incremental revelation of information so large amounts of it cannot be stolen in one go. The result would be strong, publicly evaluated cryptography embedded in independently verified hardware endpoints to produce systems that are dramatically more secure than current ones.

FBI Director James Comey has proposed compelling U.S. companies to install backdoors in every cellphone and personal computer, as well as in other network-enabled products or services, so the U.S. government can (with authorization of U.S. courts) hack in undetected. This proposal would actually *increase* the danger of cyberwar and decrease the competitiveness of almost all U.S. industry due to the emerging Internet of Things, which will soon include almost everything, thus enabling mass surveillance of citizens' private information. Comey's proposal has already increased mistrust by foreign governments and citizens alike, with the result that future exports of U.S. companies will have to be certified by corporate officers and verified by independent third parties not to have backdoors available to the U.S. government.

Following some inevitable next major terror attack, the U.S. government will likely be granted bulk access to all private information in data centers of U.S. companies. Consequently, creating a more decentralized cyberspace is fundamental to preserving creativity and freedom of expression worldwide. Statistical procedures running in data centers are used to try to find correlations in vast amounts of inconsistent information. An alternative method that can be used on citizens' cellphones and personal computers has been developed to robustly process inconsistent information² thereby facilitating new business implementations that are more decentralized—and much more secure.

References

1. Harris, S. *@War: The Rise of the Military-Internet Complex*. Eamon Dolan/Houghton Mifflin Harcourt. Boston, MA, 2014.
2. Hewitt, C. and Woods, J., assisted by Spurr, J., Editors. *Inconsistency Robustness*. College Publications. London, U.K., 2014.

Carl Hewitt, Palo Alto, CA

Ordinary Human Movement As False Positive

It might indeed prove difficult to train software to detect suspicious or threatening movements based on context alone, as in Chris Edwards's news story "Decoding the Language of Human Movement" (Dec. 2014). Such difficulty also makes me wonder if a surveillance software system trained to detect suspicious activity could view such movement as "strange" and "suspicious," given a particular location and time, and automatically trigger a security alert. For instance, I was at a bus stop the other day and a fellow rider started doing yoga-like stretching exercises to pass the time while waiting for the bus. Projecting a bit, could we end up where ordinary people like the yoga person would be compelled to move about in public like stiff robots for fear of triggering a false positive?

Eduardo Coll, Minneapolis, MN

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.

© 2015 ACM 0001-0782/15/02 \$15.00

Coming Next Month in COMMUNICATIONS

Local Laplacian Filters

Privacy Implications of Health Information Seeking on the Web

Developing Statistical Privacy for Your Data

Who Owns IT?

META II

HTTP 2.0—The IETF Is Phoning In

The Real Software Crisis: Repeatability as a Core Value

Why Did Computer Science Make a Hero Out of Turing?

Q&A with Bertrand Meyer

Plus the latest news about organic synthesis, car-to-car communication, and Python's popularity as a teaching language.

ACM

ON A MISSION TO SOLVE TOMORROW.



Dear Colleague,

Computing professionals like you are driving innovations and transforming technology across continents, changing the way we live and work. We applaud your success.

We believe in constantly redefining what computing can and should do, as online social networks actively reshape relationships among community stakeholders. We keep inventing to push computing technology forward in this rapidly evolving environment.

For over 50 years, ACM has helped computing professionals to be their most creative, connect to peers, and see what's next. We are creating a climate in which fresh ideas are generated and put into play.

Enhance your professional career with these exclusive ACM Member benefits:

- Subscription to ACM's flagship publication ***Communications of the ACM***
- Online books, courses, and webinars through the **ACM Learning Center**
- Local Chapters, Special Interest Groups, and conferences all over the world
- Savings on peer-driven specialty magazines and research journals
- The opportunity to subscribe to the **ACM Digital Library**, the world's largest and most respected computing resource

We're more than computational theorists, database engineers, UX mavens, coders and developers. Be a part of the dynamic changes that are transforming our world. Join ACM and dare to be the best computing professional you can be. Help us shape the future of computing.

Sincerely,

A stylized, handwritten signature in black ink, appearing to read 'Alexander Wolf'.

Alexander Wolf
President
Association for Computing Machinery



Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING.

JOIN ACM TODAY.

ACM is the world's largest computing society, offering benefits that can advance your career and enrich your knowledge with life-long learning resources. We dare to be the best we can be, believing what we do is a force for good, and in joining together to shape the future of computing.

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

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

ACM STUDENT MEMBERSHIP:

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

- ☐ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in all aspects of the computing field. Available at no additional cost.

Priority Code: CAPP

Payment Information

Name

ACM Member #

Mailing Address

City/State/Province

ZIP/Postal Code/Country

Email

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc, in U.S. dollars or equivalent in foreign currency.

- ☐ AMEX ☐ VISA/MasterCard ☐ Check/money order

Total Amount Due

Credit Card #

Exp. Date

Signature

Return completed application to:
ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Satisfaction Guaranteed!

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



Association for
Computing Machinery

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



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

DOI:10.1145/2714488

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

What's the Best Way to Teach Computer Science to Beginners?

Mark Guzdial questions the practice of teaching programming to new CS students by having them practice programming largely on their own.



Mark Guzdial
"How We Teach Introductory Computer Science is Wrong"

<http://bit.ly/1qnv6gy>

October 8, 2009

I have been interested in John Sweller and *Cognitive Load Theory* (<http://bit.ly/1lSmG0f>) since reading Ray Lister's ACE keynote paper from a couple years back (<http://bit.ly/1wPYrkU>). I assigned several papers on the topic (see the papers in the References) to my educational technology class. Those papers have been influencing my thinking about how we teach computing.

In general, we teach computing by asking students to engage in the activity of professionals in the field: by programming. We lecture to them and have them study texts, of course, but most of the learning is expected to occur through the practice of programming. We teach programming by having students program.

The original 1985 Sweller and Cooper paper on worked examples had five

studies with similar set-ups. There are two groups of students, each of which is shown two worked-out algebra problems. Our experimental group then gets eight more algebra problems, completely worked out. Our control group solves those eight more problems. As you might imagine, the control group takes **five times** as long to complete the eight problems than the experiment group takes to simply read them. Both groups then get new problems to solve. **The experimental group solves the problems in half the time and with fewer errors than the control group.** Not problem-solving leads to better problem-solving skills than those doing problem-solving. That's when Educational Psychologists began to question the idea that we should best teach problem-solving by having students solve problems.

The paper by Kirschner, Sweller, and Clark (KSC) is the most outspoken and most interesting of the papers in this thread of research. The title states their basic premise: "Why Minimal Guidance During Instruction Does Not Work: An

Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching." What exactly is minimal instruction? And are they really describing us? I think this quote describes how we work in computing education pretty well:

There seem to be two main assumptions underlying instructional programs using minimal guidance. First they challenge students to solve "authentic" problems or acquire complex knowledge in information-rich settings based on the assumption that having learners construct their own solutions leads to the most effective learning experience. Second, they appear to assume that knowledge can best be acquired through experience based on the procedures of the discipline (i.e., seeing the pedagogic content of the learning experience as identical to the methods and processes or epistemology of the discipline being studied; Kirschner, 1992).

That seems to reflect our practice, paraphrasing as, "people should learn to program by constructing programs from the basic information on the language, and they should do it in the same way that experts do it." The paper then goes presents evidence showing that this "minimally guided instruction" does not work.

After a half-century of advocacy associated with instruction using minimal guidance, it appears there is no body of research supporting the technique. Insofar as there is any evidence from controlled studies, it almost uniformly supports direct, strong instructional guidance rather than constructivist-based minimal guidance during the instruction of novice to intermediate learners.

There have been rebuttals to this article. What is striking is that they basically say, “But not problem-based and inquiry-based learning! Those are actually guided, scaffolded forms of instruction.” What is striking is that *no one challenges KSC on the basic premise, that putting introductory students in the position of discovering information for themselves is a bad idea!* In general, the Educational Psychology community (from the papers I have been reading) says expecting students to program as a way of learning programming is an ineffective way to teach.

What should we do instead? That is a big, open question. Peter Pirolli and Mimi Recker have explored the methods of worked examples and cognitive load theory in programming, and found they work pretty well. Lots of options are being explored in this literature, from using tools like intelligent tutors to focusing on program “completion” problems (van Merriënboer and Krammer in 1987 got great results using completion rather than program generation).

This literature is *not* saying *never* program; rather, it is a bad way to *start*. Students need the opportunity to gain knowledge first, before programming, just as with reading (<http://wapo.st/1wc4gttH>). Later, there is a *expertise reversal effect*, where the worked example effect disappears then *reverses*. Intermediate students do learn better with real programming, real problem-solving. There is a place for minimally guided student activity, including programming. It is just not at the beginning.

Overall, I find this literature unintuitive. It seems obvious to me the way to learn to program is by programming. It seems obvious to me real programming can be motivating. KSC responds:

Why do outstanding scientists who demand rigorous proof for scientific assertions in their research continue to use and indeed defend on the bias of intuition alone, teaching methods that are not the most effective?

This literature does not offer a lot of obvious answers for how to do computing education better. It does, however, provide strong evidence that what we are doing is wrong, and offers pointers to how *other* disciplines have done it *better*. It is a challenge to us to question our practice.

References

Kirschner, P.A., Sweller, J., and Clark, R.E. (2006) **Why minimal guidance during instruction does not work: an analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching.** *Educational Psychologist* 41 (2) 75–86. <http://bit.ly/1BAsE0h>

Sweller, J., and Cooper, G.A. (1985) **The use of worked examples as a substitute for problem solving in learning algebra** *Cognition and Instruction* 2 (1): 59–89. <http://bit.ly/1rXzBUv>

Comments

I would like to point out a CACM article published in March 1992; “The Case for Case Studies of Programming Problems” by Marcia Linn and Michael Clancy.

In my opinion, they describe how we should teach introductory programming primarily by reading programs, and only secondarily by writing them. I was attracted to this paper by its emphasis on learning patterns of programming. The authors used this approach for years at Berkeley and it resulted in remarkable improvement in teaching effectiveness.

—Ralph Johnson

I agree. Linn and Clancy’s case studies are a great example of using findings from the learning sciences to design effective computing education. So where is the use of case studies today? Why do so few introductory classes use case studies? Similarly, the results of using cognitive tutors for teaching programming are wonderful (and CMU makes a collection of tools for building cognitive tutors readily available at <http://bit.ly/1rXAKoK>), yet few are used in our classes. The bottom line for me is there are some great ideas out there and we are not doing enough to build on these past successes. Perhaps we need to remember as teachers some of the lessons of reuse we try to instill in our students.

—Mark Guzdial

From my experience, the “minimal guidance” part is probably the key. One of the best ways to master a new language, library, “paradigm,” etc., is to read lots of exemplary code. However, after lots of exposure to such examples, nothing cements that knowledge like actually writing similar code yourself. In fact, there’s a small movement among practitioners to create and practice “dojos” and “koans” (for example, in the TDD and Ruby communities).

—K. Wampler

Another way to think about this: Why does CS expect students to learn to write before they learn to read?

—Clif Kussmaul

This interests me as a lab teaching assistant and paper-grader for introductory Java courses. Students I help fit the mold you describe. They do not know anything about programming, yet they are expected to sit down and do it. It is easy material, but they just do not know where to start.

—Jake Swanson

K. Wampler, are you familiar with Richard Gabriel’s proposal for a Masters of Fine Arts in Software (<http://bit.ly/1KeDnPB>)? It seems similar in goal. Clifton and Jake, agreed! I do not mean no programming in CS1—I believe we need hybrid approaches where students engage in a variety of activities.

—Mark Guzdial

I have always taught introductory programming with first lessons in reading programs, understanding their structure, and analyzing them. It is a written language after all. We usually learn languages first by reading, then by writing, and continuing on in complexities of both. Unfortunately, it frustrates the “ringers” in the class who want to dive right in and start programming right away.

—Polar Humenn

I fail to see why this is considered surprising or counterintuitive. Look at CS education:

► Until 2000 or so, CS programs could not rely on any courses taught in schools. It would be as if someone going for a B.Sc. in math was not educated in differential calculus and algebra, or if a B.Sc. chemistry freshman could not balance a Redox reaction. Thus CS usually had to start from the beginning, teaching all relevant material: discrete math and logic, procedural and object-oriented styles, decomposition of problems, and so on. I am sure CS education would be easier if some of the relevant material was taught in school.

► Second, the proper way to teach, at least for beginners, is practice against an “ideal model” with corrections. It is the last part where “minimally guided instruction” fails. If you want “they should do it the same way that experts do it,” experts must be on hand to correct errors and show improvements. If this is not the case, bad habits will creep in and stay.

—Michael Lewchuk

Mark Guzdial is a professor at the Georgia Institute of Technology.

© 2015 ACM 0001-0782/15/02 \$ 15.00



SCIENCE OF SECURITY

Eligibility: Any paper that exhibits **outstanding contribution** to cybersecurity science and published from **Jan. 1, 2014 - Dec. 31, 2014.**

Entries judged on scientific merit, the strength and significance of the work reported, and the degree to which the papers exemplify how to perform and report cybersecurity scientific research.

2nd Annual Best Scientific Cybersecurity Winning Paper

"Memory Trace Oblivious Program Execution"

by Chang Liu, Dr. Michael Hicks, and Dr. Elaine Shi

3RD ANNUAL
Best Scientific Cybersecurity Paper Competition

Nominations close March 31, 2015.

Submit nominations at
<http://cps-vo.org/group/sos/papercompetition>

Hosted by the Research Directorate, National Security Agency

CAREERS at the NATIONAL SECURITY AGENCY

EXTRAORDINARY WORK

Inside our walls, you will find the most extraordinary people doing the most extraordinary work. Not just finite field theory, quantum computing or RF engineering. Not just discrete mathematics or graph analytics.

It's all of these and more, rolled into an organization that leads the world in signals intelligence and information assurance.

Inside our walls you will find extraordinary people, doing extraordinary work, for an extraordinary cause: the safety and security of the United States of America.

U.S. citizenship is required. NSA is an Equal Opportunity Employer.



APPLY TODAY



NSA

www.NSA.gov/Careers

APPLY TODAY

WHERE INTELLIGENCE GOES TO WORK®



Visualizing Sound

New techniques capture speech by looking for the vibrations it causes.

IN THE MOVIES, people often discover their room is bugged when they find a tiny microphone attached to a light fixture or the underside of a table. Depending on the plot, they can feed their eavesdroppers false information, or smash the listening device and speak freely. Soon, however, such tricks may not suffice, thanks to efforts to recover speech by processing other types of information.

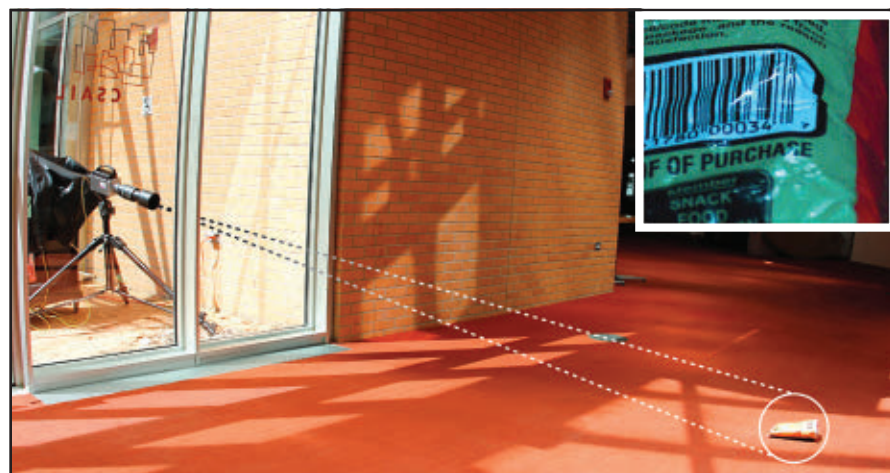
Researchers in the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology (MIT), for instance, reported at last year's SIGGRAPH meeting on a method to extract sound from video images. Among other tricks, they were able to turn miniscule motions in the leaves of a potted plant into the notes of "Mary Had a Little Lamb," and to hear a man talking based on the tiny flutterings of a potato chip bag.

The idea is fairly straightforward. Sound waves are just variations in air pressure at certain frequencies, which cause physical movements in our ears that our brains turn into information. The same sound waves can also cause tiny vibrations in objects they encounter. The MIT team merely used high-speed video to detect those motions, which were often too small for the human eye to notice, and then applied an

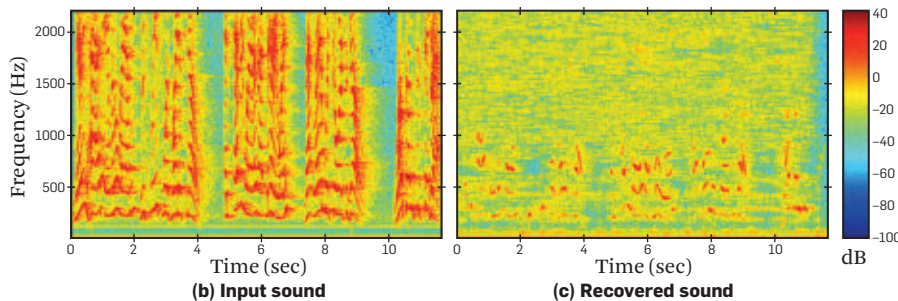
algorithm to translate the vibrations back into sound.

The work grew out of a project in MIT computer scientist William Freeman's lab that was designed not for

eavesdropping, but simply to amplify motion in video. Freeman's hope was to develop a way to remotely monitor infants in intensive care units by watching their breathing or their pulse. That



(a) Setup and representative frame



In (a), a video camera aimed at a chip bag from behind soundproof glass captures the vibrations of a spoken phrase (a single frame from the resulting 4kHz video is shown in the inset). Image (b) shows a spectrogram of the source sound recorded by a standard microphone next to the chip bag, while (c) shows the spectrogram of the recovered sound, which was noisy but understandable.

project looked for subtle changes in the phase of light from pixels in a video image, then enhanced those changes to show motion that might be otherwise unnoticeable to the naked eye.

"The focus was on amplifying and visualizing these tiny motions in video," says Abe Davis, a Ph.D. student in computer graphics, computational photography, and computer vision at MIT, and lead author of the sound recovery paper. "It turns out in a lot of cases it's enough information to infer what sound was causing it."

The algorithm, he says, is relatively simple, but the so-called visual microphone can take a lot of processing power, simply because of the amount of data involved. To capture the frequencies of human speech, the team used a high-speed camera that takes images at thousands of frames per second (fps). In one test, for instance, they filmed a bag of chips at 20,000 fps. The difficulty with such high frame rates, aside from the sheer number of images the computer has to process, is that they lead to very short exposure times, which means there must be a bright light source. At those rates, the images contain a lot of noise, making it more difficult to extract a signal.

The team got a better signal-to-noise ratio when they filmed the bag at 2,200 fps, and they improved it further with processing to remove noise. Yet even with a standard-speed camera, operating at only 60 fps—well below the 85-255Hz frequencies typical of human speech—they were able to recover intelligible sounds. They did this by taking advantage of the way many consumer video cameras operate, with a so-called rolling shutter that records the image row by row across the camera's sensor, so that the top part of the frame is exposed before the bottom part. "You have information from many different times, instead of just from the time the frame starts," explains Neal Wadwha, a Ph.D. student who works with Davis. The rolling shutter, he says, effectively increases the frame rate by eight or nine times.

Speech recovered using the rolling shutter is fairly garbled, Wadwha says, but further processing with existing techniques to remove noise and enhance speech might improve it. The method was good enough, however,

to capture "Mary Had a Little Lamb" again. "You can recover almost all of that because all the frequencies of that song are under 500Hz," he says.

Wadwha also has managed to reduce the processing time for this work, which used to take tens of minutes. Initially, researchers looked at motions at different scales and in different orientations. By picking just one view, however, they eliminated about three-quarters of the data while getting almost as good a result, Wadwha says. He is now able to process 15 seconds of video in about 10 minutes, and he hopes to reduce that further.

To their surprise, the researchers found that objects like a wine glass, which ring when struck, are not the best to sources to focus on. "We had this loose notion that things that make good sounds could make good visual microphones, and that's not necessarily the case," Davis says. Solid, ringing objects tend to produce a narrow range of frequencies, so they provide less information. Instead, light, thin objects that respond strongly to the motions of air—the potato bag, for instance, or even a piece of popcorn—are much more useful. "If you tap an object like that, you don't hear a very musical note, because the response is very broad-spectrum."

Sensing Smartphones

This imaging work is not the only way to derive sound information from vibrations. Researchers in the Applied Crypto Group at Stanford University have written software, called Gyrophone, that turns movements in a smartphone's gyroscope into speech. The gyroscopes are sensitive enough

"We had this loose notion that things that make good sounds could make good visual microphones, and that's not necessarily the case."

to pick up minute vibrations from the air or from a surface on which a handset is resting. The devices operate at 200Hz, within the frequency range of the human voice, although in any sort of signal processing, distortions creep in at frequencies above half the sampling rate, so only sounds up to 100Hz are distinguishable.

The reconstructed sound is not good enough to follow an entire conversation, says Yan Michalevsky, a Ph.D. student in the Stanford group, but there is still plenty of information to be gleaned. "You can still recognize information such as certain words and the gender of the speaker, or the identity in a group of known speakers," he says. That could be useful if, say, an intelligence agency had a speech sample from a potential terrorist it wanted to keep tabs on, or certain phrases for which it wanted to listen.

Researchers used standard machine learning techniques to train the computer to identify specific speakers in a group of known individuals, as well as to distinguish male from female speakers. They also trained it with a dictionary of 11 words—the numbers zero through 10, plus "oh." That could be useful to someone trying to steal PINs and credit card numbers. "Knowing even a couple of digits from out of this longer number would help you to guess," Michalevsky says. "The main thing here is extracting some sensitive information."

He said it would be fairly easy to place a spy program on someone's phone, disguised as a more innocent app. Most phones do not require the user to give permission to access the gyroscope or the accelerometer. On the other hand, simply changing the permissions requested could defend against the attack. Additionally, many programs that require the gyroscope would work fine with much lower sampling rates, rates that would be useless for an eavesdropper.

Sparkling Conversation

Another technique for spying on conversations, the laser microphone, has been around for some time—the CIA reportedly used one to identify the voice of Osama bin Laden. The device fires a laser beam through a window and bounces it off either an object in

the room or the window itself. An interferometer picks up vibration-induced distortions in the reflected beam and translates those into speech. Unfortunately, the setup is complicated: equipment has to be arranged so the reflected beam would return directly to the interferometer, it is difficult to separate speech from other sounds, and it only works with a rigid surface such as a window.

Zeev Zalevsky, director of the Nano Photonics Center at Bar-Ilan University, in Ramat-Gan, Israel, also uses a laser to detect sound, but he relies on a different signal: the pattern of random interference produced when laser light scatters off the rough surface of an object, known as speckle. It does not matter what the object is—it could be somebody's wool coat, or even his face. No interferometer is required. The technique uses an ordinary high-speed camera.

"The speckle pattern is a random pattern we cannot control, but we don't care," Zalevsky says. All he needs to measure is how the intensity of the pattern changes over time in response to the vibrations caused by sound. Because he relies on a small laser spot, he can focus his attention directly on a speaker and ignore nearby noise sources. The laser lets him listen from distances of a few hundred meters. The technique even works if the light has to pass through a semi-transparent object, such as clouded glass used in bathroom windows. It can use infrared lasers, which produce invisible beams that will not hurt anyone's eyes. Best of all, Zalevsky says, "The complexity of the processing is very low."

He is less interested in the spy movie aspect of the technology than in biomedical applications. It can, for instance, detect a heartbeat, and might be included in a bracelet that would measure heart rate, respiration, and blood oxygen levels. He's working with a company to commercialize just such an application.

Davis, too, sees other uses for his video technique. It might provide a way to probe the characteristics of a material without having to touch it, for instance. Or it might be useful in video editing, if an editor needs to synchronize an audio track with the picture. It might even be interesting, Da-

Zalevsky's technique relies on the pattern of random interference produced when laser light scatters off the rough surface of an object.

vis says, to use the technique on films where there is no audio, to try and recover sounds from the silence.

What it will not do, he says, is replace microphones, because the existing technology is so good. However, his visual microphone can fill in the gaps when an audio microphone is not available. "It's not the cheapest, fastest, or most convenient way to record sound," Davis says of his technique. "It's just there are certain situations where it might be the only way to record sound." **C**

Further Reading

Davis, A., Rubinstein, M., Wadhwa, N., Mysore, G.J., Durand, F., Freeman, W.T. *The Visual Microphone: Passive Recovery of Sound from Video*, *ACM Transactions on Graphics*, 2014, Vancouver, CA

Michalevsky, Y., Boneh, D. *Gyrophone: Recognizing Speech from Gyrophone Signals*, *Proceedings of the 23rd USENIX Symposium*, 2014, San Diego, CA.

Zalevsky, Z., Beiderman, Y., Margalit, I., Gingold, S., Teicher, M., Mico, V., Garcia, J. *Simultaneous remote extraction of multiple speech sources and heart beats from secondary speckles pattern*, *Optics Express*, 2009.

Wang, C.-C., Trivedi, S., Jin, F., Swaminathan, V., Prasad, N.S. *A New Kind of Laser Microphone Using High Sensitivity Pulsed Laser Vibrometer*, *Quantum Electronics and Laser Science Conference*, 2008, San Jose, CA.

The Visual Microphone
<https://www.youtube.com/watch?v=FKX0ucXB4a8>

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

© 2015 ACM 0001-0782/15/02 \$15.00

ACM Member News

BIG PICTURE ANALYTICS AND VISUALIZATION



When it comes to airplane design and assembly, David J. Kasik, senior technical fellow for

Visualization and Interactive Techniques at the Boeing Co. in Seattle, WA, takes a big-picture view—literally. Kasik spearheaded the technologies that let engineers view aircraft like Boeing's 787, and its new 777X with its 234-foot wingspan, in their entirety.

A 33-year Boeing veteran and the only computing expert among the company's 60 Senior Technical Fellows, Kasik earned his B.A. in quantitative studies from The Johns Hopkins University in 1970 and his M.S. in computer science from the University of Colorado in 1972.

Kasik's twin passions are visual analytics (VA) and massive model visualization (MMV).

VA utilizes big data analytics enabling engineers to view an entire aircraft at every stage of assembly and manufacturing, which "accelerates the design and manufacturing and lets engineers proactively debug," he says.

MMV stresses interactive performance for geometric models exceeding CPU or GPU memory; for example, a Boeing 787 model exceeds 700 million polygons and 20GB of storage.

Boeing workers use Kasik's VA and MMV work to design and build airplanes, saving the aerospace manufacturer \$5 million annually. Specialists using VA can identify issues that could endanger technicians or passengers, locate causes of excessive tool wear, and derive actionable information from myriad data sources. "We analyzed multiple databases and determined assembly tasks in the Boeing 777 that caused repetitive stress injuries," Kasik says. "Once the tasks were redesigned, the injury rate dropped dramatically."

Kasik's passion for the big picture is evident in his favorite leisure activity: New York-style four-wall handball.

—Laura DiDio

Online Privacy: Regional Differences

How do the U.S., Europe, and Japan differ in their approaches to data protection — and what are they doing about it?

ONE OF THE most controversial topics in our always-online, always-connected world is privacy. Even casual computer users have become aware of how much “they” know about our online activities, whether referring to the National Security Agency spying on U.S. citizens, or the constant barrage of ads related to something we once purchased.

Concerns over online privacy have brought different responses in different parts of the world. In the U.S., for example, many Web browsers let users enable a Do Not Track option that tells advertisers not to set the cookies through which those advertisers track their Web use. Compliance is voluntary, though, and many parties have declined to support it. On the other hand, European websites, since 2012, have been required by law to obtain visitors’ “informed consent” before setting a cookie, which usually means there is a notice on the page saying something like “by continuing to use this site, you consent to the placing of a cookie on your computer.” Why are these approaches so different?

A Short History

As the use of computers to store, cross-reference, and share data among corporations and government agencies grew through the 1960s and 1970s, so did concern about proper use and protection of personal data. The first data privacy law in the world was passed in the German region of Hesse in 1970. That same year, the U.S. implemented its Fair Credit Reporting Act, which also contained some data privacy elements. Since that time, new laws have been passed in the U.S., Europe, Japan, and elsewhere to try and keep up with technology and citizens’ concerns. Research by Graham Greenleaf of the University of New South Wales published in



Protesters marching in Washington, D.C., in 2013 in opposition to governmental surveillance of telephone conversations and online activity.

June 2013 (<http://bit.ly/ZAYgX7>) found 99 countries with data privacy laws and another 21 countries with relevant bills under consideration.

There remain fundamental differences in the approaches taken by the U.S., Europe, and Japan, however. One big reason for this, according to Katitza Rodriguez, international rights director of the Electronic Frontier Foundation (EFF), is that most countries around the world regard data protection and privacy as a fundamental right—that is written into the European Constitution, and is a part of the Japanese Act Concerning Protection of Personal Information. No such universal foundation exists in the U.S., although the Obama administration is trying to change that.

These differences create a compliance challenge for international companies, especially for U.S. companies doing business in regions with tighter privacy restrictions. Several major U.S. firms—most famously Google—have run afoul of EU regulators because of their data collection practices. In an

acknowledgment of the issue’s importance and of the difficulties U.S. businesses can face, the U.S. Department of Commerce has established “Safe Harbor” frameworks with the European Commission and with Switzerland to streamline efforts to comply with those regions’ privacy laws. After making certain its data protection practices adhere to the frameworks’ standards, a company can self-certify its compliance, which creates an “enforceable representation” that it is following recommended practices.

Data Privacy in the U.S.

EFF’s Rodriguez describes data protection in the U.S. as “sectorial.” The 1996 Health Insurance Portability and Accountability Act (HIPAA), for example, applies to medical records and other health-related information, but nothing beyond that. “In Europe, they have general principles that apply to any sector,” she says.

The U.S. relies more on a self-regulatory model, while Europe favors explicit

laws. An example of the self-regulatory model is the Advertising Self-Regulatory Council (ASRC) administered by the Council of Better Business Bureaus. The ASRC suggests placing an icon near an ad on a Web page that would link to an explanation of what information is being collected and allow consumers to opt out; however, there is no force of law behind the suggestion. Oddly, Rodriguez points out, while the formal U.S. regulatory system is much less restrictive than the European approach, the fines handed down by the U.S. Federal Trade Commission—which is charged with overseeing what privacy regulations there are—are much harsher than similar fines assessed in Europe.

The Obama administration, in a January 2012 white paper titled *Consumer Data Privacy in a Networked World: A Framework for Protecting Privacy and Promoting Innovation in the Global Digital Economy*, outlined seven privacy principles and proposed a Consumer Privacy Bill of Rights (CPBR). It stated that consumers have a right:

- ▶ to expect that data collection and use will be consistent with the context in which consumers provide the data,
- ▶ to secure and responsible handling of personal data,
- ▶ to reasonable limits on the personal data that companies collect and retain,
- ▶ to have their data handled in ways that adhere to the CPBR,
- ▶ to individual control over what personal data companies collect from them and how they use it,
- ▶ to easily understandable and accessible information about privacy and security practices, and
- ▶ to access and correct personal data in usable formats.

The CPBR itself takes a two-pronged approach to the problem: it establishes obligations for data collectors and holders, which should be in effect whether the consumer does anything or even knows about them, and “empowerments” for the consumer. The obligations address the first four principles in the list, while the empowerments address the last three.

Part of the impetus for the CPBR is to allay some EU concerns over U.S. data protection. The framework calls for working with “international partners” on making the multiple privacy schemes interoperable, which will make things

The EU is concerned with anyone that collects and tracks data, while in the U.S. the larger concern is government surveillance.

simpler for consumers and easier to negotiate for international business.

There has been little progress on the CPBR since its introduction. Congress has shown little appetite for addressing online privacy, before or after the administration’s proposal. Senators John Kerry (now U.S. Secretary of State, then D-MA) and John McCain (R-AZ) introduced the Commercial Privacy Bill of Rights Act of 2011, and Senator John D. Rockefeller IV (D-WV) introduced the Do-Not-Track Online Act of 2013; neither bill made it out of committee. At present, the online privacy situation in the U.S. remains a mix of self-regulation and specific laws addressing specific kinds of information.

Data Privacy in Europe

As EFF’s Rodriguez pointed out, the 2000 *Charter of Fundamental Rights of the European Union* has explicit provisions regarding data protection. Article 8 says, “Everyone has the right to the protection of personal data concerning him or her. Such data must be processed fairly for specified purposes and on the basis of the consent of the person concerned or some other legitimate basis laid down by law. Everyone has the right of access to data which has been collected concerning him or her, and the right to have it rectified.”

Even before the Charter’s adoption, a 1995 directive of the European Parliament and the Council of the European Union read, “Whereas data-processing systems are designed to serve man; whereas they must, whatever the nationality or residence of natural persons, respect their fundamental rights and freedoms.” These documents establish the EU-wide framework and

foundation for online privacy rights.

The roots of the concern, says Rodriguez, lie in the countries’ memory of what happened under Nazi rule. “They understand that state surveillance is not only a matter of what the government does, but that a private company that holds the data can give it to the government,” she says. Consequently, the EU is concerned with anyone that collects and tracks data, while in the U.S. the larger concern is government surveillance rather than corporate surveillance, “though I think that’s changing.”

The EU’s principles cover the entire Union, but it is up to individual countries to carry them out in practice. “Implementation and enforcement varies from country to country,” explains Rodriguez. “In Spain, Google is suffering a lot, but it’s not happening so much in Ireland. It’s not uniform.”

In December 2013, the Spanish Agency for Data Protection fined Google more than \$1 million for mismanaging user data. In May 2014, the European Court of Justice upheld a decision by the same agency that Google had to remove a link to obsolete but damaging information about a user from its results; in response, Google set up a website to process requests for information removal, and by the end of that month claimed to have received thousands of requests.

Online Privacy in Japan

The legal framework currently governing data privacy in Japan is the 2003 Act Concerning Protection of Personal Information. The Act requires businesses handling personal information to specify the reason and purpose for which they are collecting it. It forbids businesses from changing the information past the point where it still has a substantial relationship to the stated use and prohibits the data collector from using personal information more than is necessary for achieving the stated use without the user’s consent. The Act stipulates exceptions for public health reasons, among others.

Takashi Omamuya, a staff writer for Japanese Information Technology (IT) publication *Nikkei Computer*, says the Japanese government was expected to revise the 2003 law this year, “due to the fact that new technologies have weakened its protections.” Changes probably will be influenced by both the

European Commission's Data Protection Directive and the U.S. Consumer Privacy Bill of Rights (as outlined in the Obama administration white paper), as well as by the Organization for Economic Co-operation and Development (OECD) 2013 privacy framework.

In preparation for such revisions, the Japanese government established a Personal Information Review Working Group. "Some Japanese privacy experts advocate that the U.S. Consumer Privacy Bill of Rights and FTC (Federal Trade Commission) staff reports can be applied in the revision," says Omam-yuda, "but for now these attempts have failed." Meanwhile, Japanese Internet companies are arguing for voluntary regulation rather than legal restrictions, asserting such an approach is necessary for them to be able to utilize big data and other innovative technologies and to support international data transfer.

As one step in this process, the Japanese government announced a "policy outline" for the amendment of these laws in June 2014. "The main issue up for revision," says Omam-yuda, "is permitting the transfer of de-identified data to third parties under the new 'third-party authority.'" The third-party authority would be an independent

body charged with data protection. "No one is sure whether this amendment would fill the gap between current policy and the regulatory approaches to on-line privacy in the EU and U.S."

The Japanese government gathered public comments, including a supportive white paper from the American Chamber of Commerce in Japan which, unsurprisingly, urged that any reforms "take the least restrictive approach, respect due process, [and] limit compliance costs."

Conclusion

With the world's data borders becoming ever more permeable even as companies and governments collect more and more data, it is increasingly important that different regions are on the same page about these issues. With the U.S. trying to satisfy EU requirements for data protection, and proposed reforms in Japan using the EU's principles and the proposed U.S. CPBR as models, policies appear to be moving in that direction. 

Further Reading

2014 Japanese Privacy Law Revision Public Comments, Keio University International Project for the Internet & Society <http://bit.ly/1E8X3kR>

Act Concerning Protection of Personal Information (Japan Law No. 57, 2003) <http://bit.ly/1rIjZ3M>

Charter of Fundamental Rights of the European Union <http://bit.ly/1oGRu37>

Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data <http://bit.ly/1E8UxuT>

Greenleaf, G.
Global Tables of Data Privacy Laws and Bills
<http://bit.ly/ZAYgX7>

Consumer Data Privacy in a Networked World: A Framework for Protecting Privacy and Promoting Innovation in the Global Digital Economy, Obama Administration White Paper, February 2012, <http://1.usa.gov/1rRdMUw>

The OECD Privacy Framework, Organization for Economic Co-operation and Development, <http://bit.ly/1tnkiil>

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

© 2015 ACM 0001-0782/15/02 \$15.00

Milestones

U.S. Honors Creator of 1st Computer Database

U.S. President Barack Obama recently presented computer technology pioneer, data architect, and ACM A.M. Turing Award recipient Charles W. Bachman with the National Medal of Technology and Innovation for fundamental inventions in database management, transaction processing, and software engineering, for his work designing the first computer database.

The ceremony at the White House was followed by a gala celebrating the achievements and contributions to society of Bachman and other pioneers in science and technology.

Bachman received his bachelor's degree in mechanical engineering from Michigan State University, and a master's degree in that discipline from the University of Pennsylvania.

He went to work for Dow Chemical in 1950, eventually becoming that company's first data processing manager. He joined General Electric, where in 1963 he developed the Integrated Data Store (IDS), one of the first database management systems.

He received the ACM A.M. Turing Award in 1973 for "his outstanding contributions to database technology." Thomas Haigh, an associate professor of information studies at the University of Wisconsin, Milwaukee, and chair of the SIGCIS group for historians of computing, wrote at the time, "Bachman was the first Turing Award winner without a Ph.D., the first to be trained in engineering rather than science, the first to win for the application of computers to business administration, the first to win for a specific piece of software,

and the first who would spend his whole career in industry."

On being presented with the National Medal of Technology and Innovation, Bachman said, "As a boy growing up in Michigan making Soap Box Derby racers, I knew that all I wanted to do when I grew up was to build things. I wanted to be an engineer. And I wanted to make the world a better place. An honor like this is something I never expected, so I'm deeply grateful to the President, Senator Edward J. Markey, and everyone at the Department of Commerce who voted for the recognition."

He added, "It is important for me to credit my late wife, Connie, who was my partner in creativity, in business, and in life. There are a lot of friends, family and colleagues who helped along the way, of course.

I'd really like to thank them all, and especially those at General Electric who gave me the creative opportunities to invent. It is amazing how much faith GE had in our team with no guarantee of a useful result.

"I hope that young people just starting out can look at an honor like this and see all of the new creative opportunities that lay before them today, and the differences they can make for their generation and for future generations."

President Obama said Bachman and the other scientists honored with the National Medal of Science and the National Medal of Technology and Innovation embody the spirit of the nation and its "sense that we push against limits and that we're not afraid to ask questions."

—Lawrence M. Fisher

Using Technology to Help People

Companies are creating technological solutions for individuals, then generalizing them to broader populations that need similar assistance.

SOCIAL ENTREPRENEURS ARE not content just to give a fish or teach how to fish. They will not rest until they have revolutionized the fishing industry.”

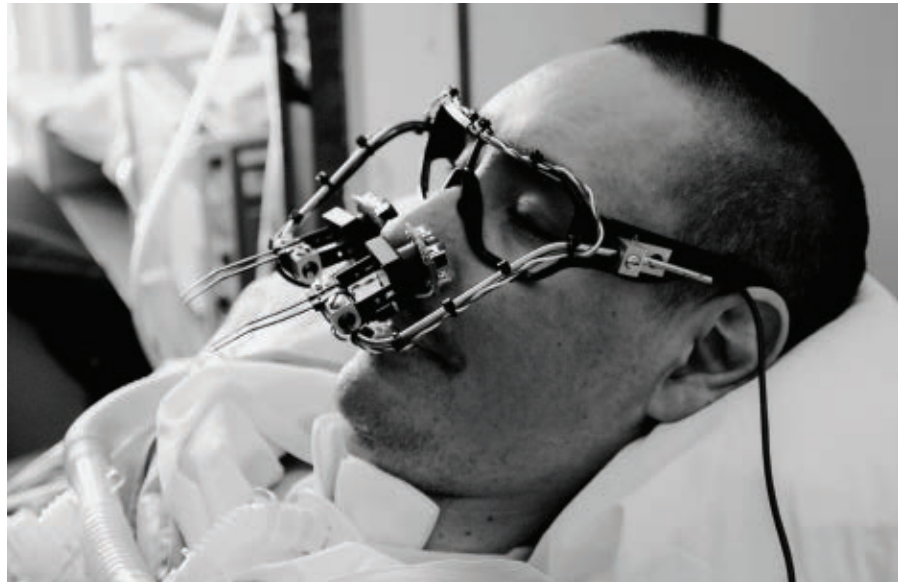
—Bill Drayton, *Leading Social Entrepreneurs Changing the World*

Entrepreneur Elliot Kotek and his business partner Mick Ebeling have taken Bill Drayton’s observation to heart, working to ensure technology can not only help those in need, but that those in need may, in turn, help create new technologies that can help the world.

Kotek and Ebeling co-founded Not Impossible Labs, a company that finds solutions to problems through brainstorming with intelligent minds and sourcing funding from large companies in exchange for exposure. Unlike most charitable foundations or commercial developers, Not Impossible Labs seeks out individuals with particular issues or problems, and works to find a solution to help them directly.

Not Impossible Labs initially began in 2009, when co-founder Mick Ebeling organized a group of computer hackers to find a solution for a young graffiti artist named Tempt One, who was diagnosed with amyotrophic lateral sclerosis (ALS) and quickly became fully paralyzed, unable to move any part of his body except his eyes.

The initial plan was simply to do a fundraiser, but the graffiti artist’s brother told Ebeling that more than money, the artist just wanted to be able to communicate. Existing technology to allow patients with severely restricted physical movement to communicate via their eyes (such as the system used by Stephen Hawking, the noted physicist afflicted with ALS, or Lou Gehrig’s



Paralyzed artist Tempt One wears the Eyewriter, a low-cost, open source eye-tracking system that allows him to draw using just his eyes.

disease) cost upward of \$100,000 then, which was financially out of reach for the young artist and his family.

As a result of the collaboration between the hackers, a system was designed that could be put together for just \$250, which allowed him to draw again. “Mick Ebeling brought some hackers to his house, and they came up

Ebeling continued to put together projects designed to bring technology to those who were not in a position to simply buy a solution.

with the Eyewriter software, which enabled him to draw using ocular recognition software,” Kotek explains.

Based on the success of the Tempt One project, Ebeling continued to put together projects designed to bring technology to those who were not in a position to simply buy a solution. He soon attracted the attention of Kotek who, with Ebeling, soon drew up another 20 similar projects that they wanted to find solutions for in a similar way.

One of the projects that received significant attention is Project Daniel, which was born out of the duo’s reading about a child living in the Sudan, who lost both of his arms in a bomb attack. “We read about this doctor, Tom Catena, who was operating in a solar-powered hospital in what is effectively a war zone, and how this kid was struck by this bomb,” Kotek says, noting that he and Ebeling both felt there was a need to help Daniel, or someone like him, who probably did not have access to things

such as modern prostheses. “It was that story that compelled us to seek a solution to help Daniel or people like him.”

The project kicked off, even though Not Impossible Labs had no idea whether Daniel was still alive. However, when a group of specialists was pulled together to work on the problem, they got on a call with Catena, who noted that Daniel, several years older by now, was despondent about his condition and being a burden to his family. After finding out that Daniel, the person who inspired the project, was still alive and would benefit directly from the results of the project, the team redoubled its efforts, and came up with a solution that uses 3D printers to create simple yet customized prosthetic limbs, which are now used by Daniel.

The group left Catena on site with a 3D printer and sufficient supplies to help others in the Sudan who also had lost limbs. They trained people who remain there to use the equipment to help others, generalizing the specific solution they had developed for Daniel. That is emblematic of how Not Impossible works: creating a technology solution to meet the need of an individual, and then generalizing it out so others may benefit as well.

Not Impossible is now establishing 15 other labs around the world, which are designed to replicate and expand upon the solutions developed during Project Daniel. The labs are part of the company’s vision to create a “sustainable global community,” in which solutions that are developed in one locale

That is emblematic of how Not Impossible works: creating a technological solution to meet the need of an individual, and then generalizing it out so others may benefit as well.

can be modified, adapted, or improved upon by the users of that solution, and then sent back out to benefit others.

The aim is to “teach the locals how to use the equipment like we did in the Sudan, and teach them in a way so that they’re able to design alterations, modifications, or a completely new tool that helps them as an indigenous population,” Kotek says. “Only then can we look at what they’re doing there, and take it back out to the world through these different labs.”

Project Daniel was completed with the support of Intel Corp., and Not Impossible Labs has found other partners to support its other initiatives, including Precipart, a manufacturer of precision mechanical components, gears, and motion control systems; WPP, a

large advertising and PR company; Groundwork Labs, a technology accelerator; and MakerBot, a manufacturer of 3D printers. Not Impossible Labs will create content (usually a video) describing the problem, and detail how a solution was devised. Supporting sponsors can then use this content as a way to highlight their support projects being completed for the public good, rather than for profit.

“We delivered to Intel some content around Project Daniel,” Kotek explains, noting that the only corporate branding included with the video about the project is a simple “Thanks to Intel and Precipart for believing in the Not Impossible.” As a result of the success of the project, “Now other brands are interested in seeing how they can get involved, too, which will allow us to start new projects.”

One of the key reasons Not Impossible Labs has been able to succeed is due to the near-ubiquity of the Internet, which allows people from around the world to come together virtually to tackle a problem, either on a global or local scale.

“What we want to do is show people that regular guys like us can commit to helping someone,” Kotek says. “The resources that everyone has now, by virtue of just being connected by the Internet, via communities, by hacker communities, or academic communities ... We can be doing something to help someone close to us, without having to be an institution, or a government organization, or a wealthy philanthropist.”

Although Not Impossible Labs began as a 501(c)(3) charitable organization, it recently shifted its structure to that of a traditional for-profit corporation. Kotek says this was done to ensure the organization can continue to address a wide variety of challenges, rather than merely the cause du jour.

“As a foundation, you’re subject to various trends,” Kotek says, highlighting the success the ALS Foundation had with the ice bucket challenge, which raised more money in 2014 than the organization had in the 50 preceding years. Kotek notes that while it is a good thing that people are donating money to ALS research as a result of the ice bucket challenge, such a campaign generally impacts thousands of other worthy causes all fighting for the same dollars.



Daniel Omar, the focus of Project Daniel, demonstrating how he can use his 3D-printed prosthetic arm.

Not Impossible Labs is hardly the only organization trying to leverage technology to help people. Japanese robotics company Cyberdyne is working on the development of the HAL (hybrid assistive limb) exoskeleton, which can be attached to a person with restricted mobility to help them walk. Scientists at Cornell University are working on technology to create customized ear cartilage out of living cells, using 3D printers to create plastic molds to hold the cartilage, which can then be inserted in the ear to allow people to hear again.

Yet not all technology being developed to help people revolves around the development of complex solutions to problems. Gavin Neate, an entrepreneur and former guide dog mobility instructor, is developing applications that take advantage of technology already embedded in smartphones.

His Neate Ltd. provides two applications based around providing greater access to people with disabilities.

The Pedestrian Neatbox is an application that directly connects a smartphone to a pedestrian crossing signal, allowing a person in a wheelchair or those without sight to control the crossing activation button via their handset. Neate Ltd. has secured a contract with Edinburgh District Council in the U.K. to install the system in crossings within that city, which has allowed further development of the application and system.

Meanwhile, the Attraction Neatebox is an application that can interface with a tourist attraction, sending pre-recorded or selected content directly to a smartphone, to allow those who cannot physically visit an attraction a way to experience it. The company has conducted a trial with the National Air Museum in Edinburgh, and the company projects its first application will go live in Edinburgh City Centre by the end of 2014.

Neate says that while his applications are useful, truly helping people via technology will only come about when developers design products from the ground up to ensure accessibility by all people.

“Smart devices have the potential to level the playing field for the very first time in human history, but only if we realize that it is not in retrofitting solutions that the answers lie, but in designing from the outset with an un-

derstanding of the needs of all users,” Neate says. “Apple, Samsung, Microsoft, and others have recognized the market is there and invested millions. They have big teams dedicated to accessibility and are providing the tools as standard within their devices.”

Like Kotek, Neate believes genuinely innovative solutions will come from users themselves. “I believe the charge, however, is being led by the users themselves in their demand for more usable and engaging tools as their skills improve,” he says, noting “most solutions start with the people who understand the problem, and it is the entrepreneurs’ challenge (if they are not the problem holders themselves) to ensure that these experts are involved in the process of finding the solutions.”

Indeed, the best solutions need not even be rooted in the latest technologies. The clearest example can be found in Jason Becker, a now-45-year-old composer and former guitar phenomenon who, just a few short years after working with rocker David Lee Roth, was struck by ALS in 1989 at the age of 20.

Though initially given just a few years left to live after his diagnosis, he has continued to compose music using only his eyes, thanks to a system called Vocal Eyes developed by his father, Gary Becker. The hand-painted system allows Becker to spell words by moving his eyes to letters in separate sections of the hand-painted board, though his family has learned how to “read” his eye movements at home without the letter board. Though there are other more technical systems out there, Becker told the SF-Gate.com Web site that “I still haven’t found anything as quick and efficient as my dad’s system.” **C**

Further Reading

Not Impossible Now:
www.notimpossiblenow.com

The Pedestrian Neatebox:
<http://www.theinfohub.org/news/going-places>

Jason Becker’s Vocal Eyes demonstration:
http://www.youtube.com/watch?v=DL_ZMWru1IU

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

© 2015 ACM 0001-0782/15/02 \$15.00

Milestones

EATCS Names First Fellows

The European Association for Theoretical Computer Science (EATCS) recently recognized 10 members for outstanding contributions to theoretical science with its first EATCS fellowships.

The new EATCS Fellows are:

- Susanne Albers, Technische Universität München, Germany, for her contributions to the design and analysis of algorithms.
- Giorgio Ausiello, Università di Roma “La Sapienza,” Italy, for the impact of his work on algorithms and computational complexity.
- The late Wilfried Brauer, Technische Universität München, for contributions “to the foundation and organization of the European TCS community.”
- Herbert Edelsbrunner, Institute of Science and Technology, Austria, and Duke University, U.S., for contributions to computational geometry.
- Mike Fellows, Charles Darwin University, Australia, for “his role in founding the field of parameterized complexity theory... and for being a leader in computer science education.”
- Yuri Gurevich, Microsoft Research, U.S., for development of abstract state machines and contributions to algebra, logic, game theory, complexity theory and software engineering.
- Monika Henzinger, University of Vienna, Austria, a pioneer in web algorithms.
- Jean-Eric Pin, LIAFA, CNRS, and University Paris Diderot, France, for contributions to the algebraic theory of automata and languages in connection with logic, topology, and combinatorics.
- Paul Spirakis, University of Liverpool, UK, and University of Patras, Greece, for seminal papers on random graphs and population protocols, algorithmic game theory, and robust parallel distributed computing.
- Wolfgang Thomas, RWTH Aachen University, Germany, for contributing to the development of automata theory as a framework for modeling, analyzing, verifying and synthesizing information processing systems.

Privacy and Security

We Need a Building Code for Building Code

A proposal for a framework for code requirements addressing primary sources of vulnerabilities for building systems.

THE MARKET FOR cybersecurity professionals is booming. Reports attest to the difficulty of hiring qualified individuals; experts command salaries in excess of \$200K.⁴ A May 2013 survey of 500 individuals reported the mean salary for a mid-level “cyber-pro” as approximately \$111,500. Those with only an associate’s degree, less than one year of experience, and no certifications could still earn \$91,000 a year.⁷

Is cybersecurity a profession, or just an occupation? A profession should have “stable knowledge and skill requirements,” according to a recent National Academies study,⁵ which concluded that cybersecurity does not have these yet and hence remains an occupation. Industry training and certification programs are doing well, regardless. There are enough different certification programs now that a recent article featured a “top five” list.

Schools and universities are ramping up programs in cybersecurity, including a new doctoral program at Dakota State University. In 2010, the Obama administration began the Na-

tional Initiative for Cybersecurity Education, expanding a Bush-era initiative. The CyberCorps (Scholarships for Service) program has also seen continuing strong budgets. The National Security Agency and the Department of Homeland Security recently designated Centers of Academic Excellence in Information Assurance/Cyber Defense in 44 educational institutions.

What do cybersecurity professionals do? As the National Academies study observes, the cybersecurity work-

force covers a wide range of roles and responsibilities, and hence encompasses a wide range of skills and competencies.⁵ Nevertheless, the report centers on responsibilities in dealing with attacks, anticipating what an attacker might do, configuring systems so as to reduce risks, recovering from the aftereffects of a breach, and so on.

If we view software systems as buildings, it appears cybersecurity professionals have a lot in common with firefighters. They need to configure systems to reduce the risk of fire, but they also need to put fires out when they occur and restore the building. Indeed, the original Computer Emergency Response Team (CERT) was created just over a quarter-century ago to fight the first large-scale security incident, the Internet Worm. Now there are CERTs worldwide. Over time, CERT activities have expanded to include efforts to help vendors build better security into their systems, but its middle name remains “emergency response.”

This whole economic boom in cybersecurity seems largely to be a consequence of poor engineering. We

This whole economic boom in cybersecurity seems largely to be a consequence of poor engineering.



have allowed ourselves to become dependent on an infrastructure with the characteristics of a medieval firetrap—a maze of twisty little streets and passages bordered by buildings highly vulnerable to arson. The components we call firewalls have much more in common with fire doors: their true purpose is to enable communication, and, like physical fire doors, they are all too often left propped open. Naturally, we need a lot of firefighters. And, like firefighters everywhere, they become heroes when they are able to rescue a company's data from the flames, or, as White Hat hackers, uncover latent vulnerabilities and install urgently needed patches.¹⁰

How did we get to this point? No doubt the threat has increased. Symantec's latest Internet Threat report compares data from 2013 and 2012.⁸ Types and numbers of attacks fluctuate, but there is little doubt the past decade has seen major increases in attacks by both criminals and nation-states. Although defenses may have improved, attacks have grown more sophisticated as well, and the balance remains in favor of the attacker.

To a disturbing extent, however, the kinds of underlying flaws exploited by attackers have not changed very much. Vendors continue to release systems with plenty of exploitable flaws. Attackers continue to seek and find them. One of the most widespread vulnerabilities found recently, the so-called Heartbleed flaw in OpenSSL, was apparently overlooked by attackers (and everyone else) for more than two years.⁶ What was the flaw? Failure to apply adequate bounds-checking to a memory buffer. One has to conclude that the supply of vulnerabilities is more than sufficient to meet the current demand.

Will the cybersecurity professionals we are training now have a significant effect on reducing the supply of vulnerabilities? It seems doubtful. Most people taking these jobs are outside the software development and maintenance loops where these vulnerabilities arise. Moreover, they are fully occupied trying to synthesize resilient systems from weak components, patching those systems on a daily basis, figuring out whether they have already been compromised, and clean-

ing them up afterward. We are hiring firefighters without paying adequate attention to a building industry is continually creating new firetraps.

How might we change this situation? Historically, building codes have been created to reduce the incidence of citywide conflagrations.^{a,9} The analog of a building code for software security could seriously reduce the number and scale of fires cybersecurity personnel must fight.

Of course building codes are a form of regulation, and the software industry has, with few exceptions, been quite successful at fighting off any attempts at licensing or government regulation. The exceptions are generally in areas such as flight control software and nuclear power plant controls where public safety concerns are overwhelming. Government regulations aimed at improving commercial software security, from the TCSEC to today's Common Criteria, have affected small corners of the marketplace but have had little

a Further history on the development of building codes is available in Landwehr.³

effect on industrial software development as a whole. Why would a building code do better?

First, building codes generally arise from the building trades and architecture communities. Governments adopt and tailor them—they do not create them. A similar model, gaining consensus among experts in software assurance and in the industrial production of software, perhaps endorsed by the insurance industry, might be able to have significant effects without the need for contentious new laws or regulations in advance. Hoping for legislative solutions is wishful thinking; we need to get started.

Second, building codes require relatively straightforward inspections. Similar kinds of inspections are becoming practical for assuring the absence of classes of software security vulnerabilities. It has been observed² that the vulnerabilities most often exploited in attacks are not problems in requirements or design: they are implementation issues, such as in the Heartbleed example. Past regimes for evaluating software security have more often focused on assuring that security functions are designed and implemented correctly, but a large fraction of today's exploits depend on vulnerabilities that are at the code level and in portions of code that are outside the scope of the security functions.

There has been substantial progress in the past 20 years in the techniques of static and dynamic analysis of software, both at the programming language level and at the level of binary

analysis. There are now companies specializing in this technology, and research programs such as IARPA's STONESOUP¹ are pushing the frontiers. It would be feasible for a building code to require evidence that software for systems of particular concern (for example, for self-driving cars or SCADA systems) is free of the kinds of vulnerabilities that can be detected automatically in this fashion.

It will be important to exclude from the code requirements that can only be satisfied by expert and intensive human review, because qualified reviewers will become a bottleneck. This is not to say the code could or should ignore software design and development practices. Indeed, through judicious choice of programming languages and frameworks, many kinds of vulnerabilities can be eliminated entirely. Evidence that a specified set of languages and tools had indeed been used to produce the finished product would need to be evaluated by the equivalent of a building inspector, but this need not be a labor-intensive process.

If you speak to builders or architects, you will find they are not in love with building codes. The codes are voluminous, because they cover a multitude of building types, technologies, and systems. Sometimes builders have to wait for an inspection before they can proceed to the next phase of construction. Sometimes the requirements do not fit the situation and waivers are needed. Sometimes the code may dictate old technology or demand that dated but functional technology be replaced.

Nevertheless, architects and builders will tell you the code simplifies the entire design and construction process by providing an agreed upon set of ground rules for the structure that takes into account structural integrity, accessibility, emergency exits, energy efficiency, and many other aspects of buildings that have, over time, been recognized as important to the occupants and to the community in which the structure is located.

Similar problems may occur if we succeed in creating a building code for software security. We will need to have mechanisms to update the code as technologies and conditions change. We may need inspectors. We may need

a basis for waivers. But we should gain confidence that our systems are not vulnerable to the same kinds of attacks that have been plaguing them for an embarrassing period of years.

I do not intend to suggest we do not need the cybersecurity professionals that are in such demand today. Alas, we do, and we need to educate and train them. But the scale and scope of that need should be an embarrassment to our profession.

The kind of building code proposed here will not guarantee our systems are invulnerable to determined and well-resourced attackers, and it will take time to have an effect. But such a code could provide a sound, agreed-upon framework for building systems that would at least take the best known and primary sources of vulnerability in today's systems off the table. Let's get started! 

References

1. Intelligence Advanced Research Projects Activity (IARPA): Securely Taking on New Executable Software Of Uncertain Provenance (STONESOUP); <http://www.iarpa.gov/index.php/research-programs/stonesoup>.
2. Jackson, D., Thomas, M., and Millett, L., Eds. *Committee on Certifiably Dependable Systems, Software for Dependable Systems: Sufficient Evidence?* National Academies Press, 2007; http://www.nap.edu/catalog.php?record_id=11923.
3. Landwehr, C.E. A building code for building code: Putting what we know works to work. In *Proceedings of the 29th Annual Computer Security Applications Conference (ACSAC)*, (New Orleans, LA, Dec. 2013).
4. Libicki, M.C., Senty, D., and Pollak, J. *H4CKER5 WANTED: An Examination of the Cybersecurity Labor Market*. RAND Corp., National Security Research Division, 2014. ISBN 978-0-8330-8500-9; http://www.rand.org/content/dam/rand/pubs/research_reports/RR400/RR430/RAND_RR430.pdf.
5. National Research Council, Computer Science and Telecommunications Board. *Professionalizing the Nation's Cybersecurity Workforce?* D.L. Burley and S.E. Goodman, Co-Chairs; http://www.nap.edu/openbook.php?record_id=18446.
6. Perlroth, N. Study finds no Evidence of Heartbleed attacks before flaw was exposed. *New York Times* Bits blog (Apr. 16, 2014); <http://bits.blogs.nytimes.com/2014/04/16/study-finds-no-evidence-of-heartbleed-attacks-before-the-bug-was-exposed/>.
7. Semper Secure. *Cyber Security Census*. (Aug. 5, 2013); http://www.sempersecure.org/images/pdfs/cyber_security_census_report.pdf.
8. Symantec. *Internet Security Threat Report 2014: Vol. 19*. Symantec Corp. (Apr. 2014); www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf.
9. *The Great Fire of London, 1666*. Luminarium Encyclopedia Project; <http://www.luminarium.org/encyclopedia/greatfire.htm>.
10. White hats to the rescue. *The Economist* (Feb. 22, 2014); <http://www.economist.com/news/business/21596984-law-abiding-hackers-are-helping-businesses-fight-bad-guys-white-hats-rescue>.

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

Copyright held by author.

I am honored and delighted to have the opportunity to take the reins of *Communications' Privacy and Security* column from Susan Landau. During her tenure, Susan developed a diverse and interesting collection of columns, and I hope to continue down a similar path. I have picked up the pen myself this month, but I expect that to be the exception, not the rule. There is so much happening in both privacy and security these days that I am sure we will not lack for interesting and important topics. I will appreciate feedback from you, the reader, whether in the form of comments on what is published or as volunteered contributions.

—Carl Landwehr

Economic and Business Dimensions

Three Paradoxes of Building Platforms

Insights into creating China's Taobao online marketplace ecosystem.

P LATFORMS HAVE APPARENTLY become the holy grail of business on the Internet. The appeal is plain to see: platforms tend to have high growth rates, high entry barriers, and good margins. Once they reach critical mass, they are self-sustaining. Due to strong network effects, they are difficult to topple. Google and Apple are clear winners in this category.

Taobao.com—China's largest online retailer—has become such a successful platform as well. Taobao merchants require myriad types of services to run their storefronts: apparel models, product photographers, website designers, customer service representatives, affiliate marketers, wholesalers, repair services, to name a few. Each of these vertical services requires differentiated talent and skills, and as the operational needs of sellers change, Taobao too has evolved, giving birth to related platforms in associated industries like logistics and finance. Although it has become habit to throw the word “ecosystem” around without much consideration for its implications, Taobao is in fact a thriving ecosystem that creates an enormous amount of value.

I have three important lessons to share with future platform stewards that can be summarized in three fundamental paradoxes. These paradoxes encapsulate the fundamental difficul-



ties you will run into on the road toward your ideal platform.

The Control Paradox

Looking back at Taobao's success and failures over the past 10 years, I have come to believe that success-

ful platforms can only be built with a conviction that the platform you wish to build will thrive with the partners who will build it with you. And that conviction requires giving up a certain amount of control over your platform's evolution.



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**



People are used to being “in control.” It is a natural habit for most people to do things on their own, and by extension command-and-control has become a dominant way of thinking for most businesses. But platforms and ecosystems require completely new mind-sets. Being a platform means you must rely on others to get things done, and it is usually the case that you do not have any control over the “others” in question. Yet your fate depends on them. So especially in the early days, you almost have to have a blind faith in the ecosystem and play along with its growth.

Taobao began with a belief in third-party sellers, not a business model where we do the buying and selling. On one hand, it was our task to build the marketplace. On the other hand, Taobao grew so fast that many new services were provided on our platforms before we realized it. In a sense, our early inability to provide all services on our own caused us to “wake up” to an ecosystem mind-set, and the growing conviction that we had to allow our platform to evolve on its own accord.

Despite such strong beliefs, however, there were still many occasions when our people wanted to be in control, to do things on their own, and in the process ended up competing against our partners. When such things happen, partners start to doubt whether they can make money on your platform, and this may hamper ecosystem momentum. For example, we once introduced standard software for store design, hoping to provide a helpful service and make some extra money. However, it soon became apparent that our solution could not meet the diverse needs of millions of power sellers, and at the same time, it also impacted the business of service providers who made their living through sales of store design services. Later, we decided to offer a very simple basic module for free and left the added-value market to our partners.

What makes this paradox particularly pernicious is the fact that working with partners can be very difficult, especially when the business involved grows more and more complex. In the early days of a platform, customers are

often not happy with the services provided. Platform leaders have to work very hard to keep all parties aligned and working toward the same goal. If the platform decides to take on responsibilities itself, it will stifle growth of the ecosystem. Yet the incubation period is a long and difficult process, and requires a lot of investment. Without strong conviction, it is very difficult to muddle through this stage, which may take years.

In very specific terms, you must sell your vision of a third-party driven service ecosystem to the end user when you have very few partners (or even no partners to speak of) and when your service level is always below expectations. You must convince people to become your partners when all you have is belief in the future. And you must market your future ecosystem to investors when you do not even have a successful vertical application. Most importantly, you need to keep your faith in the ecosystem, and resist the temptation to take the quick but shortsighted path of doing everything yourself.

More than strategy, more than capital, more than luck, you need conviction. It will take a long time before you can enjoy all the nice things about a platform—critical mass, network effects, profit. Until then, you will just have to keep trying and believing.

The Weak Partner Paradox

Last year, Taobao produced about 12 million parcels per day. How do you handle a number like that? Our meteoric growth makes it impossible to run our own logistics operations: we would soon be a company with more than one million employees just running delivery centers. So we knew quite early on that we needed an open logistics platform.

But where to begin? By the time Amazon got its start, UPS, Fedex, and even Walmart had already developed mature business models, logistics networks, and human resources. You could easily build a team by hiring from these companies and leveraging third-party services providers. But China’s delivery infrastructure is weak, and its human capital even weaker. The country’s enormous size and varied terrain has ensured there

We knew quite early on that we needed an open logistics platform.

are no logistics companies that can effectively service the entire country down to the village level.

So the first question I asked myself was: Can we build a fourth-party logistics platform when there have never been third-party service providers?

Therein lies the paradox. The strong third-party logistics companies did not believe in our platform dream, while the partners who wanted to work with us were either just startups or the weakest players in the industry. Obviously, the giants did not want to join us, considering us their future competitors. But could we realize our vision by working with startups who were willing to believe, but whose ability was really not up to snuff?

No growing platform can avoid this paradox. By definition, a new service, especially one that aims to become an enormous ecosystem, must grow from the periphery of an industry with only peripheral partners to rely on. At the same time, your partners will be fighting tooth and nail amongst themselves to capture bigger shares of your growing ecosystem. Your job is to guide them together toward a nebulous goal that everyone agrees with in a very abstract sense.

If you want to build a platform, take a good, hard look at your industry. Who can you work with? Who will work with you? Do they share your conviction? How will you help them grow as you grow with them, while at the same time supervising their competition and conflict? There are no easy answers to these questions. This is why we call it an “ecosystem”: there is a strong sense of mutual dependence and coevolution. Over time, hopefully, you and your partners will become better and better at meeting your customers’ needs together.

This story has a happy ending. Our logistics platform handled five billion packages in 2013 (more than UPS), and now employs over 950,000 delivery personnel in 600 cities and 31 provinces. There are now powerful network effects at work: sellers’ products can be assigned to different centers, shipping routes can be rerouted based on geographical loading, costs and shipping timeframes continue to drop. Most importantly, we now have 14 strategic logistics partners. Once weak, they have grown alongside us, and are now professional, agile, and technologically adept.

Incidentally, one of Alibaba’s core ideals goes as follows. With a common belief, ordinary people can accomplish extraordinary things. Any successful platform begins ordinarily, even humbly, mostly because you have no other choice.

The Killer App Paradox

Back in 2008, when Salesforce.com was exploding and everyone started to realize the importance of SaaS (Software-as-a-Service), we deliberately tried to construct a platform to provide IT service for small and medium-sized enterprises in China. Taobao had already become quite successful, but China at the time had no good IT solutions for small businesses. We christened our new platform AliSoft, built all the infrastructure, invited lots of developers, and opened for business.

Unfortunately, AliSoft was a spectacular failure. We supplied all the resources a fledgling platform would need. However, there was one problem. Users were not joining, and the developers had no clients. Within two years, we had to change the business. The fundamental flaw was in our intricate but lifeless planning: we had a complete platform infrastructure that was unable to offer specific, deliverable customer value. There was no killer app, so to speak, to attract enough users that can sustain economic gains for our developers.

Alisoft taught us an important lesson. Platform managers cannot think in the abstract. Most successful platforms evolve over time from a very strong product that has profound customer value and huge market poten-

tial, from there expanding horizontally to support more and more verticals. You cannot design an intricate but empty skeleton that will provide a suite of wonderful services—this is a contradiction in terms.

To end users as well as partners, there is no such thing as a platform, *per se*; there is only a specific, individual service. So a platform needs one vertical application to act as an anchor in order to deliver value. Without that, there is no way you can grow, because nobody will use your service.

But herein lies the killer app paradox. If your vertical application does not win the marketplace, the platform cannot roll out to other adopters. And, making that one vertical very strong requires that most resources be used to support this particular service, rather than expanding the platform to support more verticals. But a platform must expand basic infrastructural services to support different verticals with different (and often conflicting) needs and problems. In other words, platform managers must balance reliance on a single vertical with the growth of basic infrastructure, which in all likelihood may weaken your commitment to continuing the success of your killer app.

What to do? How do you decide whom to prioritize when your business becomes complicated, when your ecosystem starts to have a life of its own? Managing the simultaneous evolution of verticals and infrastructure is the most challenging part of running a platform business. There is no magic bullet that will perfectly solve all your problems. You have to live through this balancing act yourself. Or put another way, your challenge is to constantly adjust on the fly but nevertheless emerge alive.

Some concluding words of wisdom for those who are not deterred by the long, difficult road ahead. Keep your convictions. Be patient. Trust and nurture your partners. And find good venture capitalists that can deal with you losing huge quantities of money for many years. 

Ming Zeng is the chief strategy officer of Alibaba Group, having previously served as a professor of strategy at INSEAD and the Cheung Kong School of Business.

Copyright held by author.

Inside Risks

Far-Sighted Thinking about Deleterious Computer-Related Events

Considerably more anticipation is needed for what might seriously go wrong.

SEVERAL PREVIOUS COMMUNICATIONS Inside Risks columns (particularly October 2012 and February 2013) have pursued the needs for more long-term planning—particularly to augment or indeed counter some of the short-term optimization that ignores the importance of developing and operating meaningfully trustworthy systems that are accompanied by proactive preventive maintenance. This column revisits that theme and takes a view of some specific risks. It suggests that advanced planning for certain major disasters relating to security, cryptography, safety, reliability, and other critical system requirements is well worth consideration. The essential roles of preventive maintenance are also essential.

Crises, Disasters, and Catastrophes

There is a wide range of negative events that must be considered. Some tend to occur now and then from which some sort of incomplete recovery may be possible—even ones that involve acts that cannot themselves be undone such as deaths; furthermore so-called recovery from major hurricanes, earthquakes, and tsunamis does not result in the same physical state as before. Such events are generally considered to be crises or disasters. Other events may occur that are totally surprising

and truly devastating (for example, the comet activity that is believed to have caused a sudden end of the dinosaurs).

In this column, I consider some events relating to computer-related systems whose likelihood might be thought possible but perhaps seemingly remote, and whose consequences

might be very far-reaching and in extreme cases without possible recoverability. Such events are generally thought of as catastrophes or perhaps cataclysms. The primary thrust here is to anticipate the most serious potential events and consider what responses might be needed—in advance.



IMAGE BY ANDRIJ BORYS ASSOCIATES/SHUTTERSTOCK

Cryptography

This column is inspired in part by a meeting that was held in San Francisco in October 2014. The CataCrypt meeting specifically considered Risks of Catastrophic Events Related to Cryptography and its Possible Applications. *Catastrophic* was perhaps an overly dramatic adjective, in that it has an air of finality and even total nonrecoverability. Nevertheless, that meeting had at least two consensus conclusions, each of which should be quite familiar to readers who have followed many of the previous Inside Risks columns.

First, it is clear that sound cryptography is essential for many applications (SSL, SSH, key distribution and handling, financial transactions, protecting sensitive information, and much more). However, it seems altogether possible that some major deleterious events could undermine our most widely used cryptographic algorithms and their implementations. For example, future events might cause the complete collapse of public-key cryptography, such as the advance of algorithms for factoring large integers and for solving discrete-log equations, as well as significant advances in quantum computing. Furthermore, some government-defined cryptography standards (for example, AES) are generally considered to be adequately strong enough for the foreseeable future—but not forever. Others (for example, the most widely used elliptic-curve standard) could themselves already have been compromised in some generally unknown way, which could conceivably be why several major system developers prefer an alternative standard. Recent attacks involving compromisable random-number generators and hash functions provide further warning signs.

As a consequence of such possibilities, *it would be very appropriate to anticipate new alternatives and strategies for what might be possible in order to recover from such events.* In such a situation, planning now for remediation is well worth considering. Indeed, understanding that nothing is perfect or likely to remain viable forever, various carefully thought-through successive alternatives (Plan B, Plan C, and so forth) would be desirable. You might think that we already have some po-

Essentially every system architect, program-language and compiler developer, and programmer is a potential generator of flaws and risks.

tential alternatives with respect to the putative demise of public-key cryptography. For example, theoretical bases for elliptic-curve cryptography have led to some established standards and their implementations, and more refined knowledge about lattice-based cryptography is emerging—although they may be impractical for all but the most critical uses. However, the infrastructure for such a progression might not be ready for widespread adoption in time in the absence of further planning. Newer technologies typically take many years to be fully supported. For example, encrypted email has been very slow to become easily usable—including sharing secret keys in a secret-key system, checking their validity, and not embedding them in readable computer memory. Although some stronger implementations are now emerging, they may be further retarded by some nontechnical (for example, policy) factors relating to desired surveillance (as I will discuss later in this column). Also, some systems are still using single DES, which has now been shown to be susceptible to highly distributed exhaustive cracking attacks—albeit one key at a time.

Second, it is clear—even to cryptographers—that even the best cryptography cannot by itself provide total-system solutions for trustworthiness, particularly considering how vulnerable most of our hardware-software systems and networks are today. Furthermore, the U.S. government and other nations are desirous of being able to monitor potentially all computer, network, and other communications, and seek to have special access paths available for

surveillance purposes (for example, backdoors, frontdoors, and hopefully exploitable hidden vulnerabilities). The likelihood those access paths could be compromised by other parties (or misused by trusted insiders) seems much too great. Readers of past Inside Risks columns realize almost every computer-related system and network in existence is likely to have security flaws, exploitable vulnerabilities and risks of insider misuse. Essentially every system architect, program-language and compiler developer, and programmer is a potential generator of flaws and risks.

Recent penetrations, breaches, and hacking suggest that the problems are becoming increasingly worse. Key management by a single user or among multiple users sharing information could also be subverted, as a result of system security flaws, vulnerabilities, and other weaknesses. Even worse, almost all information has now been digitized, and is available either on the searchable Internet or on the unsearchable Dark Net.

This overall situation could turn out to be a disaster for computer system companies (who might now be less trusted than before by their would-be customers), or even a catastrophe in the long run (for example, if attackers wind up with a perpetual advantage over defenders because of the fundamental inadequacy of information security). As a consequence, *it is essential that systems with much greater trustworthiness be available for critical uses—and especially in support of trustworthy embeddings of cryptography and critical applications.*

Trustworthy Systems, Networks, and Applications

Both of the preceding italicized conclusions are highly relevant more generally—to computer system security, reliability, safety, and many other properties, irrespective of cryptography. Events that could compromise the future of an entire nation might well involve computer-related subversion or accidental breakdown of critical national infrastructures, one nation's loss of faith in its own ability to develop and maintain sufficiently secure systems, loss of domestic marketplace presence as a result of other nations' unwillingness to acquire and use inferior (potentially

compromisable) products, serious loss of technological expertise, and many other scenarios. The situation is further complicated by many other diverse nontechnological factors—both causes and effects—for example, involving politics, personal needs, governmental regulation or the lack thereof, the inherently international nature of the situation, diplomacy, reputations of nations, institutions and individuals, many issues relating to economics, consequences of poor planning, moral/working conditions, education, and much more.

We consider here a few examples in which the absence of sufficient trustworthiness might result in various kinds of disaster. In each case, differences in scope and negative impacts are important considerations. In some cases, an adverse event may be targeted at specific people or data sources. In other cases, the result may have much more global consequences.

Ubiquitous surveillance, especially when there is already insufficient trustworthiness and privacy in the systems and networks being surveilled. Planning for a world in which the desires for meaningfully trustworthy systems have been compromised by ubiquitous surveillance creates an almost impossible conflict. The belief that having backdoors and even systemic flaws in systems to be used by intelligence and law-enforcement operatives without those vulnerabilities not being exploited by others seems totally fatuous.² As a consequence, the likelihood of having systems that can adequately enforce security, privacy, and many other requirements for trustworthiness seem to have almost totally disappeared. The result of that reality suggests that many routine activities that depend on the existence of trustworthy systems will themselves be untrustworthy—human safety in our daily lives, financial transactions, and much more.

There is very little room in the middle for an acceptable balance of the needs for security and the needs for surveillance. A likely lack of accountability and oversight could seriously undermine both of these two needs.

Privacy and anonymity are also being seriously challenged. Privacy requires much more than secure systems to store information, because

many of the privacy violations are external to those systems. But a total privacy meltdown seems to be emerging, where there will be almost no expectation of meaningful privacy. Furthermore, vulnerable systems combined with surveillance results in potential compromises of anonymity. A recent conclusion that 81% of users of a sampling of anonymizing Tor network users can be de-anonymized by analysis of router information^{1,a} should not be surprising, although it seems to result from an external vulnerability rather than an actual flaw in Tor. Furthermore, the ability to derive accurate and relatively complete analyses from communication metadata and digital footprints must be rather startling to those who previously thought their actions were secure, when their information is widely accessible to governments, providers of systems, ISPs, advertisers, criminals, approximately two million people in the U.S. relating to healthcare data, and many others.

A Broader Scope

Although the foregoing discussion specifically focuses primarily on computer-related risks, the conclusions are clearly relevant to much broader problems confronting the world today, where long-term planning is essential but is typically deprecated. For example, each of the following areas

a Roger Dingledine's blog item and an attached comment by Sambuddho, both of which qualify the 81% number as being based on a small sample. See <https://blog.torproject.org/blog/traffic-correlation-using-netflows>.

Privacy requires much more than secure systems to store information, because many of the privacy violations are external to those systems.

is often considered to be decoupled from computer-communication technologies, but actually is often heavily dependent on those technologies. In addition, many of these areas are interdependent on one another.

► *Critical national infrastructures* are currently vulnerable, and in many cases attached directly or indirectly to the Internet (which potentially implies many other risks). *Telecommunications providers* seem to be eager to eliminate landlines wherever possible. You might think that we already have Plan B (mobile phones) and Plan C, such as Skype or encrypted voice-over-IP. However, such alternatives might assume the Internet has not been compromised, that widespread security flaws in malware-susceptible mobile devices might have been overcome, and that bugs or even potential backdoors might not exist in Skype itself. Furthermore, taking out a few cell towers or satellites or chunks of the Internet could be highly problematic. *Water supplies* are already in crisis in some areas because of recent droughts, and warning signs abound. Canadians recall the experience in Quebec Province in the winter of 1996–1997 when power distribution towers froze and collapsed, resulting in the absence of power and water for many people for almost a month. Several recent hurricanes are also reminders that we might learn more about preparing for and responding to such emergencies. *Power generation and distribution* are monitored and controlled by computer systems are potentially vulnerable. For example, NSA Director Admiral Michael Rogers recently stated that China and “probably one or two other” countries have the capacity to shut down the nation’s power grid and other critical infrastructures through a cyberattack.^b Clearly, more far-sighted planning is needed regarding such events, including understanding the trade-offs involved in promoting, developing, and maintaining efficient alternative sources.

► *Preservation and distribution of clean water supplies* clearly require extensive planning and oversight in the face of severe water shortages and

b CNN.com (Nov. 21, 2014); <http://www.cnn.com/2014/11/20/politics/nsa-china-power-grid/>.

lack of sanitation in some areas of the world, and the presence of endemic diseases where no such supplies currently exist. Computer models that relate to droughts and their effects on agriculture are not encouraging.

► Understanding the importance of proactive *maintenance of physical infrastructures* such as roadways, bridges, railway track beds, tunnels, gas mains, oil pipelines, and much more is also necessary. From a reliability perspective, many power lines and fiber-optic telecommunication lines are located close to railroad and highway rights of way, which suggests that maintenance of bridges and tunnels is particularly closely related to continuity of power, and indeed the Internet.

► *Global warming and climate change* are linked with decreasing water availability, flooding, rising ocean temperatures, loss of crops and fishery welfare. Computer modeling consistently shows incontrovertible evidence about extrapolations into the future, and isolates some of the causes. Additional computer-related connections include micro-controlling energy consumption (including cooling) for data centers, and relocating server complexes into at-risk areas—the New York Stock Exchange’s computers must be nearby, because so much high-frequency trading is affected by speed-of-light latency issues. Moving data centers would be vastly complex, in that it would require many brokerage firms to move their operations as well.

► *Safe and available world food production* needs serious planning, including consideration of sustainable agriculture, avoidance of use of pesticides in crops and antibiotics in grain feeds, and more. This issue is of course strongly coupled with climate change.

► *Pervasive health care* (especially including preventive care and effective alternative treatments) is important to all nations. The connections with information technologies are pervasive, including the safety, reliability, security and privacy of healthcare information systems and implanted devices. Note that a catastrophic event for a healthcare provider could be having its entire collection of records harvested, through insider misuse or system penetrations. The aggregate

The biggest realization may be that many of these problem areas are closely interrelated, sometimes in mysterious and poorly understood ways.

of mandated penalties could easily result in bankruptcy of the provider. Also, class-action suits against manufacturers of compromised implanted devices, test equipment, and other related components (for example, remotely accessible over the Internet or controlled by a mobile device) could have similar consequences.

► *Electronic voting systems and compromises to the democratic process* present an illustrative area that requires total-system awareness. Unauditable proprietary systems are subject to numerous security, integrity, and privacy issues. However, nontechnological issues are also full of risks, with fraud, manipulation, unlimited political contributions, gerrymandering, cronyism, and so on. Perhaps this area will eventually become a poster child for accountability, despite being highly politicized. However, remediation and restoration of trust would be difficult. Eliminating unaccountable all-electronic systems might result in going back to paper ballots but procedural irregularities remain as non-technological problems.

► *Dramatic economic changes* can result from all of the preceding concerns. Some of these potential changes seem to be widely ignored.

The biggest realization here may be that many of these problem areas are closely interrelated, sometimes in mysterious and poorly understood

ways, and that the interrelations and potential disasters are very difficult to address without visionary total-system long-term planning.

Conclusion

Thomas Friedman³ has written about the metaphor of *stampeding black elephants*, combining the black swan (an unlikely unexpected event with enormous ramifications) and the elephant in the room (a problem visible to everyone as being likely to result in black swans that no one wants to address). Friedman’s article is concerned with the holistic preservation of our planet’s environment, and is not explicitly computer related. However, that metaphor actually encapsulates many of the issues discussed in this column, and deserves mention here. Friedman’s discussion of renewed interest in the economic and national security implications is totally relevant here—and especially soundly based long-term economic arguments that would justify the needs for greater long-term planning. That may be precisely what is needed to encourage pursuit of the content of this column.

In summary, without being a predictor of doom, this column suggests we need to pay more attention to the possibilities of potentially harmful computer-related disasters, and at least have some possible alternatives in the case of serious emergencies. The principles of fault-tolerant computing need to be generalized to disaster-tolerant planning, and more attention paid to stampeding black elephants. ■

References

1. Anderson, M. 81% of users can be de-anonymised by analysing router traffic, research indicates. *The Stack*; <http://thestack.com/chakravarty-traffic-analysis-141114>.
2. Bellovin, S.M., Blaze, M., Diffie, W., Landau, S., Neumann, P.G., and Rexford, J. Risking communications security: Potential hazards of the Protect America Act. *IEEE Security and Privacy* 6, 1 (Jan.–Feb. 2008), 24–33.
3. Friedman, T.L. Stampeding black elephants: Protected land and parks are not just zoos. They’re life support systems. *The New York Times Sunday Review* (Nov. 23, 2014), 1, 9.

Peter G. Neumann (neumann@csl.sri.com) is Senior Principal Scientist in the Computer Science Lab at SRI International, and moderator of the ACM Risks Forum.

I am grateful to members of the Catacrypt steering committee and the ACM Committee on Computers and Public Policy, whose thoughtful feedback greatly improved this column.

Copyright held by author.

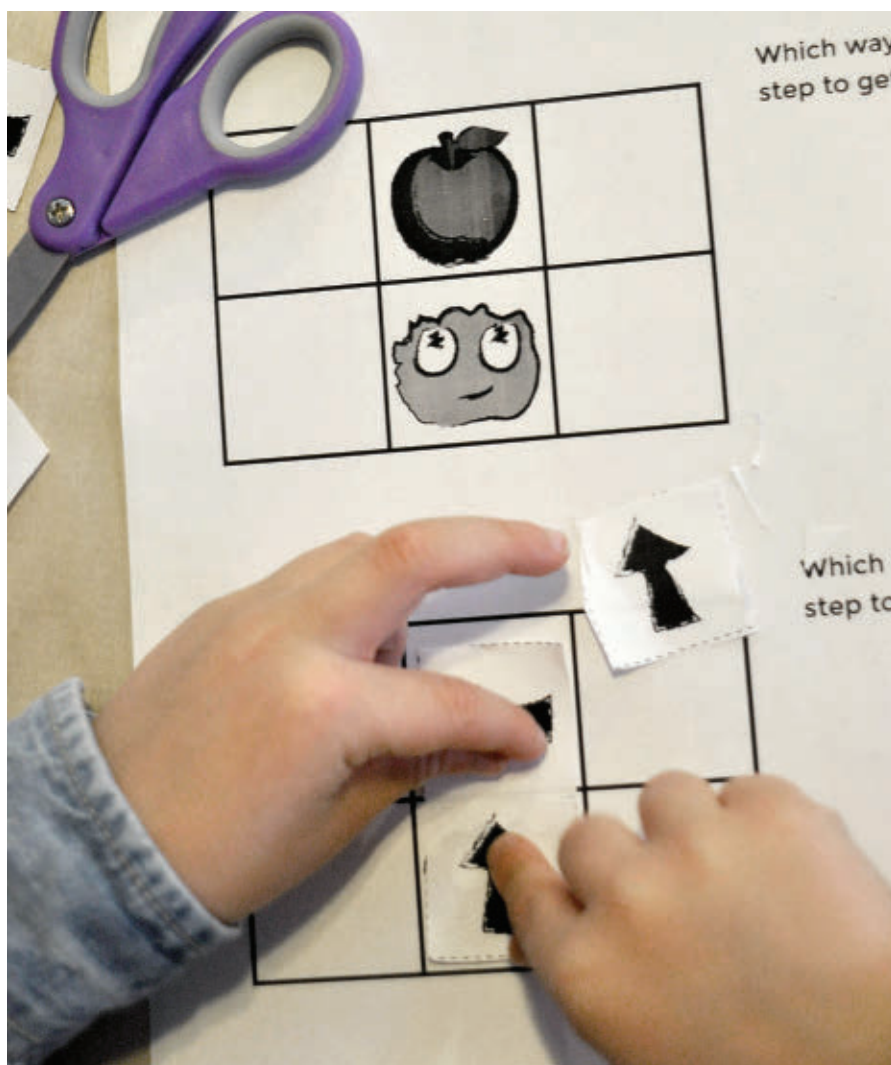
Education

Putting the Computer Science in Computing Education Research

Investing in computing education research to transform computer science education.

IN APRIL 2013, San Jose State University announced an expansion of its pilot MOOC courses with EdX based on a successful offering of EE98 Circuits Analysis. In July 2013, San Jose State University suspended its MOOC project with EdX when half the students taking the courses failed their final exams. In August 2014, Code.org released a K–6 curriculum to the world that had been created a few months earlier, not having been tested rigorously in elementary school classrooms.

What do these events have in common? Computer scientists identified a critical need in computer science education (and education in general) and developed something new to fill that need, released it, and scaled it without rigorous, scientific experiments to understand in what circumstances they are appropriate. Those involved have the best of intentions, working to create a solution in an area with far too little research. A compelling need for greater access to computing education, coupled with a dire shortage of well-supported computing education researchers, has led to deployments that come before research. High-profile failures hurt computer science's credibility in education, which in turn hurts our future students. This imbalance between the demand and supply and the chasm



A child working on the "Happy Maps" lesson from Code.org.

between computer science and education creates an opportunity for some forward-thinking departments.

If computer science wants to be a leader rather than a spectator in this field, computer science departments at Ph.D.-granting institutions must hire faculty in computing education research (CER) to transform the face of education—in undergraduate CS teaching, K–12 CS education, and education in general.

Finding a Place

As with any interdisciplinary field, we must explore in which department should this research be performed, and in what role should this researcher be hired? To understand *where* computing education research belongs, we need to understand *what* computing education research is. I divide it into two categories:

- How students learn computing concepts, whether that is K–12 or college, how their prior experiences (gender, ethnicity, socioeconomic background, geographic region, generation) influence how they learn, or how those findings influence the ways we should teach.

- What interfaces, languages, classroom techniques and themes are useful for teaching CS concepts.

The appropriate department depends on the particular research questions being asked. I am not advocating that every department should hire in CER, nor that all CER research should occur in CS departments. However, the biggest opportunity lies in hiring leaders who will assemble an interdisciplinary team of education faculty, CS faculty, CS instructors, and graduate students to make transformative contributions in these areas.

The first question is in what department should the research take place? The answer is both education (learning science, cognitive science, and so forth) and computer science. The most successful teams will be those pulling together people from both departments. Researchers need deep expertise in computer science as well as a robust understanding of the types of questions and methods used in education. If computer science departments fail to take a leadership role, computer science instruction will continue to

suffer from a gap in content, methods, and tools.

We can look in history for two examples: engineering education and computational biology. Preparation of physics, math, and science K–12 education largely happens in education departments or schools because these core subjects are required in K–12. Engineering K–12 education, on the other hand, has found a place in the college of engineering as well as education. Research on college-level instruction occurs in cognate departments. In all solutions, the cognate field is a large part of the research. Computational biology is another example. Initially, both biology and computer science departments were reluctant to hire in this area. Computer science departments felt biology was an application of current algorithms. The few departments who saw the promise of computational biology have made transformative discoveries in mapping the human genome and driving new computer science areas such as data mining. The same will hold true for computing education research. Computer scientists need to lead not only to make advances in computing education, but also to find the problems that will drive computer science research.

One might argue, then, that lecturers or teaching faculty can continue to perform computing education research. Why do we need tenure-track faculty? It is no accident that several successful transformative educational initiatives—including Scratch, Alice, and Computational Media—have been developed by teams led by tenure-track faculty. Like any systems field, making a large impact in computing education requires time and graduate students. Lecturers and teaching faculty with high loads and few graduate

students are excellent at trying new teaching techniques and reporting the effects on the grades. It is substantially more difficult to ask the deeper questions, which require focus groups, interviews, and detailed analytics, while juggling high teaching loads and being barred from serving as a student's research advisor. Tenure-track positions are necessary to give faculty the time to lead research groups, mentor graduate students, and provide jobs for excellent candidates.

CER/CS Research Collaborations

One of the exciting benefits of hiring CER researchers lies in the potential collaborations with existing computer science faculty. Computer scientists are in the perfect position to partner with CER researchers to accelerate the research process through automation, create new environments for learning, and create new curricula. Performing research in the space of Pasteur's Quadrant^a can change the world. Like computational biology, what begins as an application of the latest in computer science can grow into problems that drive computer science research.

Driving computer science and education research. Computer scientists can harness the power of automation, as long as they do not lose sight of the root questions: How do students learn, and how should we teach? The relatively new phenomenon of massive data collection (for example, MOOCs, Khan Academy) has distracted computer scientists with a new shiny set of data. Using project snapshots, machine learning can model the paths students take in solving their first programming assignment.³ Early struggles on this first assignment were correlated to future performance in the class. Tantalizing, but correlations are meaningless without understanding why they exist.

This represents a research methods gap. Small-scale research, with focus groups, direct observation, and other methods can answer *why* students follow certain development paths, identifying the mental models involved in order to inform curricu-

How do students learn, and how should we teach?

^a Research that is between pure basic and pure applied research; a quest for fundamental understanding that cares about the eventual societal use.

lar content and student feedback. Large-scale research can tell *how often* such problems occur, as well as identifying when they occur in real time. The real power, however, is in merging the two approaches; machine learning can identify anomalous actions in real time and send GSRs over to talk to those students in order to discover the conceptual cause of their development path.

Transforming society. Some may say our existing system produced successful computer scientists, so why should we spend this effort pushing the efforts to K–12? Our current flawed system has produced many fewer successful computer scientists than it could have. As Warren Buffet stated generously, one of the reasons for his great success was that he was competing with only half of the population (Sheryl Sandberg's *Lean In*). Research has shown that more diverse employees create better products, and companies want to compete. The work of Jane Margolis in *Unlocking the Clubhouse*² has inspired a generation of researchers to study how students with different backgrounds (gender, ethnic, socioeconomic) experience traditional computer science instruction. How should teaching methods, curriculum, IDEs, and other aspects look when designed for those neither confident in their abilities nor already viewing themselves as computer scientists? This mentality created Exploring Computer Science (ECS) and our interdisciplinary summer camp, Animal Tlatoque,¹ combining Mayan culture, animal conservation, art, and computer science. We specifically targeted female and Latina/os students who were not already interested in computer science. Our results after three summers were that the targeting through themes worked (95% female/minorities, 50% not interested in computer science), and that the Scratch-based camp resulted in increased interest (especially among those not already interested) and high self-efficacy. This challenge—to reach students who are not proactively seeking computer science—continues in our development of KELP CS for 4th–6th grades (perhaps the last time when computer science can be integrated into the normal classroom for all students).

Some have claimed coding skills improve problem-solving skills in other areas, but there is no research to back up the claim.

Understanding computer science's place in K–12. Another set of fundamental questions involves the relationship of computational thinking to the rest of education. Some have claimed coding skills improve problem-solving skills in other areas, but there is no research to back up the claim. Does learning programming, debugging, or project design help in other areas of development, such as logical thinking, problem solving, cause and effect, or empathy? Is time devoted to computer science instruction taking away from the fundamentals, is it providing an alternate motivation for students to build on the same skills, or is it providing a new set of skills that are necessary for innovators of the next century? These are fundamental questions that must be answered, and computer scientists have critical expertise to answer them, as well as a vested interest in the results.

Departmental Benefits

What will investing in computing education research bring the department? CER has promise with respect to teaching, funding, and department visibility.

Department Teaching. CER researchers, whether they perform research in K–12 or undergraduate education, often know about techniques to improve undergraduate teaching. Attending SIGCSE and ICER exposes researchers to the latest instructional and curricular techniques for undergraduate education, such as peer instruction⁴ and pair programming.⁵

Funding. The interdisciplinary nature

of computing education and diversity of the type of research (Kindergarten through college and theory through deployments) provides a plethora of funding opportunities in two directorates of the National Science Foundation (NSF): Educational and Human Resources (EHR) and Computer and Information Science and Engineering (CISE). With limited funding per program, CISE core calls clustered in November, and strict PI submission limits, such diverse offerings can be beneficial to a faculty member with a broad research portfolio.

External Visibility. Due to the repeated calls for more computer scientists, both at college and K–12 levels, teams that combine research and deployment can bring schools substantial visibility. In the past several years, computer science departments have made headlines for MOOCs, increasing female representation, and large-scale deployments in K–12 for the purposes of social equity, all areas in the CER domain.

Conclusion

The time is right to provide resources for computing education research. Computer science departments have sat on the sidelines for too long, reacting rather than leading. These efforts greatly affect our current and future students—the future of our field. Seize the opportunity now to make a mark on the future. C

References

1. Franklin, D., Conrad, P., Aldana, G. and Hough, S. Animal Tlatoque: Attracting middle school students to computing through culturally relevant themes. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, NY, 2011, 453–458.
2. Margolis, J. and Fisher, A. *Unlocking the Clubhouse*. MIT Press, Cambridge, MA, 2001.
3. Piech, C., Sahami, M., Koller, D., Cooper, S. and Blikstein, P. Modeling how students learn to program. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. ACM, NY, 2012, 153–160.
4. Simon, B., Parris, J. and Spacco, J. How we teach impacts student learning: Peer instruction vs. lecture in CS0. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*. ACM, NY, 2013, 41–46.
5. Williams, L. and Upchurch, R.L. In support of student pair-programming. In *Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education (SIGCSE '01)*. ACM, New York, NY, 2001, 327–331.

Diana Franklin (franklin@cs.ucsb.edu) is a tenure-equivalent teaching faculty member in the computer science department at the University of California, Santa Barbara.

Copyright held by author.



Kode Vicious Too Big to Fail

Visibility leads to debuggability.

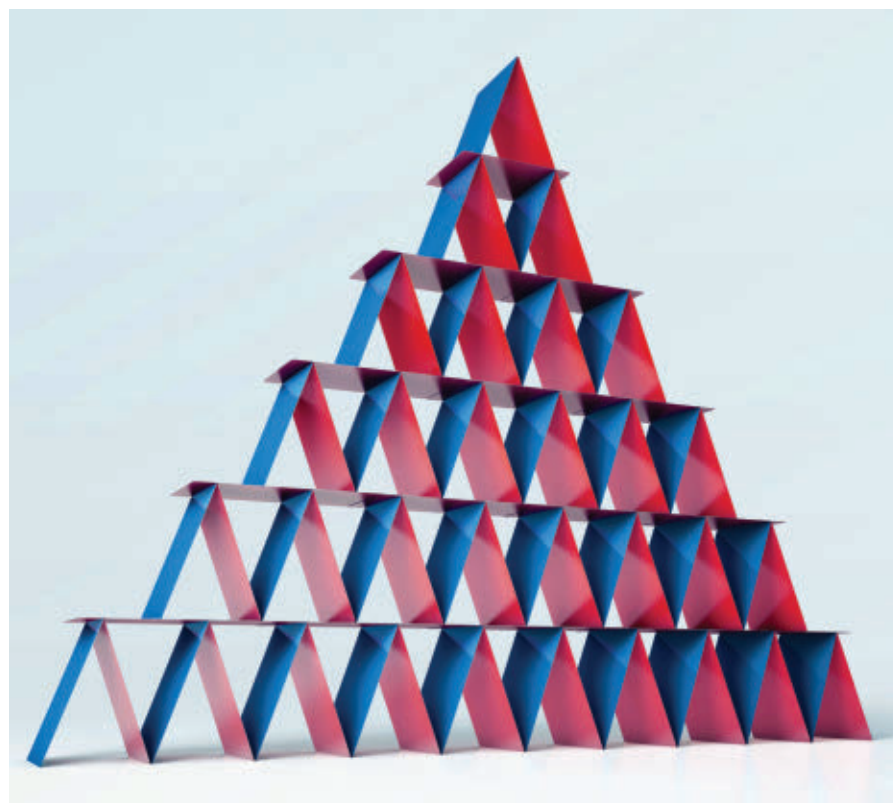
Dear KV,

Our project has been rolling out a well-known, distributed key/value store onto our infrastructure, and we have been surprised—more than once—when a simple increase in the number of clients has not only slowed things, but brought them to a complete halt. This then results in rollback while several of us scour the online forums to figure out if anyone else has seen the same problem. The entire reason for using this project's software is to increase the scale of a large system, so I have been surprised at how many times a small increase in load has led to a complete failure. Is there something about scaling systems that is so difficult that these systems become fragile, even at a modest scale?

Scaled Back

Dear Scaled,

If someone tells you that scaling out a distributed system is easy they are either lying or deranged—and possibly both. Anyone who has worked with distributed systems for more than a week should have this knowledge integrated into how they think, and if not, they really should start digging ditches. Not to say that ditch digging is easier but it does give you a nice, focused task that is achievable in a linear way, based on the amount of work you put into it. Distributed systems, on the other hand, react to increases in offered load in what can only politely be referred to as nondetermin-



istic ways. If you think programming a single system is difficult, programming a distributed system is a nightmare of Orwellian proportions where you almost are forced to eat rats if you want to join the party.

Non-distributed systems fail in much more predictable ways. Tax a single system and you run out of memory, or CPU, or disk space, or some other resource, and the system has little more than a snowball's chance surviving a Hawaiian holiday. The parts of the problem are so much

closer together and the communication between those components is so much more reliable that figuring out “who did what to whom” is tractable. Unpredictable things can happen when you overload a single computer, but you generally have complete control over all of the resources involved. Run out of RAM? Buy more. Run out of CPU, profile and fix your code. Too much data on disk? Buy a bigger one. Moore's Law is still on your side in many cases, giving you double the resources every 18 months.

ACM Transactions on Accessible Computing



This quarterly publication is a quarterly journal that publishes refereed articles addressing issues of computing as it impacts the lives of people with disabilities. The journal will be of particular interest to SIGACCESS members and delegates to its affiliated conference (i.e., ASSETS), as well as other international accessibility conferences.

www.acm.org/taccess
www.acm.org/subscribe



Association for
Computing Machinery

The problem is that eventually you will probably want a set of computers to implement your target system. Once you go from one computer to two, it is like going from a single child to two children. To paraphrase a joke, if you only have one child, it is not the same as having two or more children. Why? Because when you have one child and all the cookies are gone from cookie jar, *you know who did it!* Once you have two or more children, each has some level of plausible deniability. They can, and will, lie to get away with having eaten the cookies. Short of slipping your kids truth serum at breakfast every morning, you have no idea who is telling the truth and who is lying. The problem of truthfulness in communication has been heavily studied in computer science, and yet we still do not have completely reliable ways to build large distributed systems.

One way that builders of distributed systems have tried to address this problem is to put in somewhat arbitrary limits to prevent the system from ever getting too large and unwieldy. The distributed key store, Redis, had a limit of 10,000 clients that could connect to the system. Why 10,000? No clue, it is not even a typical power of 2. One might have expected 8,192 or 16,384, but that is probably a topic for another column. Perhaps the authors had been reading the *Tao Te Ching* and felt their universe only needed to contain 10,000 things. Whatever the reason, this seemed like a good idea at the time.

Of course the number of clients is only one way of protecting a distributed system against overload. What happens when a distributed system moves from running on 1Gbps network hardware to 10Gbps NICs? Moving from 1Gbps to 10Gbps does not “just” increase the bandwidth by an order of magnitude, it also reduces the request latency. Can a system with 10,000 nodes move smoothly from 1G to 10G? Good question, you would need to test or model that, but it is pretty likely a single limitation—such as number of clients—is going to be insufficient to prevent the system from getting into some very odd situations. Depending on how the overall system decides to parcel out work, you might wind up

with hot spots, places where a bunch of requests all get directed to a single resource, effectively creating what looks like a denial-of-service attack and destroying a node’s effective throughput. The system will then fail out that node and redistribute the work again, perhaps picking another target, and taking it out of the system because it looks like it, too, has failed. In the worst case, this continues until the entire system is brought to its knees and fails to make any progress on solving the original problem that was set for it.

Distributed systems that use a hash function to parcel out work are often dogged by this problem. One way to judge a hash function is by how well distributed the results of the hashing function are, based on the input. A good hash function for distributing work would parcel out work completely evenly to all nodes based on the input, but having a good hash function is not always good enough. You might have a great hash function, but feed it poor data. If the source data fed into the hash function does not have sufficient diversity (that is, it is relatively static over some measure, such as requests) then it does not matter how good the function is, as it still will not distribute work evenly over the nodes.

Take, for example, the traditional networking 4 tuple, source and destination IP address, and source and destination port. Together this is 96 bits of data, which seems a reasonable amount of data to feed the hashing function. In a typical networking cluster, the network will be one of the three well-known RFC 1918 addresses (192.168.0.0/16, 172.16.0.0/12, or 10.0.0.0/8). Let’s imagine a network of 8,192 hosts, because I happen to like powers of 2. Ignoring subnetting completely, we assign all 8,192

“The system is slow” is a poor bug report: in fact, it is useless.

hosts addresses from the 192.168.0.0 space, numbering them consecutively 192.168.0.1–192.168.32.1. The service being requested has a constant destination port number (for example, 6379) and the source port is ephemeral. The data we now put into our hash function are the two IPs and the ports. The source port is pseudo-randomly chosen by the system at connection time from a range of nearly 16 bits. It is nearly 16 bits because some parts of the port range are reserved for privileged programs, and we are building an underprivileged system. The destination port is constant, so we remove 16 bits of change from the input to the function. Those nice fat IPv4 addresses that should be giving us 64 bits of data to hash on actually only give us 13 bits, because that is all we need to encode 8,192 hosts. The input to our hashing function is not 96 bits, but is actually fewer than 42. Knowing that, you might pick a different hash function or change the inputs, inputs that really do lead to the output being spaced evenly over our hosts. How work is spread over the set of hosts in a distributed system is one of the main keys to whether that system can scale predictably, or at all.

An exhaustive discussion of how to scale distributed systems is a topic for a book far longer than this column, but I cannot leave the topic until I mention what debugging features exist in the distributed system. “The system is slow” is a poor bug report: in fact, it is useless. However, it is the one most often uttered in relation to distributed systems. Typically the first thing users of the system notice is the response time has increased and the results they get from the system take far longer than normal. A distributed system needs to express, in some way, its local and remote service times so the systems operators, such as the devops or systems administration teams, can track down the problem. Hot spots can be found through the periodic logging of the service request arrival and completion on each host. Such logging needs to be lightweight and not directed to a single host, which is a common mistake. When your system gets busy and the logging output starts taking out the servers, that’s bad. Recording system-level metrics, including CPU,

I am most surprised that some distributed systems work at all.

memory, and network utilization will also help in tracking down problems, as will the recording of network errors. If the underlying communications medium becomes overloaded, this may not show up on a single host, but will result in a distributed set of errors, with a small number at each node, which lead to chaotic effects over the whole system. Visibility leads to debuggability; you cannot have the latter without the former.

Coming back around to your original point, I am not surprised that small increases in offered load are causing your distributed system to fail, and, in fact, I am most surprised that some distributed systems work at all. Making the load, hot spots, and errors visible over the system may help you track down the problem and continue to scale it out even further. Or, you may find there are limits to the design of the system you are using, and you will have to either choose another or write your own. I think you can see now why you might want to avoid the latter at all costs.

KV

Related articles on queue.acm.org

KV the Loudmouth

George Neville-Neil

<http://queue.acm.org/detail.cfm?id=1255426>

There's Just No Getting around It: You're Building a Distributed System

Mark Cavage

<http://queue.acm.org/detail.cfm?id=2482856>

Corba: Gone But (Hopefully) Not Forgotten

Terry Coatta

<http://queue.acm.org/detail.cfm?id=1388786>

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

Copyright held by author.

Calendar of Events

February 2–6

Eighth ACM International Conference on Web Search and Data Mining, Shanghai, China, Sponsored: SIGMOD, SIGWEB, SIGIR, and SIGKDD, Contact: Hang Li, Email: hangli65@hotmail.com

February 7–11

20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Burlingame, CA, Sponsored: ACM/SIG, Contact: Albert Cohen, Email: albert.cohen@inria.fr

February 12–13

The 16th International Workshop on Mobile Computing Systems and Applications, Santa Fe, NM, Sponsored: ACM/SIG, Contact: Justin Gregory Manweiler, Email: justin.manweiler@gmail.com

February 18–21

Richard Tapia Celebration of Diversity in Computing Conference, Boston, MA, Sponsored: ACM/SIG, Contact: Valerie E. Taylor, Email: vtaylor@tamu.edu

February 22–24

The 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, Sponsored: ACM/SIG, Contact: George Constantinides, Email: g.constantinides@ic.ac.uk

February 27–March 1

I3D '15: Symposium on Interactive 3D Graphics and Games, San Francisco, CA, Sponsored: ACM/SIG, Contact: Li-Yi Wei, Email: liywei@stanfordalumni.org

March 2–4

Fifth ACM Conference on Data and Application Security and Privacy, San Antonio, TX, Sponsored: SIGSAC, Contact: Jaehong Park, Email: jaehpark@gmail.com

Viewpoint

Do-It-Yourself Textbook Publishing

Comparing experiences publishing textbooks using traditional publishers and do-it-yourself methods.

WE HAVE JUST survived an adventure in do-it-yourself (DIY) publishing by finishing a software-engineering textbook. Our goal was to produce a high-quality, popular textbook that was inexpensive while still providing authors a decent royalty. We can tell you DIY publishing *can* lead to a highly rated, low-priced textbook. (See the figure comparing the ratings and prices of our DIY textbook with the three software engineering textbooks from traditional publishers published this decade).^a Alas, as DIY marketing is still an open challenge, it is too early to know if DIY publishing can produce a popular textbook.^b As one of us (Patterson) has co-authored five editions of two textbooks over the last 25 years with a traditional publisher,^{2,3} we can give potential authors a lay of the two lands.

Challenges for Traditional Publishers

The past 25 years have been difficult for publishers. The more efficient



^a In May 2014, our text *Engineering Software as a Service: An Agile Approach Using Cloud Computing* was rated 4.5 out of 5 stars on Amazon.com, while the most popular competing books are rated 2.0 stars (*Software Engineering: A Practitioner's Approach* and *Software Engineering: Modern Approaches*) to 2.9 stars (*Software Engineering*).

^b There is a longer discussion of the DIY publishing in the online paper "Should You Self-Publish a Textbook? Should Anybody?"; <http://www.sasbook.info/about>.

used-book market, the importing of low-cost books printed for overseas markets, and piracy have combined to significantly reduce sales of new books or new editions. Since most of the costs of book are the labor costs of

development rather than the production costs of printing, the consequences have been to raise the prices of new books, to lower royalties to authors, and to force publishers to downsize. Obviously, higher prices make used

books, imported foreign editions, and piracy even more attractive to less-scrupulous readers and resellers, creating a vicious circle. Less obviously, publishers have laid off copy editors, indexers, graphic artists, typesetters, and so forth, many of whom have become independent contractors that are then hired by traditional publishers on an as-needed basis. This independence makes them available to DIY publishers as well. Indeed, we invested about \$10,000 to hire professionals to do all the work that we lacked the skills to do ourselves, such as cover design, graphic elements, and indexing.

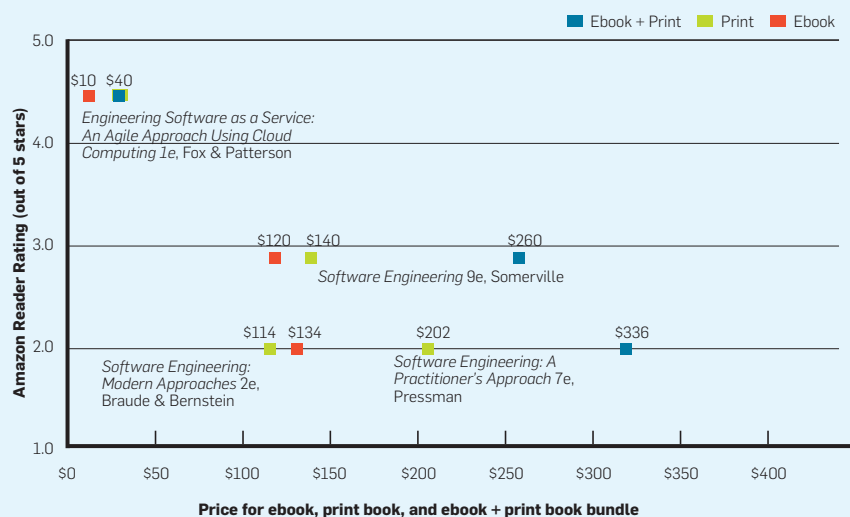
Note that the outsourcing has also made traditional publishing more error prone and slower; authors have to be much more vigilant to prevent errors from being inserted during the distributed bookmaking process, and the time between when an author is done with the draft and the book appears has stretched to nine months!

Technical Challenges of DIY Publishing

We clearly want to be able to produce both an electronic book (ebook) and a print book. One of us (Fox) created an open source software pipeline^c that can turn LaTeX into any ebook or print format. As long as authors are comfortable writing in LaTeX, the problem is now solved, although you must play with the text a bit to get figures and page breaks where you want them. While authors working with traditional publishers typically use simpler tools such as Microsoft Word, their text undergoes extensive human reprocessing, and it is distressingly easy for errors to creep in during the transcription process between file formats that many publishers now use. DIY authors must instead automate these tasks, and LaTeX is both up to the task and widely available to academic authors. There are many good choices for producing simple artwork; we used OmniGraffle for the figures we did ourselves.

While there are many options for ebooks, Amazon is the 800-pound go-

Ratings and prices on Amazon.com as of July 2014. The number of reviews are 2 (Braude and Bernstein), 38 (Fox and Patterson), 28 (Pressman), and 22 (Sommerville). Note the print+ebook bundle of Fox and Patterson costs the same as the print book only (\$40), and that Braude and Bernstein have no ebook.



Summary of self-publishing experience.

Writing effort	More work than writing with a publisher, and you must be self-motivated to stick to deadlines. Felt like a third job for both of us.
Tools	May be difficult to automate all the production tasks if you are not comfortable with LaTeX.
Book price for students	\$40/\$10 (print/ebook) in an era where \$100 printed textbooks and even \$100 e-textbooks are common.
Wide reach	Everywhere that Amazon does business, plus CreateSpace distributes to bookstores through traditional distributors such as Ingram.
Fast turnaround	Updated content available for sale 24 hours after completion; ebooks self-update automatically and (by our choice) free of charge.
Author income	On average, we receive about 50% of average selling price per copy; 15% would be more typical for a traditional publisher.
Piracy (print and ebook)	Unsolved problem for everyone, but by arrangement with individual companies, we have begun bundling services (Amazon credits, GitHub credits, and so forth) whose value exceeds the ebook's price, as a motivation to buy a legitimate copy.
Competitively priced print+ebook bundle	Amazon let us set our own price via "MatchBook" when the two are purchased together.
International distribution	Colleagues familiar with the publishing industry have told us that in some countries it is very difficult to get distribution unless you work with a publisher, so we are planning to work with publishers in those countries. Our colleagues have told us to expect the same royalty from those publishers as if they had worked on the whole book with us, even though we will be approaching them with a camera-ready translation already in hand.
Translation	The Chinese translation, handled by a traditional publisher, should be available soon. We are freelancing other translations under terms that give the translators a much higher royalty than they would receive as primary author from a publisher. We expect Spanish and Brazilian Portuguese freelanced translations to be available by spring 2015, and a Japanese translation later in the year.
Marketing/Adoption	The book is profitable but penetration into academia is slower than for Hennessy and Patterson's <i>Computer Architectures: A Quantitative Approach</i> in 1990. On the other hand, much has changed in the textbook ecosystem since then, so even if we achieve comparable popularity, we may never know all the reasons for the slower build.

^c See <http://bit.ly/1swgEsC>. There is also a description of the pipeline in the online paper mentioned previously in footnote b.

rilla of book selling. We tried several electronic portals with alpha and beta editions of our book,^d but 99% of sales went through Amazon, so we decided to just go with the Kindle format for the first edition. Note that you do not need to buy a Kindle ereader to read the Kindle format; Amazon makes free versions of software readers that run on all laptops, tablets, and smartphones.

Economics of DIY Publishing

Author royalties from traditional publishers today are typically 10% to 15% of the *net* price of the book (what the book seller pays, not the list price). Indeed, the new ACM Books program offers authors a 10% royalty, which is the going rate today.

For DIY publishing, as long as you keep the price of the ebook below \$10, Amazon offers authors 70% of the selling price of the book minus the downloading costs, which for our 5MB ebook was \$0.75. For ebooks costing more than \$10, the rate is 35%. Amazon is clearly encouraging prices for ebooks to be less than \$10, as the royalty is lower for books between \$10 and \$20. We selected CreateSpace for print books, which is a Print-On-Demand (POD) service now owned by Amazon. We chose it over the longer-established Lulu because independent authors' reviews of CreateSpace were generally more positive about customer service and turnaround time and because we assumed the connection to Amazon would result in quicker turnaround time until a finished draft was for sale on Amazon—it is 24 hours for CreateSpace. The royalty is a function of page count and list price. For our 500-page book printed in black-and-white with matte color covers, we get 35% of the difference between the book price and \$6.75, which is the cost of the POD book. The good news is that Amazon lets us bundle the ebook and print book together at a single lower price in some countries in which they operate.

One benefit of DIY publishing is that we are able to keep the prices

much lower than traditional textbooks and still get a decent royalty. We sell the ebook for \$10 and the bundle of the print book and ebook for \$40, making it a factor of 5 to 15 times cheaper than other software engineering books (see the figure).^e To get about the same royalty per book under a traditional publishing model, we'd need to raise the price of the ebook to at least \$40 and the price of the print book to at least \$60, which presumably would still sell OK given the high prices of the traditionally published textbooks.

Some have argued for community-developed online textbooks. We think that it might work for advanced graduate textbooks, where the audience is more sophisticated, happy to read longer texts, and more forgiving.^f We are skeptical it would work well for undergraduate textbooks, as it is important to have a clear, consistent perspective and vocabulary throughout the book. For undergraduates, what you omit is just as important as what you include: brevity is critical for today's students, who have much less patience for reading than earlier generations, and it is difficult for many authors to be collectively brief.

^e In July 2014 Amazon sold the print version of *Software Engineering: A Practitioner's Approach* for \$202 and the electronic version for \$134, or a total of \$336 for both. The total for *Software Engineering* is \$260 or \$120 for the ebook and \$140 for the print book.

^f For example, see *Software Foundations* by Pierce, B.C., Casinghino, C., Greenberg, M., Sjöberg, V., and Yorgey, B. (2010); <http://www.cis.upenn.edu/~bcpierce/sf/>.

One benefit of DIY publishing is that we are able to keep the prices much lower than traditional textbooks and still get a decent royalty.

Others have argued for free online textbooks.¹ We think capitalism works; the royalty from a successful traditional textbook can be as much as half a professor's salary. Making money gives authors a much stronger motivation to finish a book, keep the book up-to-date, and make constant improvements, which is critical in a fast moving field like ours. Indeed, we have updated the print book and ebook 12 times since January 2012. To put this into perspective, the experience of traditional publishers is that only 50% of authors who sign a contract ever complete a book, despite significant financial and legal incentives. Royalties also provide an income stream to employ others to help improve and complete the book. Our overall financial investment to produce and market the book was about \$12,000, so doing it for free may not make sense.

Marketing of DIY Publishing

Traditional publishers maintain publicity mailing lists, set up booths at conferences and trade shows, and send salespeople to university campuses, all of which can usually be amortized across multiple books. We quickly learned the hard way that setting up tables at conferences does not amortize well if you have only one book to sell, as it is difficult to attract people. We also spent approximately \$2,000 purchasing mailing lists, which once again has been outsourced and so now is available to DIY publishers. Fox had learned from his experience on the board of a community theater that a combination of email plus postcards works better than either one alone, so we had our designer create postcards to match the book's look and feel and we did a combined postcard-plus-email campaign. Alas, this campaign had modest impact, so it is unlikely we will do it again.

In addition, as academics we were already being invited to give talks at other universities, which gave us opportunities to expose the book as well; we would usually travel with a few "comp" copies. Of course, "comp" copies for self-publishers means we pay for these out of pocket, but we gave away a great many, since our faculty colleagues are the decision makers and a

^d We tried Amazon Kindle, the Apple iBooks, and the Barnes and Noble Nook device, which uses Epub format. We wanted to try using Google Play to distribute Epub, which is watermarked PDF, but its baffling user interface thwarted us.

physical copy of a book on your desk is more difficult to ignore.

Although we did not know it when we started the book (mid-2011), we were about to be offered the chance to adapt the first half of the campus course to a MOOC (Massive Open Online Course). MOOCs turned out to play a major role in the textbook's development. We accelerated our writing in order to have an "alpha edition" consisting of half of the content, that is, the chapters that would be most useful to the MOOC students. Indeed, based on advice from colleagues who had offered MOOCs, we were already structuring the MOOC as short video segments interspersed with self-check questions and assignments; we decided to mirror that structure in the book, with each section in a chapter mapping to a topical segment in the MOOC. While the book was only recommended and not required for the MOOC, the MOOC was instrumental in increasing the book's visibility. It also gave us class testing on steroids, as we got bug reports and comments from thousands of MOOC learners. Clearly, the MOOC helps with marketing, since faculty and practitioners enroll in MOOCs and they supply reviews on Amazon.

We have one cautionary tale about Amazon reviews. When we released the Alpha Edition, it was priced even lower than the current First Edition because more than one-third of the content had not yet been written (we wanted to release something in time for our MOOC, to get feedback from those students as well as our on-campus students). Despite repeated and prominent warnings in the book and in the Amazon book description about this fact, many readers gave us low reviews because content was missing. Based on reader feedback, we later decided to change the book's title, which also required changing the ISBN number and opening a new Amazon product listing. This switch "broke the chain" of reviews of previous editions and wiped the slate clean. This turned out well, since the vastly improved Beta edition received 4.5 stars out of 5 from Amazon readers, a high review level that has continued into the First Edition, while our main established competitors have far low-

We quickly learned the hard way that setting up tables at conferences does not amortize well if you have only one book to sell.

er Amazon reviews (see the figure). For better or worse, even people who purchase in brick-and-mortar stores rely on Amazon readers' reviews, so it was good to start from a clean slate after extensive changes. So one lesson is to change the ISBN number when transitioning out of an Alpha Edition.

We have learned two things through our "marketing" efforts. First, textbooks require domain-specific marketing: you have to identify and reach the very small set of readers who might be interested, so "mass" marketing techniques do not necessarily apply. This is relevant because many of the "marketing aids" provided by the Kindle author-facing portal are targeted at mass-market books and to authors who are likely to write more if successful, such as novelists. Second, in the past, publishers were the main source of information about new books, so your competition was similar titles from your publisher or other publishers; today, the competition has expanded to include the wealth of free information available online, an enormous used-book market, and so on, so the build-up may be slower. Indeed, the "Coca-Cola model" in which one brand dominates a field may not be an option for new arrivals.

Impact on Authors of DIY Publishing

As long as you do not mind writing in LaTeX, which admittedly is almost as much like programming as it is like writing, DIY publishing is wonderful for authors. The time between when we are done with a draft and people

can buy the book is *one day*, not nine months. We took advantage of this flexibility to make a series of alpha and beta versions of our book for class testing.

Moreover, when we find errors, we can immediately *update all ebooks already in the field* and we can immediately correct all future ebooks and print books. POD also means there is no warehouse of books that must be depleted before we can bring out a new edition, as is the case for most traditional publishers. Flexibility in correcting errors and bringing out new editions is attractive for any fast moving field, but it is *critical* in a software-intensive textbook like ours, since new releases of software on which the book relies can be incompatible with the book. Indeed, the First Edition was published in March 2014, and a new release from Heroku in May 2014 already requires changes to the "bookware" appendix. Such software velocity is inconsistent with a nine-month gap between what authors write and when it appears in print.

Conclusion

The resources are available today to produce a highly rated textbook, to do it more quickly than when working with traditional publishers, and to offer it at a price that is an order of magnitude lower. Given the marketing challenges, it is less obvious whether the book will be as popular as if we had gone with a traditional publisher, although a MOOC certainly helps. We may know in a few years if we are successful as DIY publishers; if the book never becomes popular, we may never know. But we are glad we went with DIY publishing, and we suspect others would be as well. C

References

1. Arpaci-Dusseau, R. The case for free online books (FOBs): Experiences with *Operating Systems: Three Easy Pieces*; <http://from-a-to-remzi.blogspot.com/2014/01/the-case-for-free-online-books-fobs.html>.
2. Hennessy, J. and Patterson, D. *Computer Architecture, 5th Edition: A Quantitative Approach*, 2012.
3. Patterson, D. and Hennessy, J. *Computer Organization and Design 5th Edition: The Hardware/Software Interface*, 2014.

Armando Fox (fox@cs.berkeley.edu) is a Professor of Computer Science at UC Berkeley and the Faculty Advisor to the UC Berkeley MOOC Lab.

David Patterson (patt@cs.berkeley.edu) holds the E.H. and M.E. Pardee Chair of Computer Science at UC Berkeley and is a past president of ACM.

Copyright held by authors.

Viewpoint

In Defense of Soundness: A Manifesto

Soundy is the new sound.

STATIC PROGRAM ANALYSIS is a key component of many software development tools, including compilers, development environments, and verification tools. Practical applications of static analysis have grown in recent years to include tools by companies such as Coverity, Fortify, GrammaTech, IBM, and others. Analyses are often expected to be *sound* in that their result models all possible executions of the program under analysis. Soundness implies the analysis computes an over-approximation in order to stay tractable; the analysis result will also model behaviors that do not actually occur in any program execution. The *precision* of an analysis is the degree to which it avoids such spurious results. Users expect analyses to be sound as a matter of course, and desire analyses to be as precise as possible, while being able to *scale* to large programs.

Soundness would seem essential for any kind of static program analysis. Soundness is also widely emphasized in the academic literature. Yet, in practice, soundness is commonly eschewed: we are not aware of a *single* realistic whole-program^a analysis tool (for example, tools widely used for bug detection, refactoring assistance, programming automation, and so forth)

^a We draw a distinction between whole program analyses, which need to model shared data, such as the heap, and modular analyses—for example, type systems. Although this space is a continuum, the distinction is typically well understood.



that does not purposely make unsound choices. Similarly, virtually all published whole-program analyses are unsound and omit conservative handling of common language features when applied to *real programming languages*.

The typical reasons for such choices are engineering compromises: implementers of such tools are well aware of how they could handle complex language features soundly (for example, assuming that a complex language feature can exhibit *any* behavior), but do not do so because this would make the analysis *unscalable* or *imprecise* to the point of being useless. Therefore, the

dominant practice is one of treating soundness as an engineering choice.

In all, we are faced with a paradox: on the one hand we have the ubiquity of unsoundness in any practical whole-program analysis tool that has a claim to precision and scalability; on the other, we have a research community that, outside a small group of experts, is oblivious to any unsoundness, let alone its preponderance in practice.

Our observation is that the paradox can be reconciled. The state of the art in realistic analyses exhibits consistent traits, while also integrating a sharp discontinuity. On the one hand, typical

realistic analysis implementations have a *sound core*: most common language features are *over-approximated*, modeling all their possible behaviors. Every time there are multiple options (for example, branches of a conditional statement, multiple data flows) the analysis models all of them. On the other hand, some specific language features, well known to experts in the area, are best *under-approximated*. Effectively, every analysis pretends perfectly possible behaviors cannot happen. For instance, it is conventional for an otherwise sound static analysis to treat highly dynamic language constructs, such as Java reflection or *eval* in JavaScript, under-approximately. A practical analysis, therefore, may pretend that *eval* does nothing, unless it can precisely resolve its string argument at compile time.

We introduce the term *soundy* for such analyses. The concept of *soundiness* attempts to capture the balance, prevalent in practice, of over-approximated handling of most language features, yet deliberately under-approximated handling of a feature subset well recognized by experts. Soundiness is in fact what is meant in many papers that claim to describe a sound analysis. A *soundy* analysis aims to be as sound as possible without excessively compromising precision and/or scalability.

Our message here is threefold:

- We bring forward the ubiquity of, and engineering need for, unsoundness in the static program analysis practice. For static analysis researchers, this may come as no surprise. For the rest of the community, which expects to use analyses as a black box, this unsoundness is less understood.

- We draw a distinction between analyses that are soundy—mostly sound, with specific, well-identified unsound choices—and analyses that do not concern themselves with soundness.

- We issue a call to the community to identify clearly the nature and extent of unsoundness in static analyses. Currently, in published papers, sources of unsoundness often lurk in the shadows, with caveats only mentioned in an off-hand manner in an implementation or evaluation section. This can lead a casual reader to erroneously conclude the analysis is sound. Even worse, elided details of how tricky language constructs are handled could have a

Soundness is not even necessary for most modern analysis applications, however, as many clients can tolerate unsoundness.

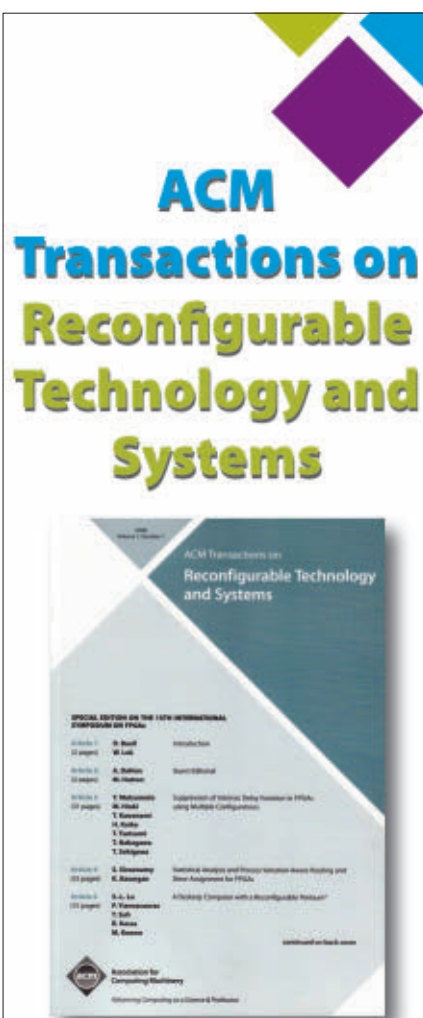
profound impact on how the paper's results should be interpreted, since an unsound handling could lead to much of the program's behavior being ignored (consider analyzing large programs, such as the Eclipse IDE, without understanding at least something about reflection; most of the program will likely be omitted from analysis).

Unsoundness: Inevitable and, Perhaps, Desirable?

The typical (published) whole-program analysis extolls its scalability virtues and briefly mentions its soundness caveats. For instance, an analysis for Java will typically mention that reflection is handled “as in past work,” while dynamic loading will be (silently) assumed away, as will be any behavior of opaque, non-analyzed code (mainly native code) that may violate the analysis' assumptions. Similar “standard assumptions” hold for other languages. Indeed, many analyses for C and C++ do not support casting into pointers, and most ignore complex features such as `setjmp/longjmp`. For JavaScript the list of caveats grows even longer, to include the `with` construct, dynamically computed fields (called properties), as well as the notorious `eval` construct.

Can these language features be ignored without significant consequence? Realistically, most of the time the answer is no. These language features are nearly ubiquitous in practice. Assuming the features away excludes the majority of input programs. For example, very few JavaScript programs larger than a certain size omit at least occasional calls to `eval`.

Could all these features be modeled



◆ ◆ ◆ ◆ ◆

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

◆ ◆ ◆ ◆ ◆

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

 Association for Computing Machinery

Some of the sources of unsoundness, sorted by language.

Language	Examples of commonly ignored features	Consequences of not modeling these features
C/C++	setjmp/longjmp ignored effects of pointer arithmetic “manufactured” pointers	ignores arbitrary side effects to the program heap
Java/C#	Reflection JNI	can render much of the codebase invisible for analysis “invisible” code may create invisible side effects in programs
JavaScript	eval, dynamic code loading data flow through the DOM	missing execution missing data flow in program

soundly? In principle, yes. In practice, however, we are not aware of a single sound whole-program static analysis tool applicable to industrial-strength programs written in a mainstream language! The reason is sound modeling of all language features usually destroys the precision of the analysis because such modeling is usually highly over-approximate. Imprecision, in turn, often destroys scalability because analysis techniques end up computing huge results—a typical modern analysis achieves scalability by maintaining precision, thus minimizing the datasets it manipulates.

Soundness is not even *necessary* for most modern analysis applications, however, as many clients can tolerate unsoundness. Such clients include IDEs (auto-complete systems, code navigation), security analyses, general-purpose bug detectors (as opposed to program verifiers), and so forth. Even automated refactoring tools that perform code transformation are unsound in practice (especially when concurrency is considered), and yet they are still quite useful and implemented in most IDEs. Third-party users of static analysis results—including other research communities, such as software engineering, operating systems, or computer security—have been highly receptive of program analyses that are unsound, yet useful.

Evaluating Sources of Unsoundness by Language

While an unsound analysis may take arbitrary shortcuts, a sound analysis that attempts to *do the right thing* faces some formidable challenges. In particular, unsoundness frequently stems from difficult-to-model language features. In the accompanying table, we list some of the sources of unsoundness, which we segregate by language.

All features listed in the table can have significant consequences on the program, yet are commonly ignored at analysis time. For language features that are most often ignored in unsound analyses (reflection, setjmp/longjmp, eval, and so forth), more studies should be published to characterize how extensively these features are used in typical programs and how ignoring these features could affect standard program analysis clients. Recent work analyzes the use of eval in JavaScript. However, an informal email and in-person poll of recognized experts in static and runtime analysis failed to pinpoint a single reliable survey of the use of so-called *dangerous* features (pointer arithmetic, unsafe type casts, and so forth) in C and C++.

Clearly, an improved evaluation methodology is required for these unsound analyses, to increase the comparability of different techniques. Perhaps, benchmarks or regression suites could be assembled to measure the effect of unsoundness. While further work is required to devise such a methodology in full, we believe that, at the least, some effort should be made in experimental evaluations to compare results of an unsound analysis with observable dynamic behaviors of the program. Such empirical evaluation would indicate whether important behaviors are being captured. It really does not help the reader for the analysis' author to declare that their analysis is sound modulo features X and Y, only to discover that these features are present in just about every real-life program! For instance, if a static analysis for JavaScript claims to be “sound modulo eval,” a natural question to ask is whether the types of input program this analysis expects do indeed use eval in a way that is highly non-trivial.

Moving Forward

We strongly feel that:

- The programming language research community should embrace sound analysis techniques and tune its soundness expectations. The notion of soundness can influence not only tool design but also that of programming languages or type systems. For example, the type system of TypeScript is unsound, yet practically very useful for large-scale development.

- *Soundy is the new sound; de facto*, given the research literature of the past decades.

- Papers involving sound analyses should both explain the general implications of their unsoundness and evaluate the implications for the benchmarks being analyzed.

- As a community, we should provide guidelines on how to write papers involving sound analysis, perhaps varying per input language, emphasizing which features to consider handling—or not handling. 

Benjamin Livshits (livshits@microsoft.com) is a research scientist at Microsoft Research.

Manu Sridharan (manu@sridharan.net) is a senior staff engineer at Samsung Research America.

Yannis Smaragdakis (smaragd@di.uoa.gr) is an associate professor at the University of Athens.

Ondřej Lhoták (olhotak@uwaterloo.ca) is an associate professor at the University of Waterloo.

J. Nelson Amaral (jamaral@ualberta.ca) is a professor at the University of Alberta.

Bor-Yuh Evan Chang (evan.chang@Colorado.EDU) is an assistant professor at the University of Colorado Boulder.

Samuel Z. Guyer (sguyer@cs.tufts.edu) is an associate professor at Tufts University.

Uday P. Khedker (uday@cse.iitb.ac.in) is a professor at the Indian Institute of Technology Bombay.

Anders Møller (amoeller@cs.au.dk) is an associate professor at Aarhus University.

Dimitrios Vardoulakis (dimvar@google.com) is a software engineer at Google Inc.

Copyright held by authors.

Systor2015

The 8th ACM International Systems
and Storage Conference

May 26 – 28 • Haifa, Israel

We invite you to submit original and innovative papers, covering all aspects of computer systems technology, such as file and storage technology; operating systems; distributed, parallel, and cloud systems; security; virtualization; and fault tolerance, reliability, and availability. SYSTOR 2015 accepts both full-length and short papers.

Paper submission deadline: March 5, 2015

Program committee chairs

Gernot Heiser, NICTA and UNSW,
Australia

Idit Keidar, Technion

General chair

Dalit Naor, IBM Research

Posters chair

David Breitgand, IBM Research

Steering committee head

Michael Factor, IBM Research

Steering committee

Ethan Miller, University of California
Santa Cruz

Liuba Shrira, Brandeis University

Dan Tsafir, Technion

Yaron Wolfsthal, IBM

Erez Zadok, Stony Brook University

www.systor.org/2015/

Sponsored by



SIGOPS

In cooperation with



usenix
THE ADVANCED
COMPUTING SYSTEMS
ASSOCIATION



Platinum sponsor



Gold sponsors



Sponsors



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

Crackers discover how to use NTP as a weapon for abuse.

BY HARLAN STENN

Securing Network Time Protocol

IN THE LATE 1970s David L. Mills began working on the problem of synchronizing time on networked computers, and Network Time Protocol (NTP) version 1 made its debut in 1980. This was when the Net was a much friendlier place—the ARPANET days. NTP version 2 appeared approximately one year later, about the same time as Computer Science Network (CSNET). National Science Foundation Network (NSFNET) launched in 1986. NTP version 3 showed up in 1993.

Depending on where you draw the line, the Internet became useful in 1991–1992 and fully arrived in 1995. NTP version 4 appeared in 1997. Now, 18 years later, the Internet Engineering Task Force (IETF) is almost done finalizing the NTP version 4 standard, and some of us are starting to think about NTP version 5.

All of this is being done by volunteers—with no budget, just by the good graces of companies and individuals who care. This is not a sustainable situation. Network Time Foundation (NTF) is the

vehicle that can address this problem, with the support of other organizations and individuals. For example, the Linux Foundation's Core Infrastructure Initiative recently started partially funding two NTP developers: Poul-Henning Kamp for 60% of his available time to work on NTP, and me for 30%–50% of my NTP development work. (Please visit <http://nwtime.org/> to see who is supporting Network Time Foundation.)

On the public Internet, NTP tends to be visible from three types of machines. One is in embedded systems. When shipped misconfigured by the vendor, these systems have been the direct cause of abuse (http://en.wikipedia.org/wiki/NTP_server_misuse_and_abuse). These systems do not generally support external monitoring, so they are not generally abusable in the context of this article. The second set of machines would





IMAGE BY RENE JANSZ

be routers, and the majority of the ones that run NTP are from Cisco and Juniper. The third set of machines tend to be Windows machines that run win32time (which does not allow monitoring, and is therefore neither monitorable, nor abusable in this context), and Unix boxes that run NTP, acting as local time servers and distributing time to other machines on the LAN that run NTP to keep the local clock synchronized.

For the first 20 years of NTP's history, these local time servers were often old, spare machines that ran a dazzling array of operating systems. Some of these machines kept much better time than others, and people would eventually run them as their master time servers. This is one of the main reasons the NTP codebase stuck with K&R C (named for its authors Brian Kernighan and Dennis Ritchie) for so

many years, as that was the only easily available compiler on some of these older machines.

It was not until December 2006 that NTP upgraded its codebase from K&R C to ANSI C. For a good while, only C89 was required. This was a full six years beyond Y2K, when a lot of these older operating systems were obsolete but still in production. By this time, however, the hardware on which NTP was "easy" to run had changed to x86 gear, and gcc (GNU Compiler Collection) was the easy compiler choice.

The NTP codebase does its job very well, is very reliable, and has had an enviable record as far as security problems go. Companies and people often run ancient versions of this software on some embedded systems that effectively never get upgraded and run well enough for a very long time.

People just expect accurate time, and they rarely see the consequences of inaccurate time. If the time is wrong, it is often more important to fix it fast and then—maybe—see if the original problem can be identified. The odds of identifying the problem increase if it happens with any frequency. Last year, NTP and our software had an estimated one *trillion* hours plus of operation. We have received some bug reports over this interval, and we have some open bug reports we would love to resolve, but in spite of this, NTP generally runs very, very well.

Having said all of this, I should re-emphasize that NTP made its debut in a much friendlier environment, and that if there was a problem with the time on a machine, it was important to fix the problem as quickly as possible. Over the years, this translated

into making it easy for people to query an NTP instance to see what it has been doing. There are two primary reasons for this: one is so it is easy to see if the remote server you want to sync with is configured and behaving adequately; the other is so it is easy to get help from others if there is a problem.

While we have been taking steps over the years to make NTP more secure and immune to abuse, the public Internet had more than seven million abusable NTP servers in the fall of last year. As a result of people upgrading software, fixing configuration files, or because, sadly, some ISPs and IXPs have decided to block NTP traffic, the number of abusable servers has dropped by almost 99% in just a few months. This is a remarkably large and fast decline, until you realize that around 85,000 abusable servers still exist, and a DDoS (distributed denial-of-service) attack in the range of 50Gbps–400Gbps can be launched using 5,000 servers. There is still a lot of cleanup to be done.

One of the best and easiest ways of reducing and even eliminating DDoS attacks is to ensure computers on your networks send packets that come from only *your* IP space. To this end, you should visit <http://www.bcp38.info/> and take steps to implement this practice for your networks, if you have not already done so.

As I mentioned, NTP runs on the

public Internet in three major places: Embedded devices; Unix and some Windows computers; and Cisco and Juniper routers. Before we take a look at how to configure the latter two groups so they cannot be abused, let's look at the NTP release history.

NTP Release History

David L. Mills, now a professor emeritus and adjunct professor at the University of Delaware, gave us NTP version 1 in 1980. It was good, and then it got better. A new “experimental” version, xntp2, installed the main binary as xntpd, because, well, that was the easy way to keep the previous version and new version on a box at the same time. Then version 2 became stable and a recommended standard (RFC 1119), so work began on xntp3. But the main program was still installed as xntpd, even though the program was not really “experimental.” Note that RFC-1305 defines NTPv3, but that standard was never finalized as a recommended standard—it remained a draft/elective standard. The RFC for NTPv4 is still in development but is expected to be a recommended standard.

As for the software release numbering, three of the releases from Mills are xntp3.3wx, xntp3.3wy, and xntp3.5f. These date from just after the time I started using NTP heavily, and I was also sending in portabil-

ity patches. Back then, you unpacked the tarball, manually edited a config. local file, and did interesting things with the makefile to get the code to build. While Perl's metaconfig was available then and was great for poking around a system, it did not support subdirectory builds and thus could not use a single set of source code for multiple targets.

GNU autoconf was still pretty new at that time, and while it did not do nearly as good a job at poking around, it did support subdirectory builds. xntp3.5f was released just as I volunteered to convert the NTP code base to GNU autoconf. As part of that conversion, Mills and I discussed the version numbers, and he was OK with my releasing the first cut of the GNU autoconf code as xntp3-5.80. These were considered alpha releases, as .90 and above were reserved for beta releases. The first production release for this code would be xntp3-6.0, the sixth major release of NTPv3, except that shortly after xntp3-5.93e was released in late November 1993, Mills decided the NTPv3 code was good enough and that it was time to start on NTPv4.

At that point, I noticed many people had problems with the version-numbering scheme, as the use of both the dash (-) and dot (.) characters really confused people. So ntp-4.0.94a was the first beta release of the NTPv4 code in July 1997. The release numbers went from ntpPROTO-Maj.Min to ntp-PROTO.Maj.Min.

While this change had the desired effect of removing confusion about how to type the version number, it meant most people did not realize going from ntp-4.1.x to 4.2.x was a *major* release upgrade. People also did not seem to understand just how many items were being fixed or improved in minor releases. For more information about this, see the accompanying table.

At one point I tried going back to a version-numbering scheme that was closer to the previous method, but I got a lot of pushback so I did not go through with it. In hindsight, I should have stood my ground. Having seen how people do not appreciate the significance of the releases—major or minor—we will go back to a numbering scheme much closer to the original after 4.2.8 is released.

NTP release history.			
Version	Release Date	EOL Date	# Bugs fixes and Improvements
ntp-4.2.8	Dec 2014		Over 1100 so far
ntp-4.2.6	Dec 2009	Dec 2014	630-1000
ntp-4.2.4	Dec 2006	Dec 2009	Over 450
ntp-4.2.2	Jun 2006	Dec 2006	Over 560
ntp-4.2.0	Oct 2003	Jun 2006	?
ntp-4.1.2	Jul 2003	Oct 2003	?
ntp-4.1.1	Feb 2002	Jul 2003	?
ntp-4.1.0	Aug 2001	Feb 2002	?
ntp-4.0.99	Jan 2000	Aug 2001	?
ntp-4.0.90	Nov 1998	Jan 2000	?
ntp-4.0.73	Jun 1998	Nov 1998	?
ntp-4.0.72	Feb 1998	Jun 1998	?
ntp-4.0	Sep 1997	Feb 1998	?
xntp3-5.86.5	Oct 1996	Sep 1997	?
xntp3.5f	Apr 1996	Oct 1996	?
xntp3.3wy	Jun 1994	Apr 1996	?
xntp3	Jun 1993	Jun 1994	?
xntp2	Nov 1989	Jun 1993	?

The major release after ntp-4.2.8 will be something like ntp4-5.0.0 (or ntp-PROTO-Maj.Min.Point, if we keep the Major Minor numbers) or ntp4-3.0 (or ntpPROTO-Release.Point, if we go to a single release number from the current Major and Minor release numbers). Our source archives reveal how the release numbering choices have evolved over the years, and how badly some of them collated.

Securing NTP

Before we delve into how to secure NTP, I recommend you listen to Dan Geer's keynote speech for Blackhat 2014, if you have not already done so (<https://www.youtube.com/watch?v=nT-TGvYOBpI>). It will be an excellent use of an hour of your time. If you watch it and disagree with what he says, then I wonder why you are reading this article to look for a solution to NTP abuse vector problems.

Now, to secure NTP, first implement BCP38 (<http://www.bcp38.info>). It is not that difficult.

If you want to ensure NTP on your Cisco or Juniper routers is protected, then consult their documentation on how to do so. You will find lots of good discussions and articles on the Web with additional updated information, and I recommend <http://www.team-cymru.org/ReadingRoom/Templates/secure-ntp-template.html> for information about securing NTP on Cisco and Juniper routers.

The NTP support site provides information on how to secure NTP through the ntp.conf file. Find some discussion and a link to that site at <http://nwtime.org/ntp-winter-2013-network-drdoS-attacks/>. NTF is also garnering the resources to produce an online ntp.conf generator that will implement BCP for this file and make it easy to update that file as our codebase and knowledge evolves.

That said, the most significant NTP abuse vectors are disabled by default starting with ntp-4.2.7p27, and these and other improvements will be in ntp-4.2.8, which was released at press time.

For versions 4.2.6 through 4.2.7p27, this abuse vector can be prevented by adding the following to your ntp.conf file:

```
restrict default ... noquery ...
```

Note that if you have additional restrict lines for IPs or networks that

do *not* include noquery restriction, ask yourself if it is possible for those IPs to be spoofed.

For version 4.2.4, which was released in December 2006 and EOled (brought to the end-of-life) in December 2009, consider the following:

- ▶ You did not pay attention to what Dan Geer said.

- ▶ Did you notice we fixed 630-1,000 issues going from 4.2.4 to 4.2.6?

- ▶ Are you still interested in running 4.2.4? Do you really have a good reason for this?

If so, add to your ntp.conf file:

```
restrict default ... noquery ...
```

For version 4.2.2, which was released in June 2006 and EOled in December 2006:

- ▶ You did not pay attention to what Dan Geer said.

- ▶ Did you notice we fixed about 450 issues going from 4.2.2 to 4.2.4, and 630-1,000 issues going from 4.2.4 to 4.2.6? That is between 1,000 and 1,500 issues. Seriously.

- ▶ Are you still interested in running 4.2.2? Do you really have a good reason for this?

If so, add to your ntp.conf file:

```
restrict default ... noquery ...
```

For version 4.2.0, which was released in 2003 and EOled in 2006:

- ▶ You did not pay attention to what Dan Geer said.

- ▶ Did you notice we fixed about 560 issues going from 4.2.0 to 4.2.2, 450 issues going from 4.2.2 to 4.2.4, and 630-1,000 issues going from 4.2.4 to 4.2.6? That is between 1,500 and 2,000 issues. Seriously.

- ▶ Are you still interested in running 4.2.2? Do you really have a good reason for this?

If so, add to your ntp.conf file:

```
restrict default ... noquery ...
```

For versions 4.0 through 4.1.1, which were released and EOled somewhere around 2001 to 2003, no numbers exist for how many issues were fixed in these releases:

- ▶ You did not pay attention to what Dan Geer said.

- ▶ There are probably in excess of

2,000-2,500 issues fixed since then.

- ▶ Are you still interested in running 4.0 or 4.1 code? Do you really have a good reason for this?

If so, add to your ntp.conf file:

```
restrict default ... noquery ...
```

Now let's talk about xntp3, which was EOled in September 1997. Do the math on how old that is, take a guess at how many issues have been fixed since then, and ask yourself and anybody else who has a voice in the matter: Why are you running software that was EOled 17 years ago, when thousands of issues have been fixed and an improved protocol has been implemented since then?

If your answer is: "Because NTPv3 was a standard and NTPv4 is not yet a standard," then I have bad news for you. NTPv3 was *not* a recommended standard; it was only a draft/elective standard. If you really want to run only officially standard software, you can drop back to NTPv2—and I do not know anybody who would want to do that.

If your answer is: "We're not sure how stable NTPv4 is," then I will point out that NTPv4 has an estimated 5-10 *trillion* operational hours at this point. How much more do you want?

But if you insist, the way to secure xntp2 and xntp3 against the described abuse vector is to add to your ntp.conf file:

```
restrict default ... noquery ...
```

Related articles on queue.acm.org

Principles of Robust Timing over the Internet

Julien Ridoux and Darryl Veitch
<http://queue.acm.org/detail.cfm?id=1773943>

Toward Higher Precision

Rick Ratzel and Rodney Greenstreet
<http://queue.acm.org/detail.cfm?id=2354406>

The One-second War (What Time Will You Die?)

Poul-Henning Kamp
<http://queue.acm.org/detail.cfm?id=1967009>

Harlan Stenn is president of Network Time Foundation in Talent, OR, and project manager of the NTP Project.

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



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

MBT has positive effects on efficiency and effectiveness, even if it only partially fulfills high expectations.

BY ROBERT V. BINDER, BRUNO LEGEARD, AND ANNE KRAMER

Model-Based Testing: Where Does It Stand?

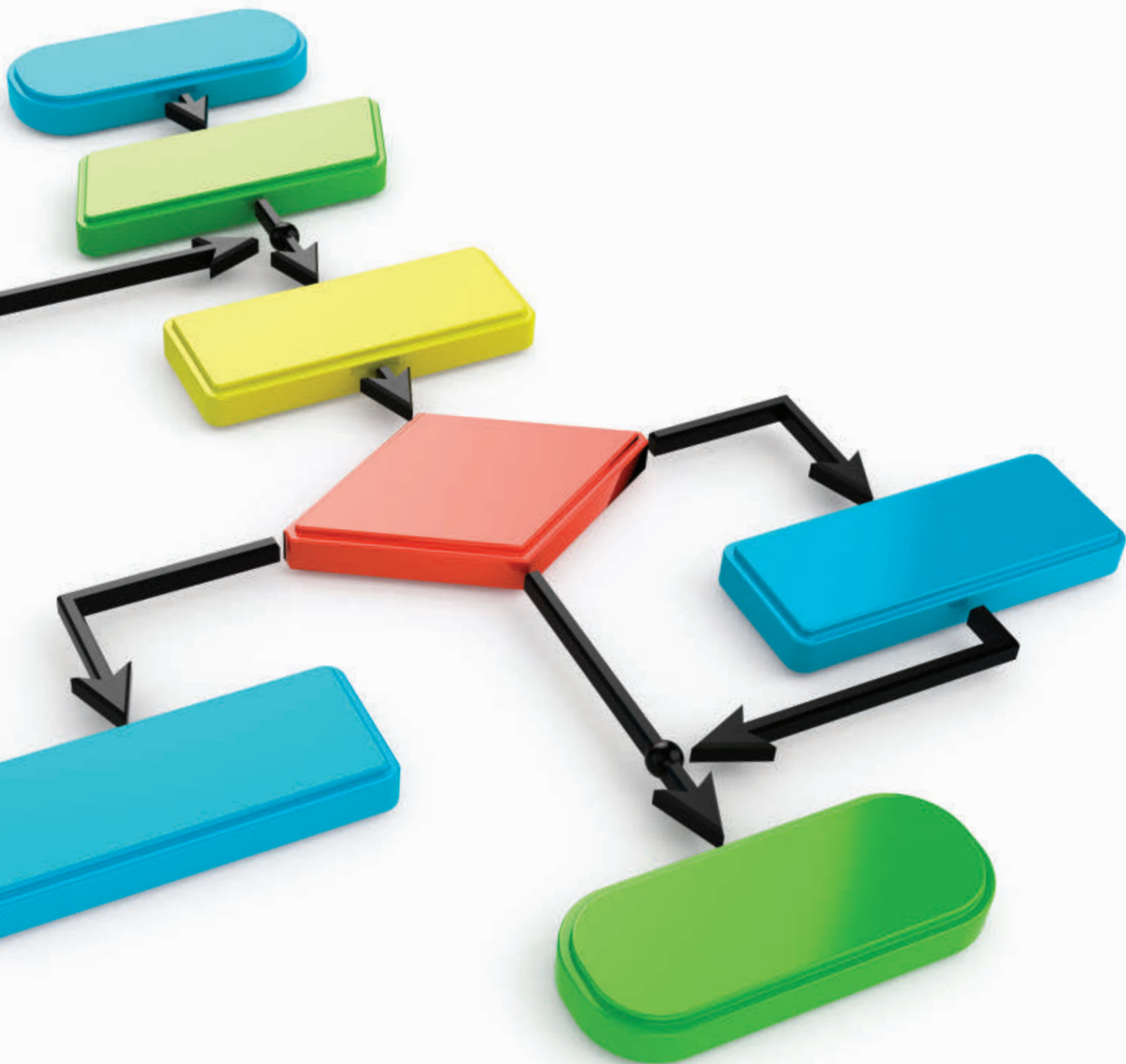
YOU HAVE PROBABLY heard about model-based testing (MBT), but like many software-engineering professionals who have not used MBT, you might be curious about others' experience with this test-design method.

From mid-June 2014 to early August 2014, we conducted a survey to learn how MBT users view its efficiency and effectiveness. The 2014 MBT User Survey, a follow-up to a similar 2012 survey (<http://robertvbinder.com/real-users-of-model-based-testing/>), was open to all those who have evaluated or used any MBT approach. Its 32 questions included some from a survey distributed at the 2013 User Conference on Advanced Automated Testing. Some questions focused on the efficiency and effectiveness of MBT, providing the figures that managers are most interested in. Other questions were

more technical and sought to validate a common MBT classification scheme. As there are many significantly different MBT approaches, beginners are often confused. A common classification scheme could help users understand both the general diversity and specific approaches.

The 2014 survey provides a realistic picture of the current state of MBT practice. This article presents some highlights of the survey findings. The





complete results are available at <http://model-based-testing.info/2014/12/09/2014-mbt-user-survey-results/>.

Survey Respondents

A large number of people received the 2014 MBT User Survey call for participation, both in Europe and North America. Additionally, it was posted with various social-networking groups and software-testing forums. Several tool providers helped

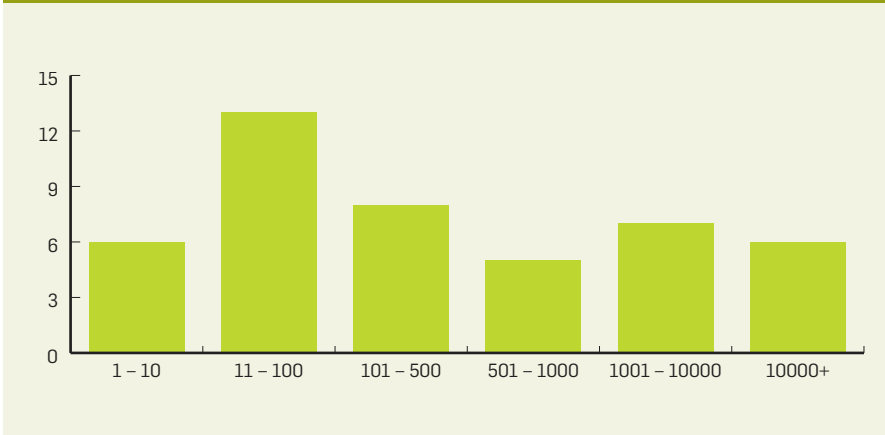
distribute the call. Last but not least, European Telecommunications Standards Institute (ETSI) supported the initiative by informing all participants at the User Conference on Advanced Automated Testing.

Exactly 100 MBT practitioners responded by the closing date. Not all participants answered every question; the number of answers is indicated if considerably below 100. Percentages for these partial response sets

are based on the actual number of responses for a particular question.

The large majority of the respondents (86%) were from businesses. The remaining 14% represented research, government, and nonprofit organizations. The organizations ranged in size from three to 300,000 employees (Figure 1).

As shown in Figure 2, almost half of the respondents had moved from evaluation and pilot to rollout or generalized use. On average, respon-

Figure 1. Survey participants come from organizations of all sizes.

dents had three years of experience with MBT. In fact, the answers ranged from zero (meaning “just started”) to 34 years.

To get an impression of how important MBT is with respect to other test-design techniques, the survey asked for the percentage of total testing effort spent on MBT, hand-coded test automation, and manual test design. Each of the three test-design methods represented approximately one-third of the total testing effort. Thus, MBT is not a marginal phenomenon. For those who use it, its importance is comparable to other kinds of test automation.

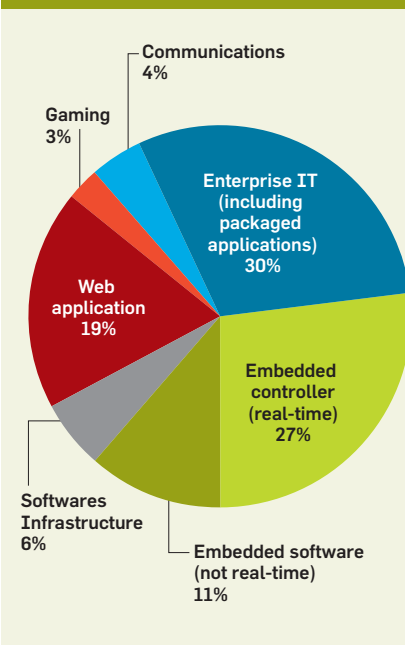
Nearly 40% of the respondents came from the embedded domain. Enterprise IT accounted for another 30%, and Web applications for approximately 20%. Other application domains for

the system under test were software infrastructure, communications, gaming, and even education. The exact distribution is given in Figure 3.

The role of external MBT consultants turned out to be less influential than expected. Although we initially speculated that MBT is driven mainly from the outside, the survey showed a different picture. A majority (60%) of those who answered the survey were in-house MBT professionals. Only 18% of respondents were external MBT consultants, and 22% were researchers in the MBT application area (Figure 4).

Does MBT Work as Expected?

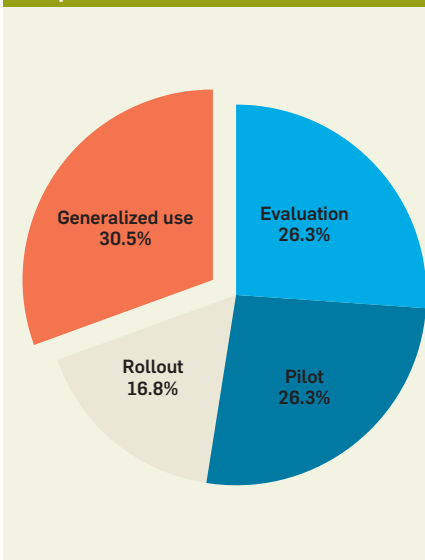
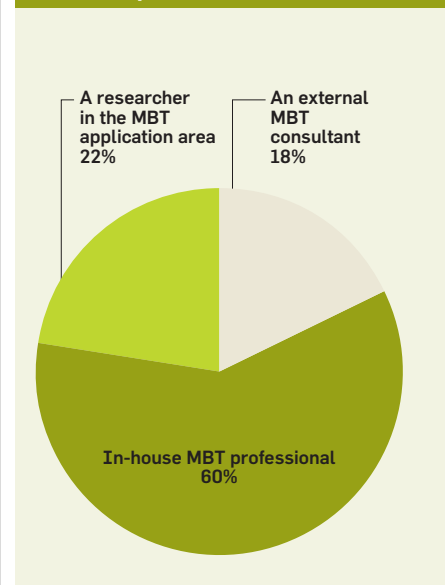
In our projects, we observed the expectations regarding MBT are usually

Figure 3. Various application domains represented.

very, if not extremely, high. The MBT user survey asked whether the expectations in five different categories are being met: testing is becoming cheaper; testing is better; the models are helping manage the complexity of the system with regard to testing; the test design is starting earlier; and, finally, models are improving communication among stakeholders.

So, we asked: “Does MBT fulfill expectations?” Figure 5 shows the number of responses and the degree of satisfaction for each of the five categories. The answers reflect a slight disenchantment. MBT does not completely fulfill the extremely high expectations in terms of cost reduction, quality improvement, and complexity, but, still, more than half of the respondents were partially or completely satisfied (indicated by the green bars in Figure 5). For the two remaining categories, MBT even exceeds expectations. Using models for testing purposes definitely improves communication among stakeholders and helps initiate test design earlier.

Overall, the respondents viewed MBT as a useful technology: 64% found it moderately or extremely effective, whereas only 13% rated the method as ineffective (Figure 6). More than 70% of the respondents stated it is very likely or extremely likely they will continue with the method. Only one respondent out of 73 rejected this idea. Participants self-selected, however, so we

Figure 2. 48% of the respondents routinely use MBT, 52% are still in the evaluation or trial phase.**Figure 4. Three in five respondents are in-house MBT professionals.**

cannot exclude a slight bias of positive attitude toward MBT.

To obtain more quantitative figures on effectiveness and efficiency, the survey asked three rather challenging questions:

- ▶ To what degree does MBT reduce or increase the number of escaped bugs—that is, the number of bugs nobody would have found before delivery?
- ▶ To what degree does MBT reduce or increase testing costs?
- ▶ To what degree does MBT reduce or increase testing duration?

Obviously, those questions were difficult to answer, and it was impossible to deduce statistically relevant information from the 21 answers obtained in the survey. Only one respondent clearly answered in the negative regarding the number of escaped bugs. All others provided positive figures, and two answers were illogical.

On the cost side, the survey asked respondents how many hours it took to become a proficient MBT user. The answers varied from zero to 2,000 hours of skill development, with a median of two work weeks (80 hours).

What Kind of Testing Is Model-Based?

Model-based testing is used at all stages of software development, most often for integration and system testing (Figure 7). Half of the survey respondents reported model-based integration testing, and about three-quarters reported conducting model-based system testing. Nearly all respondents used models for functional testing. Performance, usability, and security testing played a subordinate role with less than 20% each. Only one participant found it difficult to fit MBT into an agile approach, while 44% reported using MBT in an agile development process, with 25% at an advanced stage (rollout or generalized use).

There was a clear preference regarding model focus: 59% of the MBT models (any model used for testing purposes is referred to here as an MBT model) used by survey respondents focused on behavioral aspects only, while 35% had both structural and behavioral aspects. Purely statistical models played a minor role (6%). The trend was even more pronounced

Figure 5. Comparison between expectations and observed satisfaction level.

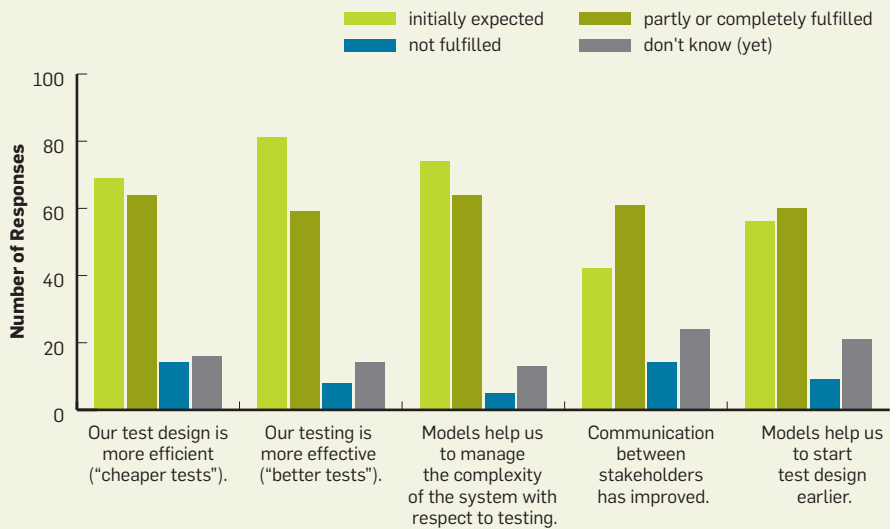


Figure 6. Nearly all participants rate MBT as being effective (to different degrees).

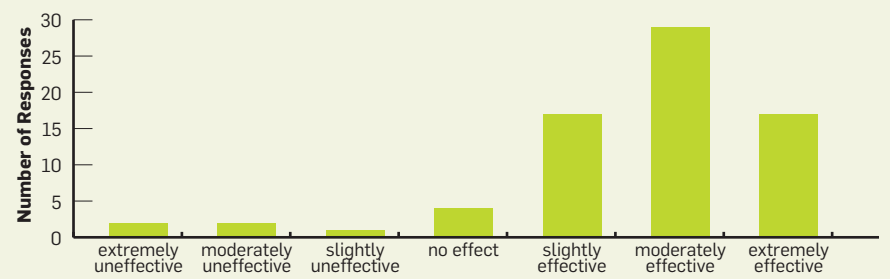
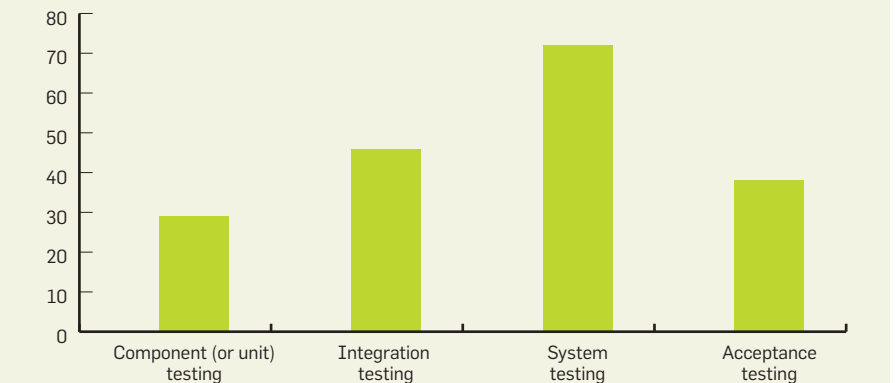


Figure 7. MBT plays an important role at all test levels.



for the notation type. Graphical notations prevailed with 81%; only 14% were purely textual MBT models—that is, they used no graphical elements. All kinds of combinations were used, however. One respondent

put it very clearly: “We use more than one model per system under test.”

Test modeling independent of design modeling. Note that more than 40% of the respondents did not use modeling in other development phases

Figure 8. Reusing models from other development phases has its limits.

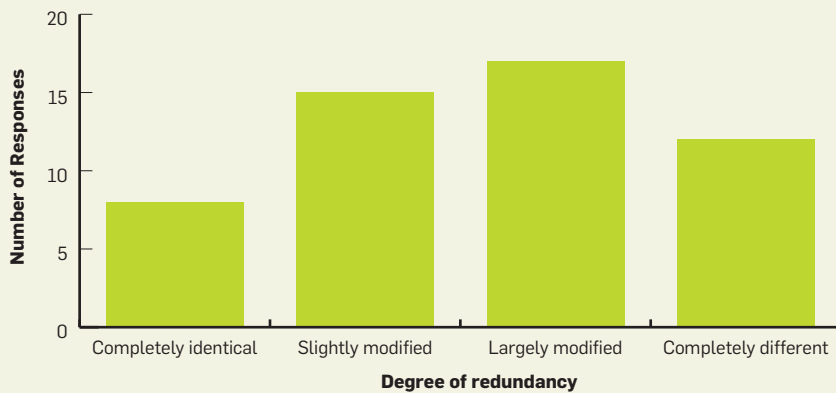
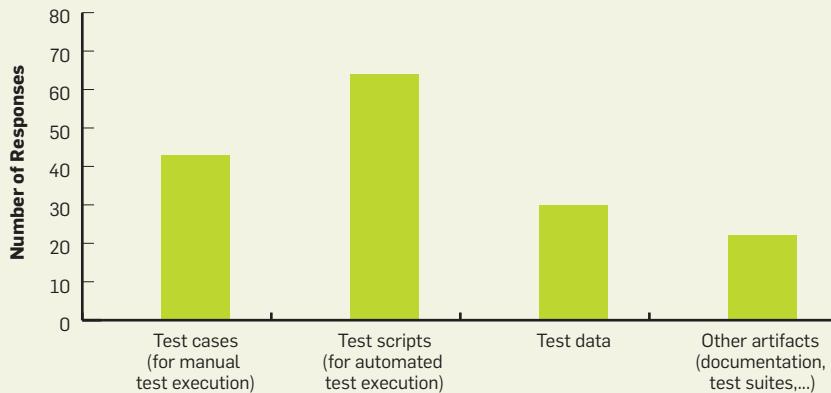


Figure 9. MBT is more than test-case generation for automated execution.



es. Only eight participants stated they reuse models from analysis or design without any modification. Figure 8 shows the varying degrees of redundancy. Twelve participants stated they wrote completely different models for testing purposes. The others more or less adapted the existing models to testing needs. This definitely showed that MBT may be used even when other development aspects are not model-based. This result was contrary to the oft-voiced opinion that model-based testing can be used only when modeling is also used for requirements and design.

Model-based testing compatible with manual testing. The 2014 MBT User Survey also showed that automated test execution is not the only way that model-based tests are applied. When asked about generated

artifacts, the large majority mentioned test scripts for automated test execution, but more than half of the respondents also generated test cases for manual execution from MBT models (see Figure 9). One-third of the respondents executed their test cases manually. Further, artifact generation did not even have to be tool-based: 12% obtained the test cases manually from the model; 36% at least partly used a tool; and 53% have established a fully automated test-case generation process.

Conclusion

Some 100 participants shared their experience in the 2014 MBT User Survey and provided valuable input for this analysis. Although the survey was not broadly representative, it provided a profile of active MBT usage over

a wide range of environments and organizations. For these users, MBT has had positive effects on efficiency and effectiveness, even if it only partially fulfills some extremely high expectations. The large majority said they intend to continue using models for testing purposes.

Regarding the common classification scheme, the responses confirmed the diversity of MBT approaches. No two answers were alike. This could put an end to the discussion of whether an MBT model may be classified as a system model, test model, or environment model. It cannot. Any model used for testing purposes is an MBT model. Usually, it focuses on all three aspects in varying degrees. Classifying this aspect appears to be an idle task. Some of the technical questions did not render useful information. Apparently, the notion of “degree of abstraction” of an MBT model is too abstract in itself. It seems to be difficult to classify an MBT model as either “very abstract” or “very detailed.”

The work is not over. We are still searching for correlations and trends. If you have specific questions or ideas regarding MBT in general and the survey in particular, please contact us. 

Related articles on queue.acm.org

Managing Contention for Shared Resources on Multicore Processors

Alexandra Fedorova, Sergey Blagodurov, and Sergey Zhuravlev
<http://queue.acm.org/detail.cfm?id=1709862>

Microsoft's Protocol Documentation Program: Interoperability Testing at Scale

A Discussion with Nico Kicillof, Wolfgang Grieskamp and Bob Binder
<http://queue.acm.org/detail.cfm?id=1996412>

FPGA Programming for the Masses

David F. Bacon, Rodric Rabbah, Sunil Shukla
<http://queue.acm.org/detail.cfm?id=2443836>

Robert V. Binder (rvbinder@sysverif.com) is a high-assurance entrepreneur and president of System Verification Associates, Chicago, IL. As test process architect for Microsoft's Open Protocol Initiative, he led the application of model-based testing to all of Microsoft's server-side APIs.

Bruno Legeard (bruno.legeard@femto-st.fr) is a professor at the University of Franche-Comté and cofounder and senior scientist at Smartesting, Paris, France.

Anne Kramer (anne.kramer@seppmed.de) is a project manager and senior process consultant at seppmed gmbh, a service provider specializing in IT solutions.

Inviting Young Scientists



Association for
Computing Machinery

Meet Great Minds in Computer Science and Mathematics

As one of the founding organizations of the Heidelberg Laureate Forum <http://www.heidelberg-laureate-forum.org/>, ACM invites young computer science and mathematics researchers to meet some of the preeminent scientists in their field. These may be the very pioneering researchers who sparked your passion for research in computer science and/or mathematics.

These laureates include recipients of the ACM A.M. Turing Award, the Abel Prize, and the Fields Medal.

The Heidelberg Laureate Forum is **August 23–28, 2015** in Heidelberg, Germany.

This week-long event features presentations, workshops, panel discussions, and social events focusing on scientific inspiration and exchange among laureates and young scientists.

Who can participate?

New and recent Ph.Ds, doctoral candidates, other graduate students pursuing research, and undergraduate students with solid research experience and a commitment to computing research

How to apply:

Online: <https://application.heidelberg-laureate-forum.org/>
Materials to complete applications are listed on the site.

What is the schedule?

Application deadline—**February 28, 2015**.

We reserve the right to close the application website early depending on the volume

Successful applicants will be notified by **April 15, 2015**.

More information available on Heidelberg social media



DOI:10.1145/2656385

Business leaders may bemoan the burdens of governing IT, but the alternative could be much worse.

BY CARLOS JUIZ AND MARK TOOMEY

To Govern IT, or Not to Govern IT?

TO GOVERN, OR not to govern information technology (IT) is no longer a choice for any organization. IT is a major instrument of business change in both private- and public-sector organizations. Without good governance, organizations face loss of opportunity and potential failure. Effective governance of IT promotes achievement of business objectives, while poor governance of IT obstructs and limits such achievement.

The need to govern IT follows from two strategic factors: business necessity and enterprise maturity. Business necessity follows from many actors in the market using technology to gain advantage. Consequently, being relevant and competitive requires organizations to deeply integrate their own IT agendas and strategic business plans to ensure appropriate positioning of technology opportunity and response to technology-enabled changes in the marketplace. Enterprise maturity follows from a narrow focus on operating infrastructure, architecture, and service management of an owned IT asset no longer being

key to development of the organization. Achieving value involves more diverse arrangements for sourcing, ownership, and control in which the use of IT assets is not linked to direct administration of IT assets. Divestment activities (such as outsourcing and adoption of cloud solutions) increasingly create unintended barriers to flexibility, as mature organizations respond to new technology-enabled pressure. Paradoxically, contemporary sourcing options (such as cloud computing and software-as-a-service) can increase flexibility and responsiveness. Business necessity and enterprise maturity thus overlap and feed each other.

The International Standard for Corporate Governance of Information Technology ISO/IEC 38500³ was developed in 2008 by experts from government and industry (<http://www.iso.org>) who understand the importance of re-setting the focus for governance of IT on business issues without losing sight of technology issues. While it does not say so explicitly, the standard leads to one inescapable three-part conclusion for which business leaders must assume responsibility:

Agenda. Setting the agenda for IT use as an integral aspect of business strategy;

Investment. Delivery of investments in IT-enabled business capability; and

Operations. Ongoing successful operational use of IT in routine business activity.

Implementation of effective ar-

» key insights

- **Governance of IT is a board and top-executive responsibility focusing on business performance and capability, not on technology details.**
- **A principles-based approach to the governance of IT, as described in the ISO/IEC 38500 standard, is consistent with broader models for guidance of the governance of organizations and accessible to business leaders without specific technology skills.**
- **Adopting ISO/IEC 38500 to guide governance of IT helps leaders plan, build, and run IT-enabled organizations.**



- I. Responsibility**
- II. Strategy**
- III. Acquisition**
- IV. Performance**
- V. Human Behavior**
- VI. Conformance**

rangements for governance of IT must also address the need for organizations to ensure value creation from investment in IT. Lack of good IT governance risks inappropriate investment, failure of services, and noncompliance with regulations.

Following de Haes and Van Grembergen,² proper governance of IT is needed to ensure investments in IT generate required business value and that risks associated with IT are mitigated. This latest consideration to value and risk is closer to the principles of good governance, but there remains in management-based published guidance on IT governance a predominantly procedural approach to the requirement for effective governance of IT.

IT Governance and Governance of IT

The notion of IT governance has existed since at least the late 1990s, producing diverse conflicting definitions. These definitions and the models that underpin them tend to focus on the supply of IT, from alignment of an organization's IT strategy to its business strategy to selection and delivery of IT projects to the operational aspects of IT systems. These definitions and models should have improved the capability of organizations to ensure their IT activities are on track to achieve their business strategies and goals. They should also have provided ways to measure IT performance, so IT governance should be able to answer questions regarding how the IT department is functioning

and generating return on investment for the business.

Understanding that older definitions and models of IT governance focus on the internal activities of the IT department leads to the realization that much of what has been called "IT governance" is in fact "IT management," and confusion has emerged among senior executives and IT managers regarding what exactly is governance and management (and even operation) of IT. The reason for this confusion is that the frontiers between them may be somewhat blurred and by a propensity of the IT industry to inappropriately refer to management activities as IT governance.¹²

There is widespread recognition

that IT is not a standalone business resource. IT delivers value only when used effectively to enable business capability and open opportunities for new business models. What were previously viewed as IT activities should instead be viewed as business activities that embrace the use of IT. Governance of IT must thus include important internal IT management functions covered by earlier IT governance models, plus external functions that address broader issues of setting and realizing the agenda for the business use of IT. Governance of IT must embrace all activities, from defining intended use of IT through delivery and subsequent operation of IT-enabled business capability.

We subscribe to the definition that governance of IT is the system to direct and control use of IT. As reinforced repeatedly through major government- and private-sector IT failures, control of IT must be performed from a business perspective, not an IT perspective. This perspective, and the definition of governance of IT, requires business leaders come to terms with what they can achieve by harnessing IT to enable and enhance business capability and focus on delivering the most valuable outcomes. Governance of IT must provide clear and consistent visibility of how IT is used, supplied, and acquired for everyone in the organization, from board members to business users to IT staff members.⁵

“Governance of IT” is equivalent to “corporate governance of IT,” “enterprise governance of IT,” and “organizational governance of IT.” Governance of IT has its origins in corporate governance. Corporate governance objectives include stewardship and management of assets and enterprise resources by the governing bodies of organizations, setting and achieving the organization’s purpose and objectives, and conformance⁹ by the organization with established and expected norms of behavior. Corporate governance is an important means of dealing with agency problems (such as when ownership and management interests do not match). Conflicts of interest between owners (shareholders), managers, and other stakeholders—citizens, clients, or users—can occur whenever these roles are separated.⁸

Lack of good IT governance risks inappropriate investment, failure of services, and noncompliance with regulations.

Corporate governance includes development of mechanisms to control actions taken by the organization and safeguard stakeholder interests as appropriate.⁴ Private and public organizations are subject to many regulations governing data retention, confidential information, financial accountability, and recovery from disasters.⁷ While no regulations require a governance-of-IT framework, many executives have found it an effective way to ensure regulatory compliance.⁶ By implementing effective governance of IT, organizations establish the internal controls they need to meet the core guidelines of many regulations.

Some IT specialists mistakenly think business leaders cannot govern IT, since they lack technology skills. Understanding the capability IT brings or planning new, improved business capability enabled by smarter, more effective use of IT does not require specialized knowledge of how to design, build, or operate IT systems. A useful metaphor in this sense is the automobile; a driver need not be a designer or a manufacturing engineer to operate a taxi service but must understand the capabilities and requirements for the vehicles used to operate the service.

Governance of IT Standardization

Australian Standard AS 8015, published in 2005, was the first formal standard to describe governance of IT without resorting to descriptions of management systems and processes. In common with many broader guides for corporate governance and governance in the public sector, AS 8015 took a principles-based approach, focusing its guidance on business use of IT and business outcomes, rather than on the technical supply of IT. ISO/IEC 38500, published in 2008, was derived from AS 8015 and is the first international standard to provide guidelines for governance of IT. The wording for the definition for governance of IT in AS 8015 and its successor, ISO/IEC 38500, was deliberately aligned with the definition of “corporate governance” in the Cadbury report.¹

Since well before release of either AS 8015 or ISO/IEC 38500, many organizations have confused governance and management of IT. This confusion is exacerbated by efforts to integrate

some aspects of governance in common de facto standards for IT management, resulting in these aspects of governance being described in management systems terms. In an effort to eliminate confusion, we no longer refer to the concept of IT governance, focusing instead on the overarching concepts for governance of IT and the detailed activities in IT management (see Figure 1).

Figure 2 outlines the final draft (issued November 2014) conceptual model for governance of IT from the proposed update of ISO/IEC 38500 and its relation with IT management. As the original ISO/IEC project editor for ISO/IEC 38500, author Mark Toomey¹² has presented evolved versions of the original ISO/IEC 38500 model that convey more clearly the distinction between governance and management activities and the business orientation essential for effective use of IT from the governance viewpoint. Figure 2 integrates Toomey's and the ISO/IEC 38500's current draft model to maximize understanding of the interdependence of governance and management in the IT context.

In the ISO/IEC 38500 model, the governing body is a generic entity (the individual or group of individuals) responsible and accountable for performance and conformance (through control) of the organization. While ISO/IEC 38500 makes clear the role of the governing body, it also allows that such delegation could result in a subsidiary entity giving more focused attention to the tasks in governance of IT (such as creation of a board committee). It also includes delegation of detail to management, as in finance and human resources. There is an implicit expectation that the governing body will require management establish systems to plan, build, and run the IT-enabled organization.

An informal interpretation of Figure 2, focused on business strategy and projects, is that there is a continuous cycle of activity that can simultaneously operate at several levels:

Evaluation. The governing body evaluates the organization's overall use of IT in the context of the business environment, directs management to perform a range of tasks relating to use of IT, and continues to monitor the use

Figure 1. Main ISO/IEC standards of IT management and governance of IT.

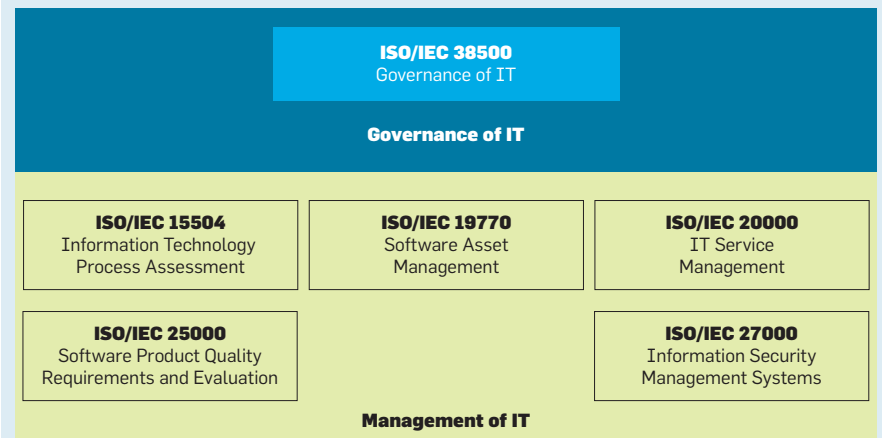
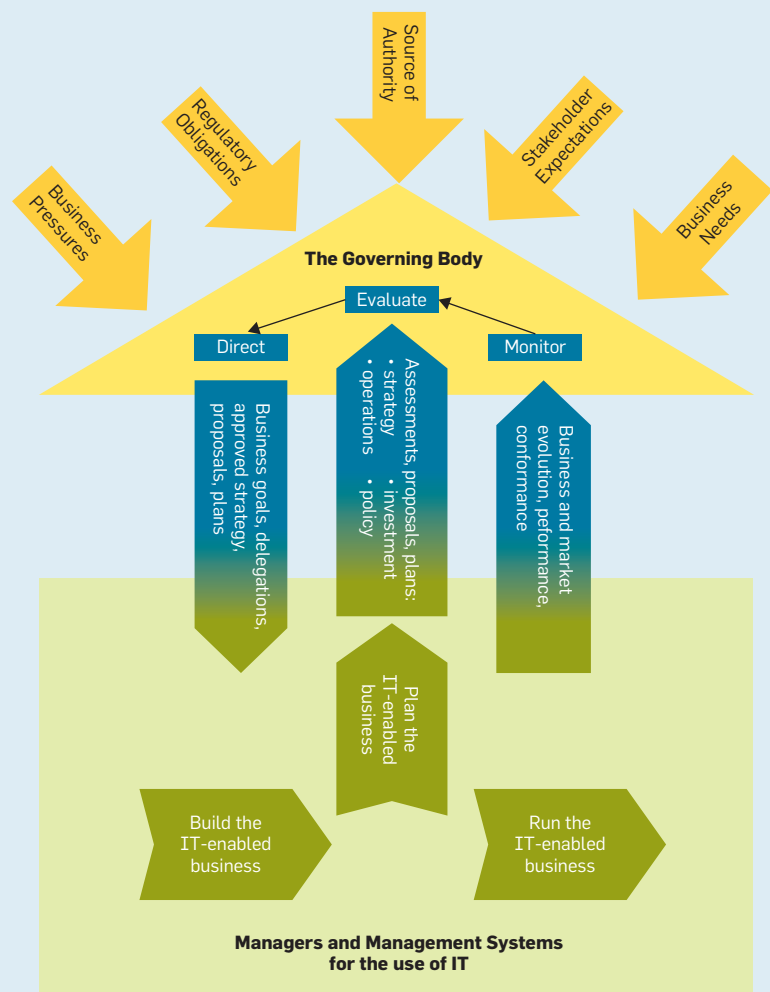


Figure 2. Model for governance of IT derived from the current Final Draft International Standard ISO/IEC 38500.³



of IT with regard to business and marketplace evolution;

Assessment. Business and IT units collaboratively develop assessment proposals and plans for business strategy, investment, operations, and policy for the IT-enabled business; and

Implementation. The governing body evaluates the proposed assessment proposals and plans and, where appropriate, directs that they should be adopted and implemented; the governing body then monitors implementation of the plans and policies as to whether they deliver required performance and conformance.

Regarding management scope, as outlined in Figure 2, managers must implement and run the following activities:

Plan. Business managers, supported by technology, organization development, and business-change professionals plan the IT-enabled business, as directed by the governing body, proposing strategy for the use of IT and investment in IT-enabled business capability;

Build. Investment in projects to build the IT-enabled business are undertaken as directed by and in con-

formance with delegation, plans, and policies approved by the board; project personnel with business-change and technology skills then work with line managers to build IT-enabled business capability;

Run. To close the virtuous cycle, once the projects become a reality, managers deliver the capability to run the IT-enabled business, supported by appropriate management systems for the operational use of IT; and

Monitor. All activities and systems involved in planning, building, and running the IT-enabled business are subject to ongoing monitoring of market conditions, performance against expectations, and conformance with internal rules and external norms.

ISO/IEC 38500 set out six principles for good corporate governance of IT that express preferred organizational behavior to guide decision making. By merging and clarifying the terms for the principles from AS 8015 and ISO/IEC 38500, we derive the following summary of the principles:

Responsibility. Establish appropriate responsibilities for decisions relating to the use and supply of IT;

Strategy. Plan, supply, and use IT to best support the organization;

Acquisition. Invest in new and ongoing use of IT;

Performance. Ensure IT performs well with respect to business needs as required;

Conformance. Ensure all aspects of decision making, use, and supply of IT conforms to formal rules; and

Human behavior. Ensure planning, supply, and use of IT demonstrate respect for human behavior.

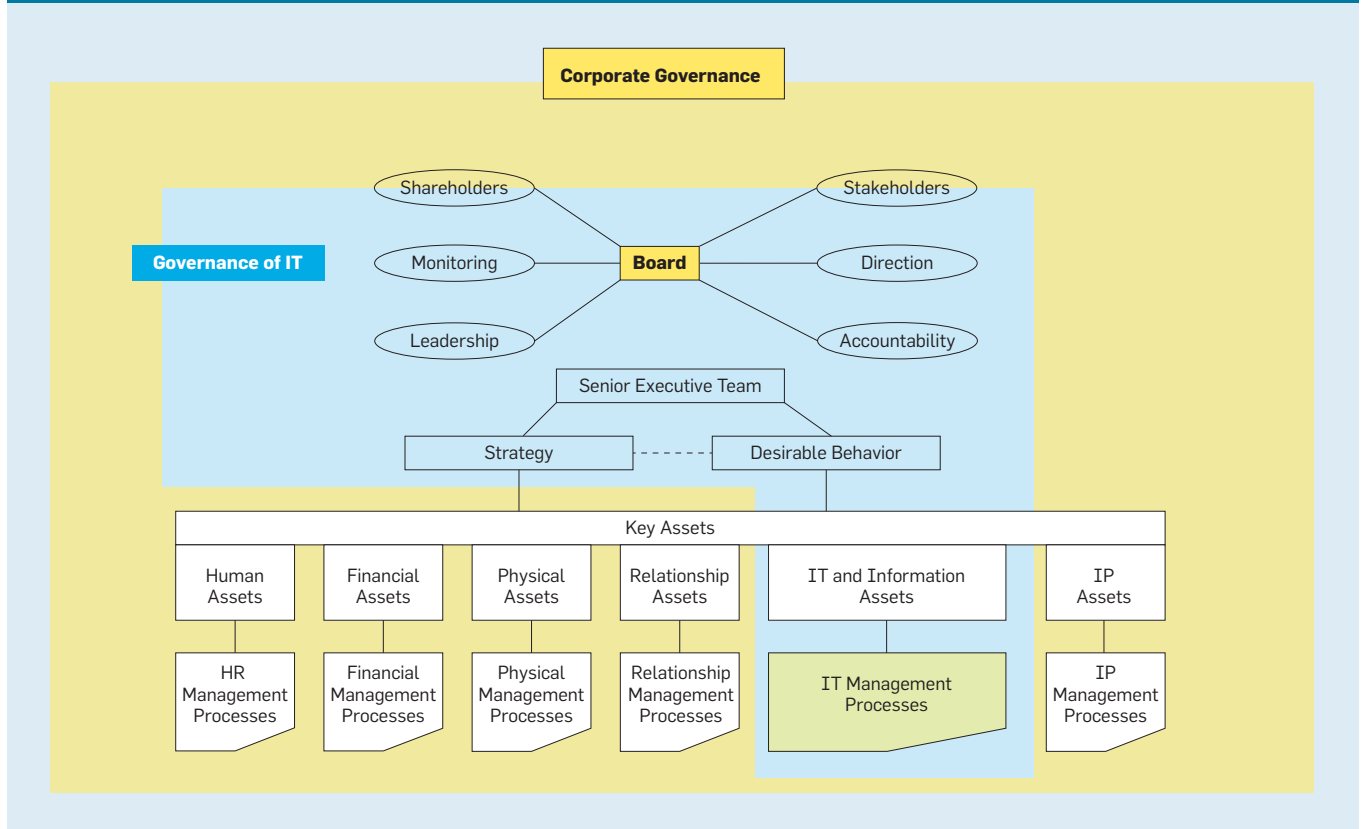
These principles and activities clarify the behavior expected from implementing governance of IT, as in Stachtchenko:¹⁰

Stakeholders. Stakeholders delegate accountability and stewardship to the governance body, expecting in exchange that body to be accountable for activities necessary to meet expectations;

Governance body. The governance body sets direction for management of the organization, holding management accountable for overall performance; and

Stewardship role. The governance body takes a stewardship role in the

Figure 3. Coverage area for behavior-oriented governance and management of IT, linking corporate and key assets (own elaboration from Weill and Ross¹⁴).



traditional sense of assuming responsibility for management of something entrusted to one's care.

Governance of IT: Process-Oriented vs. Behavior-Oriented

Van Grembergen¹³ defined governance of IT as the organizational capacity exercised by the board, executive management, and IT management to control formulation and implementation of IT strategy, ensuring fusion of business and IT. Governance consists of leadership, organizational structures, and processes that ensure the organization's IT sustains and extends the organization's strategy and objectives. This definition is loosely consistent with the IT Governance Institute's definition⁴ that governance of IT is part of enterprise governance consisting of leadership, organizational structures, communication mechanisms, and processes that ensure the organization's IT sustains and extends the organization's strategy and objectives. However, both definitions are more oriented to processes, structures, and strategy than the behavioral side of good governance, and, while embracing the notion that effective governance depends on effective management systems, tend to focus on system aspects rather than on true governance of IT aspects.

Weill and Ross¹⁴ said governance of IT involves specifying the decision rights and accountability framework to produce desired behavior in the use of IT in the organization. Van Grembergen¹³ said governance of IT is the responsibility of executives and senior management, including leadership, organizational structures, and processes that ensure IT assets support and extend the organization's objectives and strategies. Focusing on how decisions are made underscores the first ISO/IEC 38500 principle, emphasizing behavior in assigning and discharging responsibility is critical for deriving value from investment in IT and to the organization's overall performance.

Governance of IT must thus include a framework for organizationwide decision rights and accountability to encourage desirable behavior in the use of IT. Within the broader system for governance of IT, IT management focuses



**The best
process model
is often readily
defeated by poor
human behavior.**



on a small but critical set of IT-related decisions, including IT principles, enterprise architecture, IT infrastructure capabilities, business application needs, and IT investment and prioritization.¹⁴ Even though governing IT and its core is deeply behavioral, this set of IT-related decisions defines the implementation framework. These decision rights define mainly who makes decisions delegated by the governing body and what decisions they make, along with how they do it. Focusing on decision rights intrinsically defines behavioral rather than process aspects of the governance of IT.

Likewise, process-oriented IT management as described in Control Objectives for Information and Related Technology, or COBIT (<http://www.isaca.org/cobit>), and similar frameworks is also part of the governance of IT, ensuring IT activities support and enable enterprise strategy and achievement of enterprise objectives. However, focusing primarily on IT management processes does not ensure good governance of IT. IT management processes define mainly what assets are controlled and how they are controlled. They do not generally extend to broader issues of setting business strategy influenced by or setting the agenda for the use of IT. Nor do they extend fully into business capability development and operational management intrinsic to the use of IT in most organizations. The latest version of COBIT—COBIT 5—includes the ISO/IEC 38500 model for the first time. However, there is a quite fundamental and significant difference between ISO/IEC 38500 and COBIT 5 and is a key focus of our research. Whereas ISO/IEC 38500 takes a behavioral stance, offering guidance about governance behavior, COBIT 5 takes a process stance, offering guidance about process, mainly suggesting auditable performance metrics rather than process descriptions.

Process-oriented IT management frameworks, including processes for extended aspects of management dealing with the business use of IT, are frequently important, especially in larger organizations, but are insufficient to guarantee good governance and management because they are at risk of poor behavior by individu-

als and groups within and sometimes even external to the organization. The best process model is often readily defeated by poor human behavior. We see evidence of poor behavior in many investigations of failed IT projects (such as the Queensland Audit Office 2009 review of Queensland Health Payroll¹¹). On the other hand, good behavior ensures conformance with an effective process model and compensates for deficiencies in weaker process models.

In any effective approach to the governance of IT, the main activities described in ISO/IEC 38500—direct, evaluate, monitor—must be performed following the six principles of this standard and must guide behavior with respect to IT management.

Good corporate governance is not the only reason for organizations to improve governance of IT. From the outset, most discussions identify “stakeholder value drivers” as the main reason for organizations to upgrade governance of IT. Stakeholder pressure drives the need for effective governance of IT in commercial organizations. Lack of such pressure may explain why some public services have less effective governance of IT.¹² The framework depicted in Weill and Ross¹⁴ has been expanded for governance of IT (see Figure 3), showing the connection between corporate governance and key-assets governance.

Figure 3 emphasizes the system for governance of IT extends beyond the narrow domain of IT-management processes. The board’s relationships are outlined at the top of the framework. The senior executive team is commissioned by the board to help it formulate strategies and desirable behaviors for the organization, then implement the strategies and behaviors. Six key asset classes are identified below the strategy and desirable behaviors. In this framework, governance of IT includes specifying the decision rights and accountability framework responsibilities (as described in ISO/IEC 38500) to encourage desirable behavior in the use of IT. These responsibilities apply broadly throughout the organization, not only to the CIO and the IT department. Governance of IT is not conceptually different from governing other assets (such as financial, personnel, and intellectual property).

Strategy, policies, and accountability thus represent the pillars of the organization’s approach to governance of IT.

This behavioral approach is less influenced by and less dependent on processes. It is conducted through decisions of governance structures and proper communication and is much more focused on human communities and behaviors than has been proposed by any process-oriented IT management model.

Conclusion

Focusing on technology rather than on its use has yielded a culture in which business leaders resist involvement in leadership of the IT agenda. This culture is starkly evident in many analyses of IT failure. Business leaders have frequently excused themselves from a core responsibility to drive the agenda for business performance and capability through the use of all available resources, including IT.

Governance of IT involves evaluating and directing the use of IT to support the organization and monitoring this use to achieve business value. As defined in ISO/IEC 38500, governance of IT drives the IT management framework, requiring top-down focus on producing value through effective use of IT and an approach to governance of IT that engages business leaders in appropriate behavior. Governance of IT includes business strategy as the principle agenda for the use of IT, plus the policies that drive appropriate behavior, clear accountability and responsibility for all stakeholders, and recognition of the interests and behaviors of stakeholders beyond the control of the organization.

Using ISO/IEC 38500 to guide governance of IT, regardless of which models are used for management systems, ensures governance of IT has appropriate engagement of the governing body, with clear delegation of responsibility and associated accountability. It also provides essential decoupling of governance oversight from management detail while preserving the ability of the governing body to give direction and monitor performance.

Asking whether to govern IT, or not to govern IT should no longer be a question. Governing IT from the top, focusing on business capability, perfor-

mance, and value should be normal behavior in any organization, generating business value from investment in and the ongoing operation of IT-enabled business capability, with appropriate accountability for all stakeholders.

Acknowledgment

This work was partially supported by the Spanish Ministry of Economy and Competitiveness under grant TIN2011-23889. 

References

1. Cadbury, A. (chair). *Report of the Committee on the Financial Aspects of Corporate Governance*. Burgess Science Press, London, U.K., 1992.
2. de Haes, S. and Van Grembergen, W. IT governance and its mechanisms. *Information Systems Control Journal* 1 (2004), 1–7.
3. ISO/IEC. *ISO/IEC 38500: 2008 Corporate Governance of Information Technology*. ISO/IEC, Geneva, Switzerland, June 2008; http://www.iso.org/iso/catalogue_detail?csnumber=51639
4. IT Governance Institute. *Board Briefing on IT Governance, Second Edition*. IT Governance Institute, Rolling Meadows, IL, 2003; http://www.isaca.org/restricted/Documents/26904_Board_Briefing_final.pdf
5. Juiz, C. New engagement model of IT governance and IT management for the communication of the IT value at enterprises. Chapter in *Digital Enterprise and Information Systems*, E. Ariwa and E. El-Qawasmeh, Eds. *Communications in Computer and Information Science Series*, Vol. 194. Springer, 2011, 129–143.
6. Juiz, C., Guerrero, C., and Lera, I. Implementing good governance principles for the public sector in information technology governance frameworks. *Open Journal of Accounting* 3, 1 (Jan. 2014), 9–27.
7. Juiz, C. and de Pous, V. Cloud computing: IT governance, legal, and public-policy aspects. Chapter in *Organizational, Legal, and Technological Dimensions of Information System Administration*, I. Portela and F. Almeida, Eds. IGI Global, Hershey, PA, 2013, 139–166.
8. Langland, A. (chair). *Good Governance Standard for Public Services*. Office for Public Management Ltd. and Chartered Institute of Public Finance and Accountancy, London, U.K., 2004; http://www.cipfa.org/-/media/Files/Publications/Reports/governance_standard.pdf
9. Professional Accountants in Business Committee of the International Federation of Accountants. *International Framework, Good Governance in the Public Sector: Comparison of Principles*. IFAC, New York, 2014; <http://www.ifac.org/sites/default/files/publications/files/Comparison-of-Principles.pdf>
10. Stachtchenko, P. Taking governance forward. *Information Systems Control Journal* 6 (2008), 1–2.
11. Toomey, M. Another governance catastrophe. *The Infonomics Letter* (June 2010), 1–5.
12. Toomey, M. *Waltzing With the Elephant: A Comprehensive Guide to Directing and Controlling Information Technology*. Infonomics Pty Ltd., Victoria, Australia, 2009.
13. Van Grembergen, W. *Strategies for Information Technology Governance*. Idea Group Publishing, Hershey, PA, 2004.
14. Weill, P. and Ross, J.W. *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*. Harvard Business School Press, Cambridge, MA, 2004.

Carlos Juiz (cjuiz@uib.es) is an associate professor at the University of the Balearic Islands, Palma de Mallorca, Spain, and leads the Governance of IT Working Group at AENOR, the Spanish body in ISO/IEC.

Mark Toomey (mtoomey@infonomics.com.au) is managing director at Infonomics Pty Ltd., Melbourne, Australia, and was the original ISO project editor of ISO/IEC 38500.

Model checking and logic-based learning together deliver automated support, especially in adaptive and autonomous systems.

BY DALAL ALRAJEH, JEFF KRAMER,
ALESSANDRA RUSSO, AND SEBASTIAN UCHITEL

Automated Support for Diagnosis and Repair

EDWARD FEIGENBAUM AND Raj Reddy won the ACM A.M. Turing Award in 1994 for their pioneering work demonstrating the practical importance and potential impact of artificial intelligence technology. Feigenbaum was influential in suggesting the use of rules and induction as a means for computers to learn

theories from examples. In 2007, Edmund M. Clarke, E. Allen Emerson, and Joseph Sifakis won the Turing Award for developing model checking into a highly effective verification technology for discovering faults. Used in concert, verification and AI techniques

can provide a powerful discovery and learning combination. In particular, the combination of model checking¹⁰ and logic-based learning¹⁵ has enormous synergistic potential for supporting the verify-diagnose-repair cycle software engineers commonly use in complex systems development. In this article, we show how to realize this synergistic potential.

Model checking exhaustively searches for property violations in formal descriptions (such as code, requirements, and design specifications, as well as network and infrastructure configurations), producing counterexamples when these properties do not hold. However, though model checkers are effective at uncovering faults in formal descriptions, they provide only

» key insights

- **The marriage of model checking for finding faults and machine learning for suggesting repairs promises to be a worthwhile, synergistic relationship.**
- **Though separate software tools for model checking and machine learning are available, their integration has the potential for automated support of the common verify-diagnose-repair cycle.**
- **Machine learning ensures the suggested repairs fix the fault without introducing any new faults.**

limited support for understanding the causes of uncovered problems, let alone how to fix them. When uncovering a violation, model checkers usually provide one or more examples of how such a fault occurs in the description or model being analyzed. From this feedback, producing an explanation for the failure and generating a fix are complex tasks that tend to be human-intensive and error-prone. On the other hand, logic-based learning algorithms use correct examples and violation counterexamples to extend and modify a formal description such that the description conforms to the examples while avoiding the counterexamples. Although counterexamples are usually provided manually, examples and counterexamples can be provided through verification technology (such as model checking).

Consider the problem of ensuring a contract specification of an API satisfies some invariant. Automated verification can be performed through a model checker that, should the invariant be violated, will return an example sequence of operations that breaks the invariant. Such a trace constitutes a counterexample that can then be used by a learning tool to correct the contract specification so the violation can no longer occur. The correction typically results in a strengthened post-condition for some operation so as to ensure the sequence does not break the invariant or perhaps a strengthened operation pre-condition so as to ensure the offending sequence of operations is no longer possible. For example, in Alrajeh⁴ the contract specification of the engineered safety-feature-actuation subsystem for the safety-injection system of a nuclear power plant was built from scratch through the combined

use of model checking and learning.

Another software engineering application for the combined technologies is obstacle analysis and resolution in requirements goal models. In it, the problem for software engineers is to identify scenarios in which high-level system goals may not be satisfied due to unexpected obstacles preventing lower-level requirements from being satisfied; for instance, in the London Ambulance System²¹ an incident is expected to be resolved some time after an ambulance intervenes. For an incident to be so resolved, an injured patient must be admitted to the nearest hospital and the hospital must have all the resources to treat that patient. The goal is flawless performance, as it does not consider the case in which the nearest hospital lacks sufficient resources (such as a bed), a problem not identified in the original analysis. Model checking and learning helped identify and resolve this problem automatically. Model checking the original formal description of the domain against the stated goal automatically generates a scenario exemplifying this case; logic-based learning automatically revises the goal description according to this scenario by substituting the original with one saying patients should be admitted to a nearby hospital with available resources. A similar approach has also been used to identify and repair missing use cases in a television-set configuration protocol.³

The marriage of model checking and logic-based learning thus provides automated support for specification verification, diagnosis, and repair, reducing human effort and potentially producing a more robust product. The rest of this article explores a general framework for integrating model checking and logic-based learning (see Figure 1).

Basic Framework

The objective of the framework is to produce—from a given formal description and a property—a modified description guaranteed to satisfy the property. The software engineer's intuition behind combining model checking and learning is to view them as complementary approaches; model checking automatically detects errors in the formal description, and learn-

ing carries out the diagnosis and repair tasks for the identified errors, resulting in a correctly revised description.

To illustrate the framework—four steps executed iteratively—we consider the problem of developing a contract-based specification for a simplified train-controller system.²⁰ Suppose the specification includes the names of operations the train controller may perform and some of the pre- and post-conditions for each operation; for instance, the specification says there is an operation called “close doors” that causes a train's open doors to be closed. Other operations are for opening the train doors and starting and stopping the train. Two properties the system must guarantee are safe transportation ($P1$, or “train doors are closed when the train is moving”) and rapid transportation ($P2$, or “train shall accelerate when the block signal is set to go”) (see Figure 2).

Step 1. Model checking. The aim of this step is to check the formal description for violations of the property. The result is either a notification saying no errors exist, in which case the cycle terminates, or that an error exists, in which case a counterexample illustrating the error is produced and the next step initiated. In the train-controller example, the model checker checks whether the specification satisfies the properties $P1$ and $P2$. The checker finds a counterexample demonstrating a sequence of permissible operation executions leading to a state in which the train is moving and the doors are open, thereby violating the safe-transportation property $P1$. Since a violation exists, the verify-diagnose-repair process continues to the next step.

Step 2. Elicitation. The counterexample produced by the model-checking step is not an exhaustive expression of all ways property $P1$ may be violated; other situations could also lead to a violation of $P1$ and also of $P2$. This step gives software engineers an opportunity to provide additional and, perhaps, related counterexamples. Moreover, it may be that the description and properties are inconsistent; that is, all executions permitted by the description violate some property. Software engineers may therefore provide traces (called “witnesses”) that exemplify how the property should have been sat-

Figure 1. General verify-diagnose-repair framework.

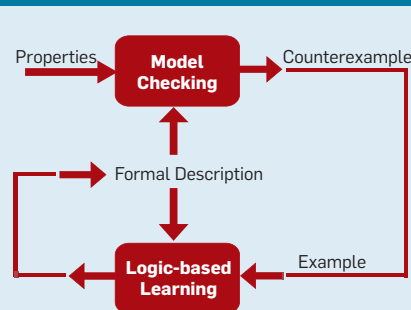
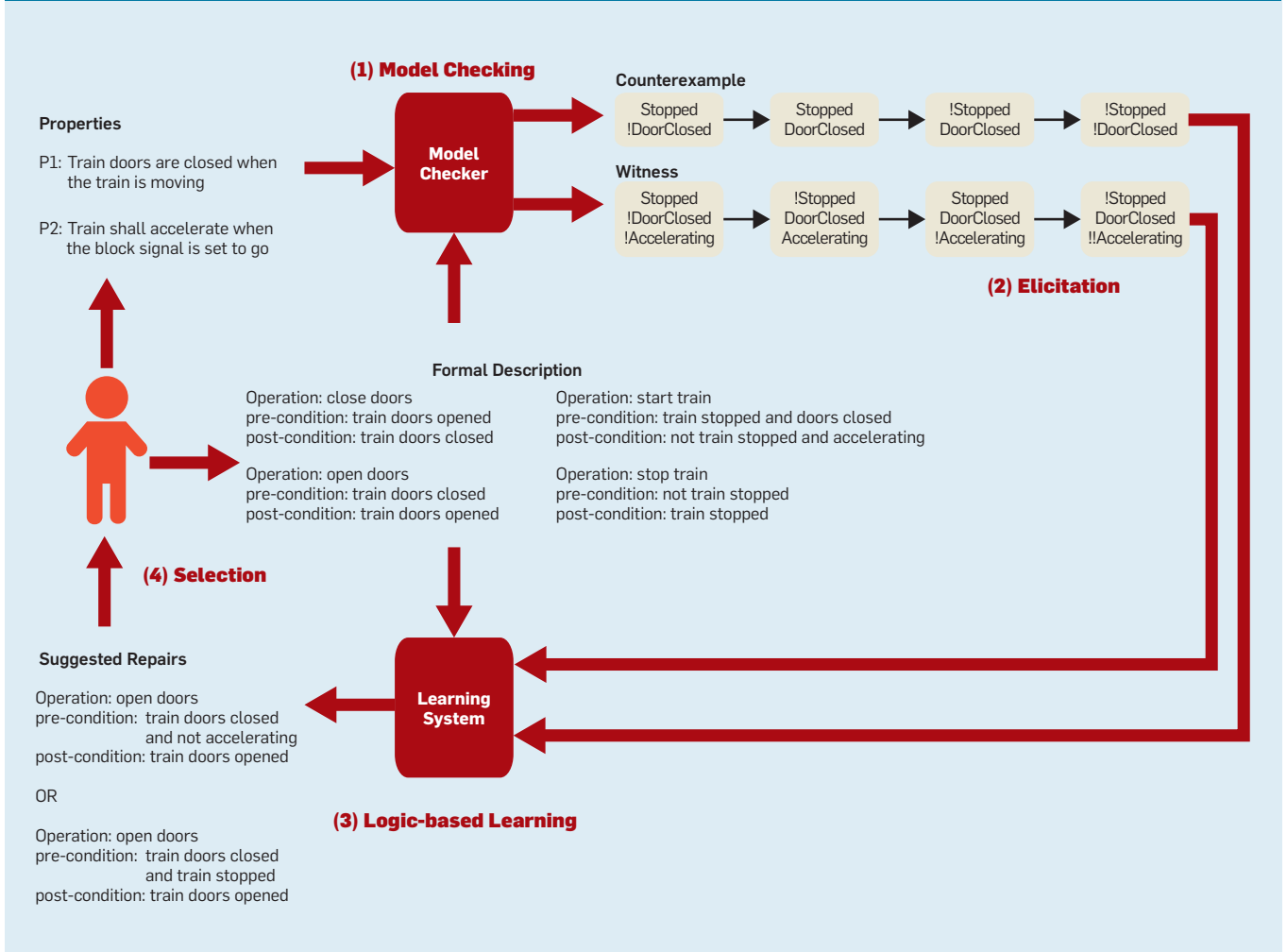


Figure 2. Train-controller example.



ified. Such examples may be manually elicited by the software engineer(s) or automatically generated through further automated analysis. In the simplified train-controller system example, a software engineer can confirm the specification and properties are consistent by automatically eliciting a witness trace that shows how *P1* can be satisfied keeping the doors closed while the train is moving and opening them when the train has stopped.

Step 3. Logic-based learning. Having identified counterexamples and witness traces, the logic-based learning software carries out the repair process automatically. The learning step's objective is to compute suitable amendments to the formal description such that the detected counterexample is removed while ensuring the witnesses are accepted under the amended description. For the train controller, the specification corresponds to the available background theory; the

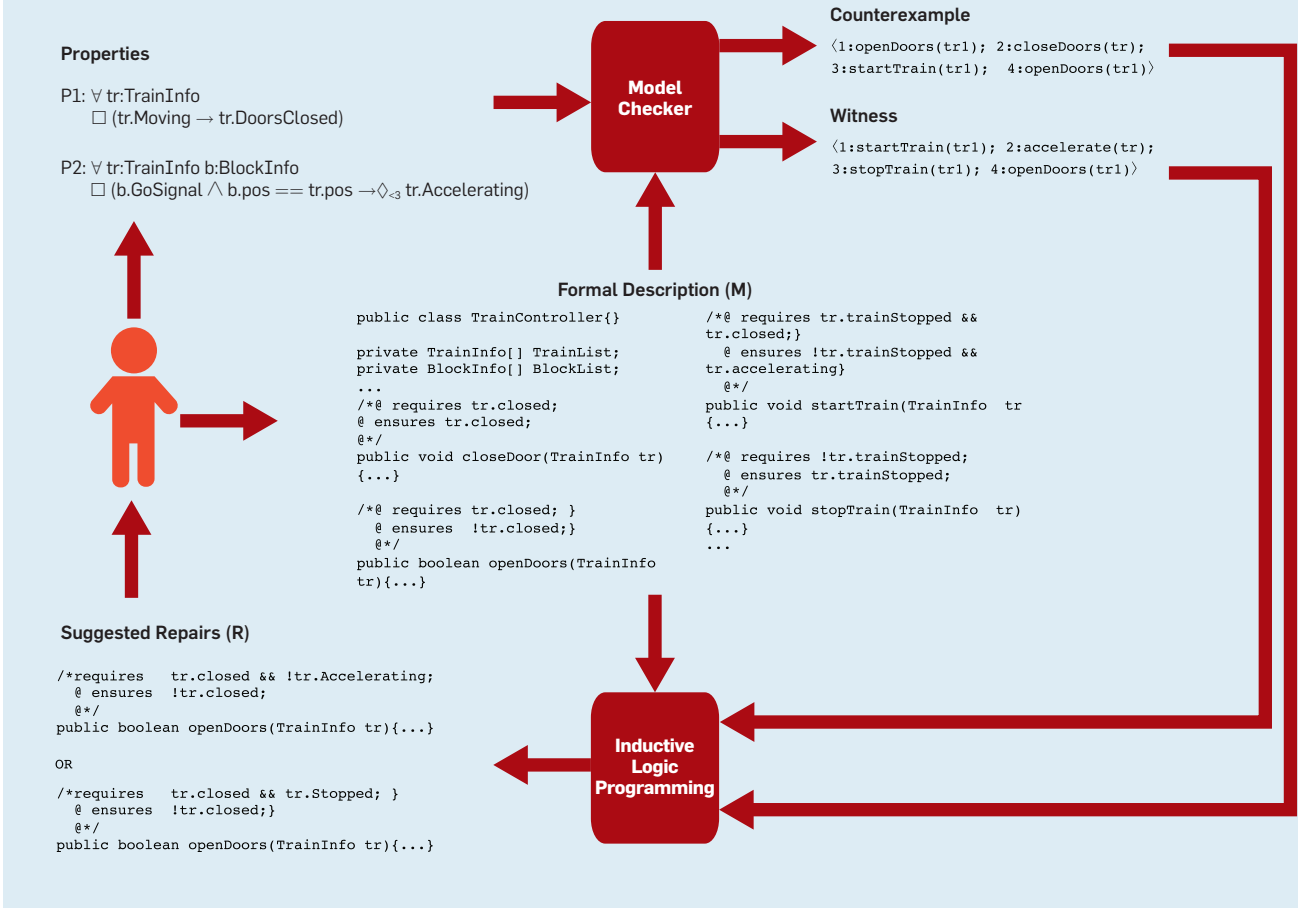
negative example is the doors opening when the train is moving, and the positive example is the doors opening when it has stopped. The purpose of the repair task is to strengthen the pre- and post-conditions of the train-controller operations to prevent the train doors from opening when undesirable to do so. The learning algorithm finds the current pre-condition of the open-door operation is not restrictive enough and consequently computes a strengthened pre-condition requiring the train to have stopped and the doors to be closed for such an operation to be executed.

Step 4. Selection. In the case where the logic-based learning algorithm finds alternative amendments for the same repair task, a selection mechanism is needed for deciding among them. Selection is domain-dependent and requires input from a human domain expert. When a selection is made, the formal description is up-

dated automatically. In the simplified train-controller-system example, an alternative strengthened pre-condition—the doors are closed, the train is not accelerating—is suggested by the learning software, in which case the domain experts could choose to replace the original definition of the open-doors operation.

The framework for combining model checking and logic-based learning is intended to iteratively repair the formal description toward one that satisfies its intended properties. The correctness of the formal description is most often not realized in a single application of the four steps outlined earlier, as other violations of the same property or other properties may still exist. To ensure all violations are removed, the steps must be repeated automatically until no counterexamples are found. When achieved, the correctness of the framework's formal description is guaranteed.

Figure 3. Concrete instantiation for train-controller example.



Concrete Instantiation

We now consider model checking more formally, focusing on Zohar Manna’s and Amir Pnueli’s Linear Temporal Logic (LTL) and Inductive Logic Programming (ILP), a specific logic-based learning approach. We offer a simplified example on contract-based specifications and discuss our experience supporting several software-engineering tasks. For a more detailed account of model checking and ILP and their integration, see Alrajeh et al.,⁵ Clarke,¹⁰ and Corapi et al.¹¹

Model checking. Model checkers require a formal description (M), also referred to as a “model,” as input. The input is specified using well-formed expressions of some formal language (L_M) and a semantic mapping ($s: L_M \rightarrow D$) from terms in L_M to a semantic domain (D) over which analysis is performed. They also require that the property (P) be expressed in a formal language (L_P) for which there is a satisfiability relation ($\models \subseteq D \times L_P$) capturing when an element of D satisfies the property. Given a for-

mal description M and a property P , the model checker decides if the semantics of M satisfies the property $s(M) \models P$.

Model checking goes beyond checking for syntactic errors a description M may have by performing an exhaustive exploration of its semantics. An analogy can be made with modern compilers that include sophisticated features beyond checking if the code adheres to the program language syntax and consider semantic issues (such as to de-reference a pointer with a null value). One powerful feature of model checking for system fault detection is its ability to automatically generate counterexamples that are meant to help engineers identify and repair the cause of a property violation (such as an incompleteness of the description with respect to the property being checked, or $s(M) \not\models P$ and $s(M) \not\models \neg P$), an incorrectness of the description with respect to the property, or $s(M) \models \neg P$, and the property itself being invalid. However, these tasks are complex, and only limited automated support exists

for resolving them consistently. Even in relatively small simplified descriptions, such resolution is not trivial since counterexamples are expressed in terms of the semantics rather than the language used to specify the description or the property, counterexamples show symptoms of the cause but not the cause of the violation itself, and any manual modification to the formal description could fail to resolve the problem and introduce violations to other desirable properties.

Consider the example outlined in Figure 3. The formal description M is a program describing a train-controller class using L_M , a JML-like specification language. Each method in the class is coupled with a definition of its preconditions (preceded with the keyword *requires*) and post-conditions (preceded by the keyword *ensures*). The semantics of the program is defined over a labeled transition system (LTS) in which nodes represent the different states of the program and edges represent the method calls that cause

the program to transit from one state to another. Property P is an assertion indicating what must hold at every state in every execution of the LTS $s(M)$. The language L_P used for expressing these properties is LTL. The first states it should always be the case (where $*$ means always) that if a train tr is moving, then its doors are closed. The second states the train tr shall accelerate within three seconds of the block signal b at which it is located being set to go. To verify $s(M) \models P$, an explicit model checker first synthesizes an LTS that represents all possible executions permitted by the given program M . It then checks whether P is satisfied in all executions of the LTS.

In the train-controller example, there is an execution of $s(M)$ that violates $P1 \wedge P2$; hence a counterexample is produced, as in Figure 3. Despite the simplicity and size of this counterexample, the exact cause of the violation is not obvious to the software engineer. Is it caused by an incorrect method invocation, a missing one, or both? If an incorrect method invocation, which method should not have been called? Should this invocation be corrected by strengthening its precondition or changing the post-condition of previously called operations? If caused by a missing invocation, which method should have been invoked? And under what conditions?

To prepare the learning step for a proper diagnosis of the encountered violations, witness traces to the properties are elicited. They may be provided either by the software engineer through specification, simulation, and animation techniques or through model checking. Figure 3 includes a witness trace elicited from $s(M)$ by model checking against $(\neg P1 \vee \neg P2)$. In this witness, the train door remains closed when the train is moving, satisfying $P1$ and satisfying $P2$ vacuously.

Inductive Logic Programming

Once a counterexample and witness traces have been produced by the model checker, the next step involves generating repairs to the formal description. If represented declaratively, automatic repairs can be computed by means of ILP. ILP is a learning technique that lies at the intersection of machine learning and logic program-



The marriage of model checking and logic-based learning thus provides automated support for specification verification, diagnosis, and repair, reducing human effort and potentially producing a more robust product.



ming. It uses logic programming as a computational mechanism for representing and learning plausible hypothesis from incomplete or incorrect background knowledge and set of examples. A logic program is defined as a set of rules of the form $h \leftarrow b_1, \dots, b_j, \text{not } b_{j+1}, \dots, \text{not } b_n$, which can be read as whenever b_1 and ... and b_j hold, and b_{j+1} and ... and b_n do not hold, then h holds. In a given clause, h is called the “head” of the rule, and the conjunction $\{b_1, \dots, b_j, \text{not } b_{j+1}, \dots, \text{not } b_n\}$ is the “body” of the rule. A rule with an empty body is called an “atom,” and a rule with an empty head is called an “integrity constraint.” Integrity constraints given in the initial description are assumed to be correct (therefore not revisable) and must be satisfied by any learned hypothesis.

In general, ILP requires as input background knowledge (B) and set of positive (E^+) and negative (E^-) examples that, due to incomplete information, may not be inferable but that are consistent with the current background knowledge. The task for the learning algorithm is to compute a hypothesis (H) that extends B so as to entail the set of positive examples ($B \wedge H \models E^+$) without covering the negative ones ($B \wedge H \not\models E^-$). Different notions of entailment exist, some weaker than others;¹⁶ for instance, cautious (respectively brave) entailment requires what appears on the right-hand side of the entailment operator to be true (in the case of $\models$) or false (in the case of $\not\models$) in every (respectively at least one) interpretation of the logic program on the right. ILP, like all forms of machine learning, is fundamentally a search process. Since the goal is to find a hypothesis for given examples, and many alternative hypotheses exist, these alternatives are considered automatically during the computation process by traversing a lattice-type hypothesis space based on a generality-ordering relation for which the more examples a hypothesis explains, the more general is the hypothesis.

“Non-monotonic” ILP is a particular type of ILP that is, by definition, capable of learning hypothesis H that alters the consequences of a program such that what was true in B alone is not necessarily true once B is extended with H . Non-monotonic ILP is

therefore well suited for computing revisions of formal descriptions, expressed as logic programs. The ability to compute revisions is particularly useful when an initial, roughly correct specification is available and a software engineer wants to improve it automatically or semi-automatically according to examples acquired incrementally over time; for instance, when evidence of errors in a current specification is detected, a revision is needed to modify the specification such that the detected error is no longer possible. However, updating the description with factual information related to the evidence would simply amount to recording facts. So repairs must generalize from the detected evidence and construct minimal but general revisions of the given initial specification that would ensure its semantics no longer entails the detected errors. The power of non-monotonic ILP to make changes to the semantics of a given description makes it ideal for the computation of repairs. Several non-monotonic ILP tools (such as XHAIL and ASPAL) are presented in the machine learning literature where the soundness and completeness of their respective algorithms have been shown. These tools typically aim to find minimal solutions according to a predefined score function that considers the size of the constructed hypotheses, number of positive examples covered, and number of negative examples not covered as parameters.

Integration. Integration of model checking with ILP involves three main steps: translation of the formal description; counterexamples and witness traces generated by the model checker into logic programs appropriate for ILP learning; computation of hypotheses; and translation of the hypotheses into the language L_M of the specification (see Figure 4).

Consider the background theory in Figure 4 for our train example. This is a logic program representation of the description M together with its semantic properties. Expressions “train(tr1)” and “method(openDoors(Tr)) ϕ train(Tr)” say tr1 is a train and openDoors(Tr) is a method, whereas the expression “ ϕ execute(M , T), requires(M , C), not holds(C , T)”



Model checking automatically detects errors in the formal description, and learning carries out the diagnosis and repair tasks for the identified errors, resulting in a correctly revised description.



is an integrity constraint that captures the semantic relationship between method execution and their pre-conditions; the system does not allow for a method M to be executed at a time T when its pre-condition C does not hold at that time. Expressions like execute(openDoors(Tr), T) denote the narrative of method execution in a given execution run.

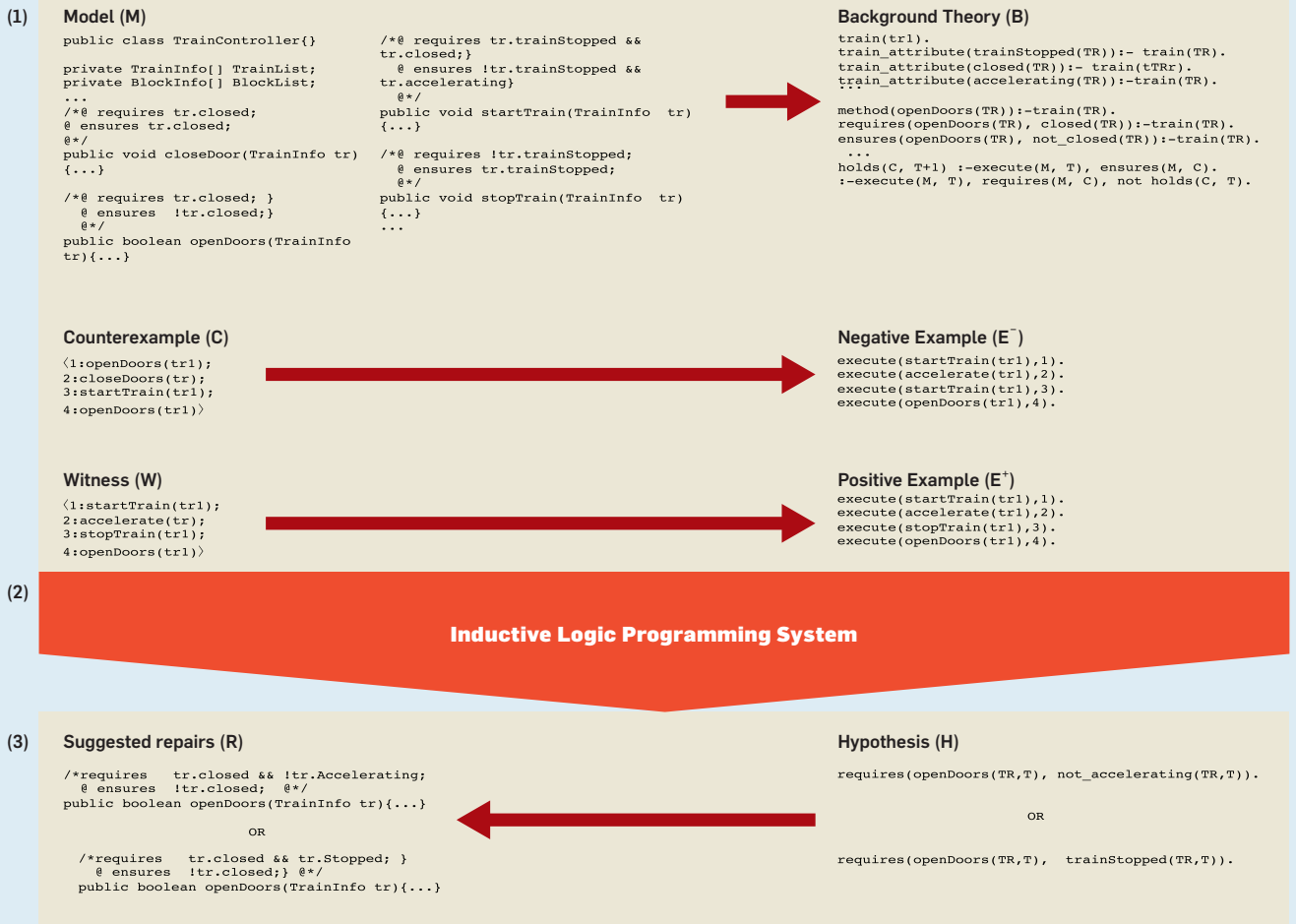
The repair scenario in Figure 4 assumes a notion of brave entailment for positive examples E^+ and a notion of cautious entailment for E^- . Although Figure 4 gives only an excerpt of the full logic program representation, it is possible to intuitively see that, according to this definition of entailment, the conjunction of atoms in E^+ is consistent with B , but the conjunction of the negative examples in E^- is also consistent with B , since all defined pre-conditions of the methods executed in the example runs are satisfied. The current description expressed by the logic program B is thus erroneous. The learning phase, in this case, must find hypotheses H , regarding method pre- and post-conditions, that together with B would ensure the execution runs represented by E^- would no longer be consistent with B . In the train-controller scenario, two alternative hypotheses are found using ASPAL.

Once the learned hypotheses are translated back into the language L_M , the software engineer can select from among the computed repairs, and the original description can be updated accordingly.

Applications. As mentioned earlier, we have successfully applied the combination of model checking and ILP to address a variety of requirements-engineering problems (see the table here). Each application consisted of a validation against several benchmark studies, including London Ambulance System (LAS), Flight Control System (FCS), Air Traffic Control System (ATCS), Engineered Safety Feature Actuation System (ESFAS) and Philips Television Set Configuration System (PTSCS).

The size of our case studies was not problematic. In general, our approach was dependent on the scalability of the underlying model checking and ILP tools influenced by the size of the formal description and proper-

Figure 4. ILP for train-controller example.



ties being verified, expressiveness of the specification language, number and size of examples, notion of entailment adopted, and characteristics of the hypotheses to be constructed. As a reference, in the goal operationalization of the ESFAS, the system model consists of 29 LTL propositional atoms and five goal expressions. We used LTL model checking, which is coNP-hard and PSPACE-complete, and XHAIL, the implementation of which is based on a search mechanism that is NP-complete. We had to perform 11 iterations to reach a fully repaired model, with an average cycle computation time of approximately 88 seconds; see Alrajeh et al.⁴ for full details on these iterations.

Related Work

Much research effort targets fault detection, diagnosis, and repair, some looking to combine verification and

machine learning in different ways; for example, Seshia¹⁷ showed tight integration of induction and deduction helps complete incomplete models through synthesis, and Seshia¹⁷ also made use of an inductive inference engine from labeled examples. Our approach is more general in both scope and technology, allowing not only for completing specifications but also for changing them altogether.

Testing and debugging are arguably the most widespread verify-diagnose-repair tasks, along with areas of runtime verification, (code) fault localization, and program repair. Runtime verification aims to address the boundary between formal verification and testing and could provide a valid alternative to model checking and logic-based learning, as we have described here. Fault localization has come a long way since

Modeling languages, tools, and case studies for requirements-engineering applications; for more on Progol5, see <http://www.doc.ic.ac.uk/~shm/Software/progol5.0>; for MTSA, see <http://sourceforge.net/projects/mtsa>; for LTSA, see <http://www.doc.ic.ac.uk/ltsa>; and for TAL, see Corapi et al.¹¹

Application	L _M	Model Checker	ILP System	Case Studies
Goal operationalization ⁴	FLTL	LTSA	XHAIL, Progol5	LAS, ESFAS
Obstacle detection ²¹	LTL	LTSA	XHAIL, ASPAL	LAS, FCS
Vacuity resolution ³	Triggered scenarios	MTSA	ASPAL, TAL	PTSCS, ATCS

Mark Weiser's²² breakthrough work on static slicing, building on dynamic slicing,¹ delta debugging,²³ and others. Other approaches to localization based on comparing invariants, path conditions, and other formulae from faulty and non-faulty program versions also show good results.¹² Within the fault localization domain, diagnosis is often based on statistical inference.¹⁴

Model checking and logic-based reasoning are used for program repair; for example, Buccafurri et al.⁸ used abductive reasoning to locate errors in concurrent programs and suggest repairs for very specific types of errors (such as variable assignment and flipped consecutive statements). This limitation was due to the lack of a reasoning framework that generalizes from ground facts. Logic-based learning allows a software engineer to compute a broader range of repairs.

A different, but relevant, approach for program synthesis emerged in early 2010¹⁸ where the emphasis was on exploiting advances in verification (such as inference of invariants) to encode a program-synthesis problem as a verification problem. Heuristic-based techniques (such as genetic algorithm-based techniques¹³) aimed to automatically change a program to pass a particular test. Specification-based techniques aim to exploit intrinsic code redundancy.⁹ Contrary to our work and that of Buccafurri et al.,⁸ none of these techniques guarantees a provably correct repair.

Theorem provers are able to facilitate diagnosing errors and repairing descriptions at the language level. Nonetheless, counterexamples play a key role in human understanding, and state-of-the-art provers (such as Nitpick⁶) have been extended to generate them. Beyond counterexample generation, repair is also being studied; for instance, in Sutcliffe and Puzis¹⁹ semantic and syntactic heuristics for selecting axioms were changed. Logic-based learning offers a means for automatically generating repairs over time rather than requiring the software engineer to predefine them. Machine-learning approaches that are not logic-based have been used in conjunction with theorem proving to find useful premises that help prove a new conjecture based on previously solved mathematical problems.²

Conclusion

To address the need for automated support for verification, diagnosis, and repair in software engineering, we recommend the combined use of model checking and logic-based learning. In this article, we have described a general framework combining model checking and logic-based learning. The ability to diagnose faults and propose correct resolutions to faulty descriptions, in the same language engineers used to develop them, is key to support for many laborious and error-prone software-engineering tasks and development of more-robust software.

Our experience demonstrates the significant benefits this integration brings and indicates its potential for wider applications, some of which were explored by Borjes et al.⁷ Nevertheless, important technical challenges remain, including support for quantitative reasoning like stochastic behavior, time, cost, and priorities. Moreover, diagnosis and repair are essential not only during software development but during runtime as well. With the increasing relevance of adaptive and autonomous systems, there is a crucial need for software-development infrastructure that can reason about observed and predicted runtime failures, diagnose their causes, and implement plans that help them avoid or recover from them. **□**

References

- Agrawal, H. and Horgan, J.R. Dynamic program slicing. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation* (White Plains, New York, June 20–22). ACM Press, New York, 1990, 246–256.
- Alama, J., Heskes, T., Kühlwein, D., Tsvitvadze, E., and Urban, J. Premise selection for mathematics by corpus analysis and kernel methods. *Journal of Automated Reasoning* 52, 2 (Feb. 2014), 191–213.
- Alrajeh, D., Kramer, J., Russo, A., and Uchitel, S. Learning from vacuously satisfiable scenario-based specifications. In *Proceedings of the 15th International Conference on Fundamental Approaches to Software Engineering* (Tallinn, Estonia, Mar. 24–Apr. 1). Springer, Berlin, 2012, 377–393.
- Alrajeh, D., Kramer, J., Russo, A., and Uchitel, S. Elaborating requirements using model checking and inductive learning. *IEEE Transaction Software Engineering* 39, 3 (Mar. 2013), 361–383.
- Alrajeh, D., Russo, A., Uchitel, S., and Kramer, J. Integrating model checking and inductive logic programming. In *Proceedings of the 21st International Conference on Inductive Logic Programming* (Windsor Great Park, U.K., July 31–Aug. 3). Springer, Berlin, 2012, 45–60.
- Blanchette, J.C. and Nipkow, T. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In *Proceedings of the first International Conference on Interactive Theorem Proving* (Edinburgh, U.K., July 11–14). Springer, Berlin, 2010, 131–146.
- Borges, R.V., d'Ávila Garcez, A.S., and Lamb, L.C. Learning and representing temporal knowledge in recurrent networks. *IEEE Transactions on Neural Networks* 22, 12 (Dec. 2011), 2409–2421.

- Buccafurri, F., Eiter, T., Gottlob, G., and Leone, N. Enhancing model checking in verification by AI techniques. *Artificial Intelligence* 112, 1–2 (Aug. 1999), 57–104.
- Carzaniga, A., Gorla, A., Mattavelli, A., Perino, N., and Pezzè, M. Automatic recovery from runtime failures. In *Proceedings of the 35th International Conference on Software Engineering* (San Francisco, CA, May 18–26). IEEE Press, Piscataway, NJ, 2013, 782–791.
- Clarke, E.M. The birth of model checking. In *25 Years of Model Checking*, O. Grumberg and H. Veith, Eds. Springer, Berlin, 2008, 1–26.
- Corapi, D., Russo, A., and Lupu, E. Inductive logic programming as abductive search. In *Technical Communications of the 26th International Conference on Logic Programming*, M. Hermenegildo and T. Schaub, Eds. (Edinburgh, Scotland, July 16–19). Schloss Dagstuhl, Dagstuhl, Germany, 2010, 54–63.
- Eichinger, F. and Böhm, K. Software-bug localization with graph mining. In *Managing and Mining Graph Data*, C.C. Aggarwal and H. Wang, Eds. Springer, New York, 2010, 515–546.
- Forrest, S., Nguyen, T., Weimer, W., and Le Goues, C. A genetic programming approach to automated software repair. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (Montreal, Canada, July 8–12). ACM Press, New York, 2009, 947–954.
- Liblit, B., Naik, M., Zheng, A.X., Aiken, A., and Jordan, M.I. Scalable statistical bug isolation. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation* (Chicago, June 12–15). ACM Press, New York, 2005, 15–26.
- Muggleton, S. and Marginean, F. Logic-based artificial intelligence. In *Logic-Based Machine Learning*, J. Minker, Ed. Kluwer Academic Publishers, Dordrecht, the Netherlands, 2000, 315–330.
- Sakama, C. and Inoue, K. Brave induction: A logical framework for learning from incomplete information. *Machine Learning* 76, 1 (July 2009), 3–35.
- Seshia, S.A. Sciduction: Combining induction, deduction, and structure for verification and synthesis. In *Proceedings of the 49th ACM/EDAC/IEEE Design Automation Conference* (San Francisco, CA, June 3–7). ACM, New York, 2012, 356–365.
- Srivastava, S., Gulwani, S., and Foster, J.S. From program verification to program synthesis. *SIGPLAN Notices* 45, 1 (Jan. 2010), 313–326.
- Sutcliffe, G. and Puzis, Y. Sraas: A semantic relevance axiom selection system. In *Proceedings of the 21st International Conference on Automated Deduction* (Bremen, Germany, July 17–20). Springer, Berlin, 2007, 295–310.
- van Lamsweerde, A. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. John Wiley & Sons, Inc., New York, 2009.
- van Lamsweerde, A. and Letier, E. Handling obstacles in goal-oriented requirements engineering. *IEEE Transaction on Software Engineering* 26, 10 (Oct. 2000), 978–1005.
- Weiser, M. Program slicing. In *Proceedings of the Fifth International Conference on Software Engineering* (San Diego, CA, Mar. 9–12). IEEE Press, Piscataway, NJ, 1981, 439–449.
- Zeller, A. Yesterday, my program worked. Today, it does not. Why? In *Proceedings of the Seventh European Software Engineering Conference* (held jointly with the Seventh ACM SIGSOFT International Symposium on Foundations of Software Engineering) (Toulouse, France, Sept. 6–10), Springer, London, 1999, 253–267.

Dalal Alrajeh (dalal.alrajeh@imperial.ac.uk) is a junior research fellow in the Department of Computing at Imperial College London, U.K.

Jeff Kramer (j.kramer@imperial.ac.uk) is a professor of distributed computing in the Department at Computing of Imperial College London, U.K.

Alessandra Russo (a.russo@imperial.ac.uk) is a reader in applied computational logic in the Department of Computing at Imperial College London, U.K.

Sebastian Uchitel (s.uchitel@imperial.ac.uk) is a reader in software engineering in the Department of Computing at Imperial College London, U.K., and an ad-honorem professor in the Departamento de Computación and the National Scientific and Technical Research Council, or CONICET, at the University of Buenos Aires, Argentina.

6th Annual ACM SIGPLAN Conference on

Systems,
Programming,
Languages, and
Applications:
Software for
Humanity

Location

Sheraton Pittsburgh at
Station Square Hotel

Events

- 30th Annual OOPSLA
- Onward!
- Wavefront
- Generative Programming and Component Engineering (GPCE)
- Dynamic Languages Symposium (DLS)
- Software Language Engineering (SLE)
- Pattern Languages of Programming (PLoP)
- and more

General Chair

Jonathan Aldrich
Carnegie Mellon University

OOPSLA Papers Chair

Patrick Eugster
Purdue University

Onward! Papers Chair

Gail Murphy
University of British Columbia

Onward! Essays Chair

Guy Steele
Oracle Labs

<http://splashcon.org>
info@splashcon.org



**From theoretical possibility
to near practicality.**

BY MICHAEL WALFISH AND ANDREW J. BLUMBERG

Verifying Computations without Reexecuting Them

IN THIS SETUP, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware.

—Babai, Fortnow, Levin, Szegedy, 1991⁴

How *can* a single PC check a herd of supercomputers with unreliable software and untested hardware?

This classic problem is particularly relevant today, as much computation is now outsourced: it is performed by machines that are rented, remote, or both. For example, service providers (SPs) now offer storage, computation, managed desktops, and more. As a result, relatively weak devices (phones, tablets, laptops, and PCs) can run computations (storage, image processing, data

analysis, video encoding, and so on) on banks of machines controlled by someone else.

This arrangement is known as cloud computing, and its promise is enormous. A lone graduate student with an intensive analysis of genome data can now rent a hundred computers for 12 hours for less than \$200. And many companies now run their core computing tasks (websites, application logic, storage) on machines owned by SPs, which automatically replicate applications to meet demand. Without cloud computing, these examples would require buying hundreds of physical machines when demand spikes ... and then selling them back the next day.

But with this promise comes risk. SPs are complex and large-scale, making it unlikely that execution is always correct. Moreover, SPs do not necessarily have strong incentives to ensure correctness. Finally, SPs are black boxes, so faults—which can include misconfigurations, corruption of data in storage or transit, hardware problems, malicious operation, and more³³—are unlikely to be detectable. This raises a central question, which goes beyond cloud computing: *How can we ever trust results computed by a third-party, or the integrity of data stored by such a party?*

A common answer is to replicate computations.^{15,16,34} However, replication assumes that failures are uncorrelated, which may not be a valid assumption: the hardware and software

» key insights

- Researchers have built systems that allow a local computer to efficiently check the correctness of a remote execution.
- This is a potentially radical development; there are many applications, such as defending against an untrusted hardware supply chain, providing confidence in cloud computing, and enabling new kinds of distributed systems.
- Key enablers are PCPs and related constructs, which have long been of intense theoretical interest.
- Bringing this theory to near practicality is the focus of an exciting new interdisciplinary research area.



platforms in cloud computing are often homogeneous. Another answer is auditing—checking the responses in a small sample—but this assumes that incorrect outputs, if they occur, are relatively frequent. Still other solutions involve trusted hardware³⁹ or attestation,³⁷ but these mechanisms require a chain of trust and assumptions that the hardware or a hypervisor works correctly.

But what if the third party returned its results along with a proof that the results were computed correctly? And what if the proof were inexpensive to check, compared to the cost of redoing the computation? Then few assumptions would be needed about the

kinds of faults that can occur: either the proof would check, or not. We call this vision *proof-based verifiable computation*, and the question now becomes: *Can this vision be realized for a wide class of computations?*

Deep results in complexity theory and cryptography tell us that in principle the answer is “yes.” Probabilistic proof systems^{24,44}—which include interactive proofs (IPs),^{3,26,32} probabilistically checkable proofs (PCPs),^{1,2,44} and argument systems¹³ (PCPs coupled with cryptographic commitments³⁰)—consist of two parties: a *verifier* and a *prover*. In these protocols, the prover can efficiently convince the verifier of a mathemati-

cal assertion. In fact, the acclaimed PCP theorem,^{1,2} together with refinements,²⁷ implies that a verifier only has to check three randomly chosen bits in a suitably encoded proof!

Meanwhile, the claim “*this* program, when executed on *this* input, produces *that* output” can be represented as a mathematical assertion of the necessary form. The only requirement is the verifier knows the program, the input (or at least a digest, or fingerprint, of the input), and the purported output. And this requirement is met in many uses of outsourced computing; examples include Map Reduce-style text processing, scientific computing and simulations, da-

tabase queries, and Web request-response.^a Indeed, although the modern significance of PCPs lies elsewhere, an original motivation was verifying the correctness of remotely executed computations: the paper quoted in our epigraph⁴ was one of the seminal works that led to the PCP theorem.

However, for decades these approaches to verifiable computation were purely theoretical. Interactive protocols were prohibitive (exponential-time) for the prover and did not appear to save the verifier work. The proofs arising from the PCP theorem (despite asymptotic improvements^{10,20}) were so long and complicated that it would have taken thousands of years to generate and check them, and would have needed more storage bits than there are atoms in the universe.

But beginning around 2007, a number of theoretical works achieved results that were especially relevant to the problem of verifying cloud computations. Goldwasser et al., in their influential Muggles paper,²⁵ refocused the theory community's attention on verifying outsourced computations, in the context of an interactive proof system that required only *polynomial* work from the prover, and that ap-

plied to computations expressed as certain kinds of circuits. Ishai et al.²⁹ proposed a novel cryptographic commitment to an entire linear function, and used this primitive to apply *simple* PCP constructions to verifying general-purpose outsourced computations. A couple of years later, Gentry's breakthrough protocol for fully homomorphic encryption (FHE)²³ led to work (GGP) on *non-interactive* protocols for general-purpose computations.^{17,21} These developments were exciting, but, as with the earlier work, implementations were thought to be out of the question. So the theory continued to remain theory—until recently.

The last few years have seen a number of projects overturn the conventional wisdom about the hopeless impracticality of proof-based verifiable computation. These projects aim squarely at building real systems based on the theory mentioned earlier, specifically PCPs and Muggles (FHE-based protocols still seem too expensive). The improvements over naive theoretical protocols are dramatic; it is not uncommon to read about factor-of-a-trillion speedups. The projects take different approaches, but broadly speaking, they apply both refinements of the theory and systems techniques. Some projects include a full pipeline: a programmer specifies a computation in a high-level language, and then a compiler (a) transforms the computation to the for-

malism that the verification machinery uses and (b) outputs executables that implement the verifier and prover. As a result, achieving verifiability is no harder for the programmer than writing the code in the first place.

The goal of this article is to survey this blossoming area of research. This is an exciting time for work on verifiable computation: while none of the works we discuss here is practical enough for its inventors to raise venture capital for a startup, they merit being referred to as “systems.” Moreover, many of the open problems cut across subdisciplines of computer science: programming languages, parallel computing, systems, complexity theory, and cryptography. The pace of progress has been rapid, and we believe real applications of these techniques will appear in the next few years.

A note about scope. We focus on solutions that provide integrity to, and in principle are very efficient for, the verifier.^b Thus, we do not cover exciting work on efficient implementations of secure multiparty protocols.^{28,31} We also exclude FHE-based approaches based on GGP²¹ (as noted earlier, these techniques seem too expensive) and the vast body of domain-specific solutions (surveyed elsewhere^{36,42,47}).

A Problem and Theoretical Solutions

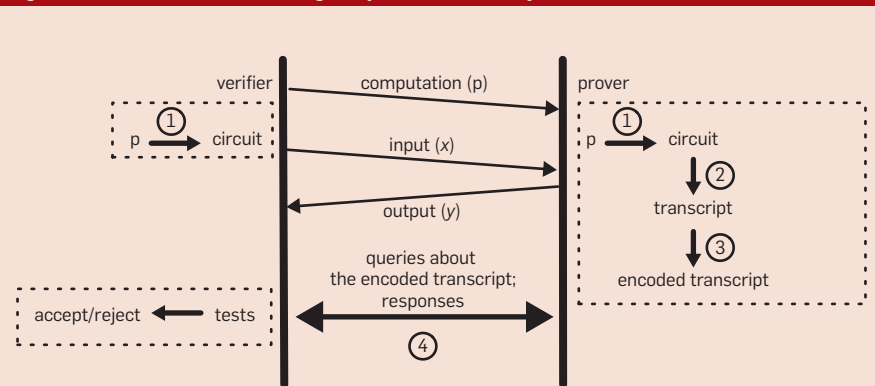
The problem statement and some observations about it. A verifier sends the specification of a computation p (for example, the text of a program) and input x to a *prover*. The prover computes an output y and returns it to the verifier. If $y = p(x)$, then a correct prover should be able to convince the verifier of y 's correctness, either by answering some questions or by providing a certificate of correctness. Otherwise, the verifier should reject y with high probability.

In any protocol that solves this problem, we desire three things. First, the protocol should provide some advantage to the verifier: either the protocol should be cheaper for the verifier than executing $p(x)$ locally, or else the protocol should handle computations p that

a The condition does not hold for “proprietary” computations whose logic is concealed from the verifier. However, the theory can be adapted to this case too, as we discuss near the end of the article.

b Some of the systems (those known as *zero knowledge SNARKs*) also keep the prover's input private.

Figure 1. A framework for solving the problem in theory.



Framework in which a verifier can check that, for a computation p and desired input x , the prover's purported output y is correct. Step 1: The verifier and prover compile p , which is expressed in a high-level language (for example, C) into a Boolean circuit, C . Step 2: the prover executes the computation, obtaining a transcript for the execution of C on x . Step 3: the prover encodes the transcript, to make it suitable for efficient querying by the verifier. Step 4: the verifier probabilistically queries the encoded transcript; the structure of this step varies among the protocols (for example, in some of the works,^{7,36} explicit queries are established before the protocol begins, and this step requires sending only the prover's responses).

the verifier could not execute itself (for example, operations on state private to the prover). Second, we do not want to make any assumptions that the prover follows the protocol. Third, p should be general; later, we will have to make some compromises, but for now, p should be seen as encompassing all C programs whose running time can be statically bounded given the input size.

Some reflections about this setup are in order. To begin with, we are willing to accept some overhead for the prover, as we expect assurance to have a price. Something else to note is that whereas some approaches to computer security attempt to reason about what *incorrect* behavior looks like (think of spam detection, for instance), we will specify *correct* behavior and ensure anything other than this behavior is visible as such; this frees us from having to enumerate, or reason about, the possible failures of the prover.

Finally, one might wonder: How does our problem statement relate to NP-complete problems, which are easy to check but believed to be hard to solve? The answer is that the “check” of an NP solution requires the checking entity to do work polynomial in the length of the solution, whereas our verifier will do far less work than that! (Randomness makes this possible.) Another answer is that many computations (for example, those that run in deterministic polynomial time) do not admit an asymmetric checking structure—unless one invokes the fascinating body of theory that we turn to now.

A framework for solving the problem in theory. A framework for solving the problem in theory is depicted in Figure 1. Because *Boolean circuits* (networks of AND, OR, NOT gates) work naturally with the verification machinery, the first step is for the verifier and prover to transform the computation to such a circuit. This transformation is possible because any of our computations p is naturally modeled by a Turing Machine (TM); meanwhile, a TM can be “unrolled” into a Boolean circuit that is not much larger than the number of steps in the computation.

Thus, from now on, we will talk only about the circuit C that represents our computation p (Figure 1, step 1). Consistent with the problem statement earlier, the verifier supplies the input

Encoding a Circuit's Execution in a Polynomial

This sidebar demonstrates a connection between *program execution* and *polynomials*. As a warmup, consider an AND gate, with two (binary) inputs, z_1, z_2 . One can represent its execution as a *function*:

$$\text{AND}(z_1, z_2) = z_1 \cdot z_2.$$

Here, the function AND behaves exactly as the gate would: it evaluates to 1 if z_1 and z_2 are both 1, and it evaluates to 0 in the other three cases. Now, consider this function of three variables:

$$\begin{aligned} f_{\text{AND}}(z_1, z_2, z_3) &= z_3 - \text{AND}(z_1, z_2) \\ &= z_3 - z_1 \cdot z_2. \end{aligned}$$

Observe that $f_{\text{AND}}(z_1, z_2, z_3)$ evaluates to 0 when, and only when, z_3 equals the AND of z_1 and z_2 . For example, $f_{\text{AND}}(1, 1, 1) = 0$ and $f_{\text{AND}}(0, 1, 0) = 0$ (both of these cases correspond to correct computation by an AND gate), but $f_{\text{AND}}(0, 1, 1) \neq 0$.

We can do the same thing with an OR gate:

$$f_{\text{OR}}(z_1, z_2, z_3) = z_3 - z_1 - z_2 + z_1 \cdot z_2.$$

For example, $f_{\text{OR}}(0, 0, 0) = 0$, $f_{\text{OR}}(1, 1, 1) = 0$, and $f_{\text{OR}}(0, 1, 0) \neq 0$. In all of these cases, the function is determining whether its third argument (z_3) does in fact represent the OR of its first two arguments (z_1 and z_2). Finally, we can do this with a NOT gate:

$$f_{\text{NOT}}(z_1, z_2) = 1 - z_1 + z_2.$$

The intent of this warmup is to communicate that *the correct execution of a gate can be encoded in whether some function evaluates to 0*. Such a function is known as an *arithmetization* of the gate.

Now, we extend the idea to a line $L(t)$ over a dummy variable, t :

$$L(t) = (z_3 - z_1 \cdot z_2) \cdot t.$$

This line is parameterized by z_1, z_2 , and z_3 ; depending on their values, $L(t)$ becomes different lines. A crucial fact is that this line is the 0-line (that is, it covers the horizontal axis, or equivalently, evaluates to 0 for all values of t) if and only if z_3 is the AND of z_1 and z_2 . This is because the y-intercept of $L(t)$ is always 0, and the slope of $L(t)$ is given by the function f_{AND} . Indeed, if $(z_1, z_2, z_3) = (1, 1, 0)$, which corresponds to an incorrect computation of AND, then $L(t) = t$, a line that crosses the horizontal axis only once. On the other hand, if $(z_1, z_2, z_3) = (0, 1, 0)$, which corresponds to a correct computation of AND, then $L(t) = 0 \cdot t$, which is 0 for all values of t .

We can generalize this idea to higher order polynomials (a line is just a degree-1 polynomial). Consider the following degree-2 polynomial, or parabola, $Q(t)$ in the variable t :

$$Q(t) = [z_1 \cdot z_2 (1 - z_3) + z_3 (1 - z_1 \cdot z_2)] t^2 + (z_3 - z_1 \cdot z_2) \cdot t.$$

As with $L(t)$, the parabola $Q(t)$ is parameterized by z_1, z_2 , and z_3 ; they determine the coefficients. And as with $L(t)$, this parabola is the 0 parabola (all coefficients are 0, causing the parabola to evaluate to 0 for all values of t) if and only if z_3 is the AND of z_1 and z_2 . For example, if $(z_1, z_2, z_3) = (1, 1, 0)$, which is an incorrect computation of AND, then $Q(t) = t^2 - t$, which crosses the horizontal axis only at $t = 0$ and $t = 1$. On the other hand, if $(z_1, z_2, z_3) = (0, 1, 0)$, which is a *correct* computation of AND, then $Q(t) = 0 \cdot t^2 + 0 \cdot t$, which of course is 0 for all values of t .

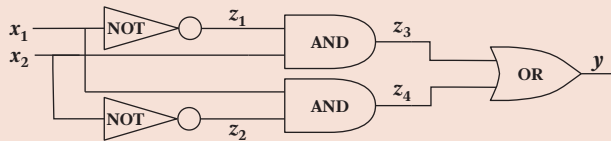
Summarizing, $L(t)$ (resp., $Q(t)$) is the 0-line (resp., 0-parabola) when and only when $z_3 = \text{AND}(z_1, z_2)$. This concept is powerful, for if there is an efficient way to check whether a polynomial is 0, then there is now an efficient check of whether a circuit was executed correctly (here, we have generalized to circuit from gate). And there are indeed such checks of polynomials, as described in the sidebar on page 78.

Probabilistically Checking a Transcript's Validity

This sidebar explains the idea behind a fast probabilistic check of a transcript's validity. As noted in the text, computations are expressed as Boolean circuits. As an example, consider the following computation, where x_1 and x_2 are bits:

```
if ( $x_1 \neq x_2$ ) {  $y = 1$  } else {  $y = 0$  }
```

This computation could be represented by a single XOR gate; for illustration, we represent it in terms of AND, OR, NOT:

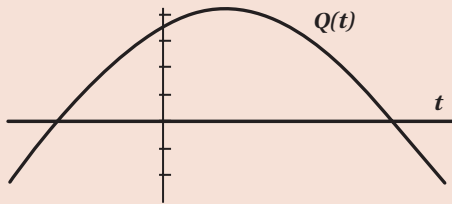


To establish the correctness of a purported output y given inputs x_1, x_2 , the prover must demonstrate to the verifier that it has a *valid transcript* (see text) for this circuit. A naive way to do this is for the prover to simply send the transcript to the verifier, and for the verifier to check it step-by-step. However, that would take as much time as the computation.

Instead, the two parties encode the computation as a polynomial $Q(t)$ over a dummy variable t . The sidebar on page 77 gives an example of this process for a single gate, but the idea generalizes to a full circuit. The result is a polynomial $Q(t)$ that evaluates to 0 for all t if and only if each gate's output in the transcript follows correctly from its inputs.

Generalizing the single-gate case, the coefficients of $Q(t)$ are given by various combinations of $x_1, x_2, z_1, z_2, z_3, z_4, y$. Variables corresponding to inputs x_1, x_2 and output y are hard-coded, ensuring that the polynomial expresses a computation based on the correct inputs and the purported output.

Now, the verifier wants a probabilistic and efficient check that $Q(t)$ is 0 everywhere (see sidebar, page 77). A key fact is: if a polynomial is *not* the zero polynomial, it has few roots (consider a parabola: it crosses the horizontal axis a maximum of two times). For example, if we take $x_1 = 0, x_2 = 0, y = 1$, which is an *incorrect* execution of the above circuit, then the corresponding polynomial might look like this:



and a polynomial corresponding to a *correct* execution is simply a horizontal line on the axis.

The check, then, is the following. The verifier chooses a random value for t (call it τ) from a pre-existing range (for example, integers between 0 and M , for some M), and evaluates Q at τ . The verifier accepts the computation as correct if $Q(\tau) = 0$ and rejects otherwise. This process occasionally produces errors since even a non-zero polynomial Q is zero sometimes (the idea here is a variant of “a stopped clock is right twice per day”), but this event happens rarely and is independent of the prover's actions.

But how does the verifier actually evaluate $Q(\tau)$? Recall that our setup, for now, is that the prover sends a (possibly long) *encoded transcript* to the verifier. The sidebar on page 81 explains what is in the encoded transcript, and how it allows the verifier to evaluate $Q(\tau)$.

x , and the prover executes the circuit C on input x and claims the output is y .^c In performing this step, the prover is expected to obtain a *valid transcript* for $\{C, x, y\}$ (Figure 1, step 2). A *transcript* is an assignment of values to the circuit wires; in a valid transcript for $\{C, x, y\}$, the values assigned to the input wires are those of x , the intermediate values correspond to the correct operation of each gate in C , and the values assigned to the output wires are y . Notice that if the claimed output is incorrect—that is, if $y \neq p(x)$ —then a valid transcript for $\{C, x, y\}$ simply does not exist.

Therefore, if the prover could establish a valid transcript exists for $\{C, x, y\}$, this would convince the verifier of the correctness of the execution. Of course, there is a simple proof that a valid transcript exists: the transcript itself. However, the verifier can check the transcript only by examining all of it, which would be as much work as having executed p in the first place.

Instead, the prover will *encode* the transcript (Figure 1, step 3) into a longer string. The encoding lets the verifier detect a transcript's validity by inspecting a small number of randomly chosen locations in the encoded string and then applying efficient tests to the contents found therein. The machinery of PCPs, for example, allows exactly this (see the three accompanying sidebars).

However, we still have a problem. The verifier cannot get its hands on the entire encoded transcript; it is longer—astronomically longer, in some cases—than the plain transcript, so reading in the whole thing would again require too much work from the verifier. Furthermore, we do not want the prover to have to write out the whole encoded transcript: that would also be too much work, much of it wasteful, since the verifier looks at only small pieces of the encoding. And unfortunately, we cannot have the verifier simply ask the prover

^c The framework also handles circuits where the prover supplies some of the inputs and receives some of the outputs (enabling computations over remote state inaccessible to the verifier). However, the accompanying techniques are mostly beyond our scope (we will briefly mention them later). For simplicity we are treating p as a pure computation.

what the encoding holds at particular locations, as the protocols depend on the element of surprise. That is, if the verifier's queries are known in advance, then the prover can arrange its answers to fool the verifier.

As a result, the verifier must issue its queries about the encoding carefully (Figure 1, step 4). The literature describes three separate techniques for this purpose. They draw on a richly varied set of tools from complexity theory and cryptography, and are summarized next. Afterward, we discuss their relative merits.

► *Use the power of interaction.* One set of protocols proceeds in rounds: the verifier queries the prover about the contents of the encoding at a particular location, the prover responds, the verifier makes another query, the prover responds, and so on. Just as a lawyer's questions to a witness restrict the answers the witness can give to the next question, until a lying witness is caught in a contradiction, the prover's answers in each round about what the encoding holds limit the space of valid answers in the next round. This continues until the last round, at which point a prover that has answered perfidiously at any point—by answering based on an invalid transcript or by giving answers that are untethered to any transcript—simply has no valid answers. This approach relies on interactive proof protocols,^{3,26,32} most notably Muggles,²⁵ which was refined and implemented.^{18,45–47}

► *Extract a commitment.* These protocols proceed in two rounds. The verifier first requires the prover to *commit* to the full contents of the encoded transcript; the commitment relies on standard cryptographic primitives, and we call the committed-to contents a *proof*. In the second round, the verifier generates queries—locations in the proof the verifier is interested in—and then asks the prover what values the proof contains at those locations; the prover is forced to respond consistent with the commitment. To generate queries and validate answers, the verifier uses PCPs (they enable probabilistic checking, as described in the sidebar entitled “Probabilistically Checkable Proofs”). This approach



**This is an exciting
time for work
on verifiable
computation.**



was outlined in theory by Kilian,³⁰ building on the PCP theorem.^{1,2} Later, Ishai et al.²⁹ (IKO) gave a drastic simplification, in which the prover can commit to a proof without materializing the whole thing. IKO led to a series of refinements, and implementation in a system.^{40–43,47}

► *Hide the queries.* Instead of extracting a commitment and then revealing its queries, the verifier pre-encrypts its queries—as with the prior technique, the queries describe locations where the verifier wants to inspect an eventual proof, and these locations are chosen by PCP machinery—and sends this description to the prover prior to their interaction. Then, during the verification phase, powerful cryptography achieves the following: the prover answers the queries without being able to tell which locations in the proof are being queried, and the verifier recovers the prover's answers. The verifier then uses PCP machinery to check the answers, as in the commitment-based protocols. The approach was described in theory by Gennaro et al.²² (see also Bitansky et al.¹²), and refined and implemented in two projects.^{7,9,36}

Progress: Implemented Systems

The three techniques described are elegant and powerful, but as with the prior technique, naive implementations would result in preposterous costs. The research projects that implemented these techniques have applied theoretical innovations and serious systems work to achieve *near* practical performance. Here, we explain the structure of the design space, survey the various efforts, and explore their performance (in doing this, we will illustrate what “near practical” means).

We restrict our attention to *implemented systems* with published experimental results. By “system,” we mean code (preferably publically released) that takes some kind of representation of a computation and produces executables for the verifier and the prover that run on stock hardware. Ideally, this code is a compiler toolchain, and the representation is a program in a high-level language.

The landscape. As depicted in Figure 2, we organize the design space in

terms of a three-way trade-off among cost, expressiveness, and functionality.^d Here, cost mainly refers to *setup costs* for the verifier; as we will see, this cost is the verifier's largest expense, and affects whether a system meets the goal of saving the verifier work. (This setup cost also correlates with the prover's cost for most of the systems discussed.) By expressiveness, we mean the class of computations the system can handle while providing a benefit to the verifier. By functionality, we mean whether the works provide properties like non-interactivity (setup costs amortize indefinitely), *zero knowledge*^{24,26} (the computation transcript is hidden from the verifier, giving the prover some privacy), and public verifiability (anyone, not just a particular verifier, can check a proof, provided that the party who generated the queries is trusted).

CMT, Allspice, and Thaler. One line of work uses “the power of interaction;” it starts from Muggles,²⁵ the interactive proof protocol mentioned earlier. CMT^{18,46} exploits an algebraic insight to save orders of magnitude for the prover, versus a naive implementation of Muggles.

For circuits to which CMT applies, performance is very good, in part because Muggles and CMT do not use cryptographic operations. In fact, refinements by Thaler⁴⁵ provide a prover

that is optimal for certain classes of computations: the costs are only a constant factor (10–100×, depending on choice of baseline) over executing the computation locally. Moreover, CMT applies in (and was originally designed for) a streaming model, in which the verifier processes and discards input as it comes in.

However, CMT's expressiveness is limited. First, it imposes requirements on the circuit's geometry: the circuit must have structurally similar parallel blocks. Of course, not all computations can be expressed in that form. Second, the computation cannot use order comparisons (less-than, and so on).

Allspice⁴⁷ has CMT's low costs but achieves greater expressiveness (under the amortization model described next).

Pepper, Ginger, and Zaatar. Another line of work builds on the “extract a commitment” technique (called an “efficient argument” in the theory literature.^{13,30}). Pepper⁴² and Ginger⁴³ refine the protocol of IKO. To begin with, they represent computations as arithmetic constraints (that is, a set of equations over a finite field); a solution to the constraints corresponds to a valid transcript of the computation. This representation is often far more concise than Boolean circuits (used by IKO and in the proof of the PCP theorem¹) or arithmetic circuits (used by CMT). Pepper and Ginger also strengthen IKO's commitment primitive, explore low-overhead PCP encodings for certain computations, and apply a number of systems techniques (such as parallelization on

distributed systems).

Pepper and Ginger dramatically reduce costs for the verifier and prover, compared to IKO. However, as in IKO, the verifier incurs setup costs. Both systems address this issue via amortization, reusing the setup work over a *batch*: multiple *instances* of the same computation, on different inputs, verified simultaneously.

Pepper requires describing constraints manually. Ginger has a compiler that targets a larger class of computations; also, the constraints can have auxiliary variables set by the prover, allowing for efficient representation of not-equal-to checks and order comparisons. Still, both handle only straight line computations with repeated structure, and both require special-purpose PCP encodings.

Zaatar⁴¹ composes the commitment protocol of Pepper and Ginger with a new linear PCP;^{1,29} this PCP adapts an ingenious algebraic encoding of computations from GGPR²² (which we return to shortly). The PCP applies to all pure computations; as a result, Zaatar achieves Ginger's performance but with far greater generality.

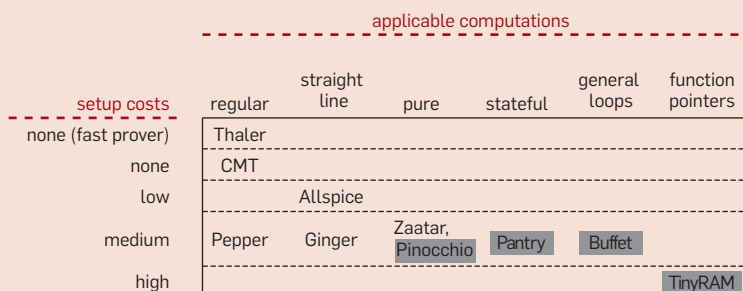
Pinocchio. Pinocchio³⁶ instantiates the technique of hiding the queries. Pinocchio is an implementation of GGPR, which is a *noninteractive* argument. GGPR can be viewed as a probabilistically checkable encoding of computations that is akin to a PCP (this is the piece that Zaatar adapts) plus a layer of sophisticated cryptography.^{12,41} GGPR's encoding is substantially more concise than prior approaches, yielding major reductions in overhead.

The cryptography also provides many benefits. It hides the queries, which allows them to be reused. The result is a protocol with minimal interaction (after a per-computation setup phase, the verifier sends only an instance's input to the prover) and, thus, qualitatively better amortization behavior. Specifically, Pinocchio amortizes per-computation setup costs over all future instances of a given computation; by contrast, recall that Zaatar and Allspice amortize their per-computation costs only over a batch. GGPR's and Pinocchio's cryptography also yield zero knowledge and public verifiability.

Compared to Zaatar, Pinocchio brings some additional expense in the

d For space, we omit recent work that is optimized for specific classes of computations; these works can be found with this article in the ACM Digital Library under Supplemental Material.

Figure 2. Design space of implemented systems for proof-based verifiable computation.



There is a three-way trade-off among cost, expressiveness, and functionality. Higher in the figure means lower cost, and rightward generally means better expressiveness. The shaded systems achieve non-interactivity, zero knowledge, and other cryptographic properties. (Pantry, Buffet, and TinyRAM achieve these properties by leveraging Pinocchio.) Here, “regular” means structurally similar parallel blocks; “straight line” means not many conditional statements; and “pure” means computations without side effects.

prover's costs and the verifier's setup costs. Pinocchio's compiler initiated the use of C syntax in this area, and includes some program constructs not present in prior work. The underlying computational model (unrolled executions) is essentially the same as Ginger's and Zaatar's.^{41,43}

Although the systems we have described so far have made tremendous progress, they have done so within a programming model that is not reflective of real-world computations. First, these systems require loop bounds to be known at compile time. Second, they do not support indirect memory references scalably and efficiently, ruling out RAM and thus general-purpose programming. Third, the verifier must handle all inputs and outputs, a limitation that is at odds with common uses of the cloud. For example, it is unreasonable to insist that the verifier materialize the entire (massive) input to a MapReduce job. The projects described next address these issues.

TinyRAM (BCGTV and BCTV). BCGTV⁷ compiles programs in C (not just a subset) to an innovative circuit representation.⁶ Applying prior insights,^{12,22,41} BCGTV combines this circuit with proof machinery (including transcript encoding and queries) from Pinocchio and GGPR.

BCGTV's circuit representation consists of the unrolled execution of a general-purpose MIPS-like CPU, called TinyRAM (and for convenience we sometimes use "TinyRAM" to refer to BCGTV and its successors). The circuit-as-unrolled-processor provides a natural representation for language features like data-dependent looping, control flow, and self-modifying code. BCGTV's circuit also includes a permutation network that elegantly and efficiently enforces the correctness of RAM operations.

BCTV⁹ improves on BCGTV by retrieving program instructions from RAM (instead of hard-coding them in the circuit). As a result, all executions with the same number of steps use the same circuits, yielding the best amortization behavior in the literature: setup costs amortize over all computations of a given length. BCTV also includes an optimized implementation of Pinocchio's protocol that cuts costs by constant factors.

Despite these advantages, the gen-

Probabilistically Checkable Proofs (simplified)

This sidebar answers the following question: how does the prover encode its transcript, and how does the verifier use this encoded transcript to evaluate Q at a randomly chosen point? (The encoded transcript is known as a probabilistically checkable proof, or PCP. For the purposes of this sidebar, we assume that the prover sends the PCP to the verifier; in the main text, we will ultimately avoid this transmission, using commitment and other techniques.)

A naive solution is a protocol in which: the prover claims that it is sending $\{Q(0), \dots, Q(M)\}$ to the verifier, the verifier chooses one of these values at random, and the verifier checks whether the randomly chosen value is 0. However, this protocol does not work: even if there is no valid transcript, the prover could cause the verifier's "check" to always pass, by sending a string of zeroes.

Instead, the prover will encode the transcript, z , and the verifier will impose structure on this encoding; in this way, both parties together form the required polynomial Q . This process is detailed in the rest of this sidebar, which will be somewhat more technical than the previous sidebars. Nevertheless, we will be simplifying heavily; readers who want the full picture are encouraged to consult the tutorials referenced in the supplemental material (online).

As a warmup, observe that we can rewrite the polynomial Q by regarding the " z " variables as unknowns. For example, the polynomial $Q(t)$ in the sidebar on page 77 can be written as:

$$Q(t, z_1, z_2, z_3) = (-2t^2) \cdot z_1 \cdot z_2 \cdot z_3 + (t^2 - t) \cdot z_1 \cdot z_2 + (t^2 + t) \cdot z_3.$$

An important fact is that for any circuit, the polynomial Q that encodes its execution can be represented as a linear combination of the components of the transcript and pairwise products of components of the transcript. We will now state this fact using notation. Assume that there are n circuit wires, labeled $(z_1, \dots, z_n)$, and arranged as a vector $\vec{z}$. Further, let $\langle \vec{a}, \vec{b} \rangle$ denote the dot product between two vectors, and let $\vec{a} \otimes \vec{b}$ denote a vector whose components are all pairs $a_i b_j$. Then we can write Q as

$$Q(t, \vec{z}) = g_0(t) + \langle \vec{g}_1(t), \vec{z} \rangle + \langle \vec{g}_2(t), \vec{z} \otimes \vec{z} \rangle,$$

where g_0 is a function from t to scalars, and $\vec{g}_1$ and $\vec{g}_2$ are functions from t to vectors.

Now, what if the verifier had a table that contained $\langle \vec{g}_1, \vec{z} \rangle$ for all vectors $\vec{z}$ (in a finite vector space), and likewise a table that contained $\langle \vec{g}_2, \vec{z} \otimes \vec{z} \rangle$ for all vectors $\vec{z}$? Then, the verifier could evaluate $Q(\tau)$ by inspecting the tables in only one location each. Specifically, the verifier would randomly choose τ , then compute $g_0(\tau)$, $\vec{g}_1(\tau)$, and $\vec{g}_2(\tau)$; then use the two tables to look up $\langle \vec{g}_1(\tau), \vec{z} \rangle$ and $\langle \vec{g}_2(\tau), \vec{z} \otimes \vec{z} \rangle$; and add these values to $g_0(\tau)$. If the tables were produced correctly, this final sum (of scalars) will yield $Q(\tau, \vec{z})$.

However, a few issues remain. The verifier cannot know that it actually received tables of the correct form, or that the tables are consistent with each other. So the verifier performs additional spot checks; the rough idea is that if the tables deviate too heavily from the correct form, then the spot checks will pick up the divergence with high probability (and if the tables deviate from the correct form but only mildly, the verifier still recovers $Q(\tau)$).

At this point, we have answered the question at the beginning of this sidebar: the correct encoding of a valid transcript z is the two tables of values. In other words, these two tables form the probabilistically checkable proof, or PCP.

Notice that the two tables are exponentially larger than the transcript. Therefore, the prover cannot send them to the verifier or even materialize them. The purpose of the three techniques discussed in the text—interactivity, commitment, hide the queries—is, roughly speaking, to allow the verifier to query the prover about the tables without either party having to materialize or handle them.

eral approach brings a steep price: BCTV's circuits are orders of magnitude larger than Pinocchio's and Zaatar's for the same high-level programs. As a result, the verifier's setup work and the prover's costs are orders of magnitude higher, and BCTV is restricted to very short executions. Nevertheless, BCGTV and BCTV intro-

duce important tools.

Pantry and Buffet. Pantry¹⁴ extends the computational model of Zaatar and Pinocchio, and works with both systems. Pantry provides a general-purpose approach to state, yielding a RAM abstraction, verifiable Map Reduce, verifiable queries on remote databases, and—using Pinocchio's

zero knowledge variant—computations that keep the prover’s state private. To date, Pantry is the only system to extensively use the capability of argument protocols (the “extract a commitment” and “hide the queries” techniques) to handle computations for which the verifier does not have all the input. In Pantry’s approach—which instantiates folklore techniques—the verifier’s explicit input includes a digest of the full input or state, and the prover is obliged to work over state that matches this digest.

Under Pantry, every operation against state compiles into the evaluation of a cryptographic hash function. As a result, a memory access is tens of thousands of times more costly than a basic arithmetic operation.

Buffet⁴⁸ combines the best features of Pantry and TinyRAM. It slashes the cost of memory operations in Pantry by adapting TinyRAM’s RAM abstraction. Buffet also brings data-dependent looping and control flow to Pantry (without TinyRAM’s expense), using a loop flattening technique inspired by the compilers literature. As a result, Buffet supports an expansive subset of C (disallowing only function pointers and goto) at costs orders of magnitude lower than both Pantry and TinyRAM. As of this writing, Buffet appears to achieve the best mix of performance and generality in the literature.

A brief look at performance. We will

answer three questions:

1. *How do the verifier’s variable (per-instance) costs compare to the baseline of local, native execution?* For some computations, this baseline is an alternative to verifiable outsourcing.

2. *What are the verifier’s setup costs, and how do they amortize?* In many of the systems, setup costs are significant and are paid for only over multiple instances of the same circuit.

3. *What is the prover’s overhead?*

We focus only on CPU costs. On the one hand, this focus is conservative: verifiable outsourcing is motivated by more than CPU savings for the verifier. For example, if inputs are large or inaccessible, verifiable outsourcing saves network costs (the naive alternative is to download the inputs and locally execute); in this case, the CPU cost of local execution is irrelevant. On the other hand, CPU costs provide a good sense of the overall expense of the protocols. (For evaluations that take additional resources into account, see Braun et al.¹⁴)

The data we present is from re-implementations of the various systems by members of our research group, and essentially match the published results. All experiments are run on the same hardware (Intel Xeon E5-2680 processor, 2.7Ghz, 32GB RAM), with the prover on one machine and the verifier on another. We perform three runs per experiment; experimental variation is minor, so we just report

the average. Our benchmarks are 128×128 matrix multiplication (of 64-bit quantities, with full precision arithmetic) and PAM clustering of 20 vectors, each of dimension 128. We do not include data for Pantry and Buffet since their innovations do not apply to these benchmarks (their performance would be the same as Zaatar or Pinocchio, depending on which machinery they were configured to run with). For TinyRAM, we report extrapolated results since, as noted earlier, TinyRAM on current hardware is restricted to executions much smaller than these benchmarks.

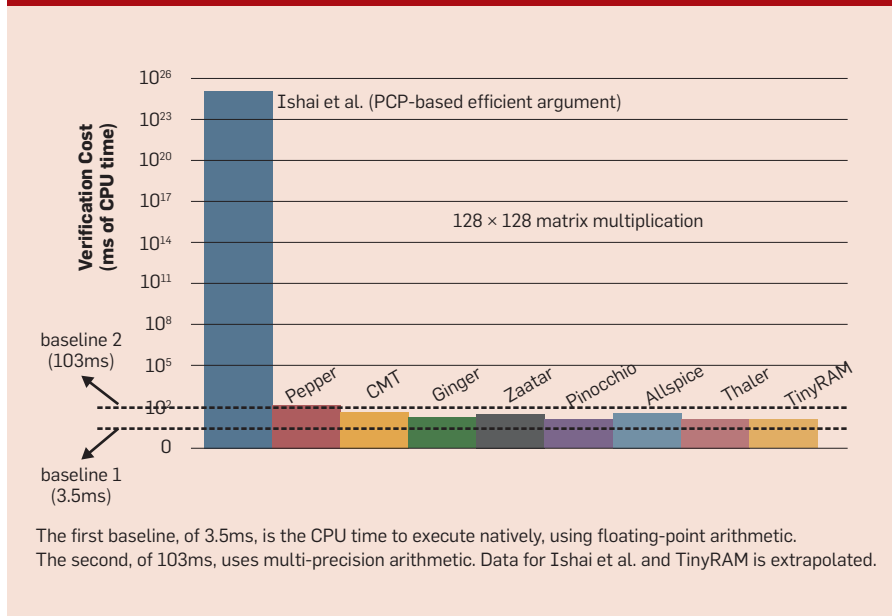
Figure 3 depicts per-instance verification costs, for matrix multiplication, compared to two baselines. The first is a native execution of the standard algorithm, implemented with floating-point operations; it costs 3.5ms, and beats all of the systems at the given input size.^e (At larger input sizes, the verifier would do better than native execution: the verifier’s costs grow linearly in the input size, which is only $O(m^2)$; local execution is $O(m^3)$.) The second is an implementation of the algorithm using a multi-precision library; this baseline models a situation in which complete precision is required.

We evaluate setup costs by asking about the *cross-over point*: how many instances of a computation are required to amortize the setup cost in the sense the verifier spends fewer CPU cycles on outsourcing versus executing locally? Figure 4 plots total cost lines and cross-over points versus the second baseline.

To evaluate prover overhead, Figure 5 normalizes the prover’s cost to the floating-point baseline.

Summary and discussion. Performance differences among the systems are overshadowed by the general nature of costs in this area. The verifier is practical if its computation is amenable to one of the less expensive (but more restricted) protocols, or if there are a large number of instances that will be run (on different inputs). And when state is remote, the verifier does not need to be faster than local computation because it would be difficult—or impossible, if the remote state is private—for the verifier to perform the computation itself (such

Figure 3. Per-instance verification costs applied to 128×128 matrix multiplication of 64-bit numbers.



^e Systems that report verification costs beating local execution choose very expensive baselines for local computation.^{36,41–43}

applications are evaluated elsewhere¹⁴).

The prover, of course, has terrible overhead: several orders of magnitude (though as noted previously, this still represents tremendous progress versus the prior costs). The prover's practicality thus depends on your ability to construct appropriate scenarios. Maybe you're sending Will Smith and Jeff Goldblum into space to save Earth; then you care a lot more about correctness than costs (a calculus that applies to ordinary satellites, too). More prosaically, there is a scenario with an abundance of server CPU cycles, many instances of the same computation to verify, and remotely stored inputs: data-parallel cloud computing. Verifiable MapReduce¹⁴ is therefore an encouraging application.

Open Questions and Next Steps

The main issue in this area is performance, and the biggest problem is the prover's overhead. The verifier's per-instance costs are also too high. And the verifier's setup costs would ideally be controlled while retaining expressivity. (This is theoretically possible,^{8,11} but overhead is very high: in a recent implementation,⁸ the prover's computational costs are orders of magnitude higher than in TinyRAM.)

The computational model is a critical area of focus. Can we identify or develop programming languages that are expressive yet compile efficiently to the circuit or constraint formalism? More generally, can we move beyond this intrinsically costly formalism?

There are also questions in systems. For example, can we develop a realistic database application, including concurrency, relational structures, and so on? More generally, an important test for this area—so far unmet—is to run experiments at realistic scale.

Another interesting area of investigation concerns privacy. By leveraging Pinocchio, Pantry has experimented with simple applications that hide the prover's state from the verifier, but there is more work to be done here and other notions of privacy that are worth providing. For example, we can provide verifiability while concealing the program that is executed (by composing techniques from Pantry, Pinocchio, and TinyRAM). A speculative application is to produce versions of

Bitcoin in which transactions can be conducted anonymously, in contrast to the status quo.^{5,19}

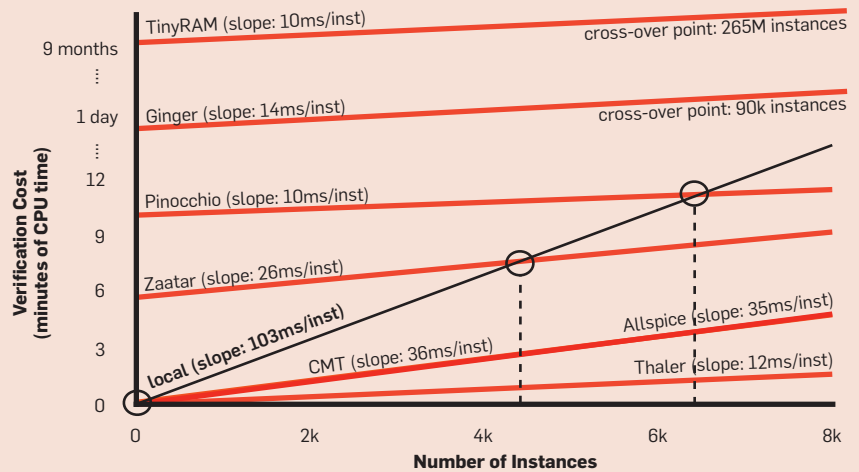
Reflections and Predictions

It is worth recalling that the intellectual foundations of this research area really had nothing to do with practice. For example, the PCP theorem is a landmark achievement of complexity theory, but if we were to implement the theory as pro-

posed, generating the proofs, even for simple computations, would have taken longer than the age of the universe. In contrast, the projects described in this article have not only built systems from this theory but also performed experimental evaluations that terminate before publication deadlines.

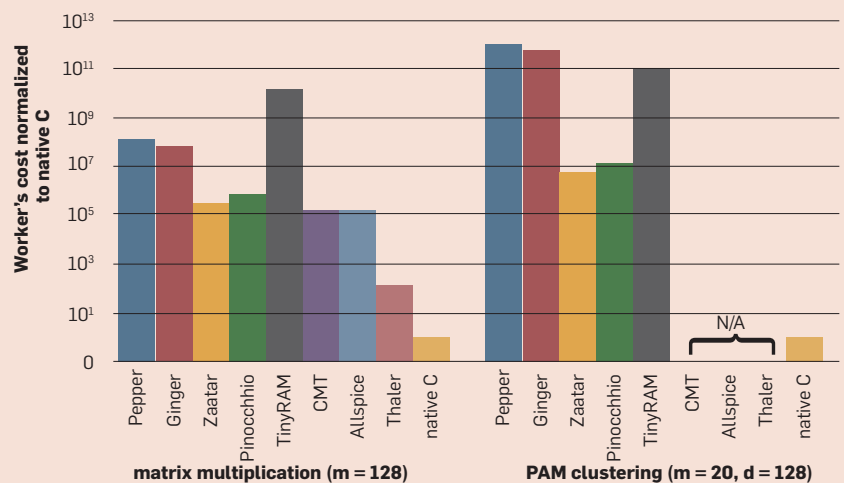
So that is the encouraging news. The sobering news, of course, is these systems are basically toys. Part of the rea-

Figure 4. Total costs and cross-over points (extrapolated), for 128×128 matrix multiplication.



The slope of each line is the per-instance cost (depicted in Figure 3); the y-intercepts are the setup costs and equal 0 for local, CMT, and Thaler. The cross-over point is the x-axis point at which a system's total cost line crosses its "local" line. The cross-over points for Zaatar and Pinocchio are in the thousands; the special-purpose approaches do far better but do not apply to all computations. Pinocchio's cross-over point could be improved by constant factors, using TinyRAM's optimized implementation.⁹ Although it has the worst cross-over point, TinyRAM has the best amortization regime, followed by Pinocchio and Zaatar (see text).

Figure 5. Prover overhead normalized to native execution cost for two computations. Prover overheads are generally enormous.



son we are willing to label them near-practical is painful experience with what the theory *used* to cost. (As a rough analogy, imagine a graduate student's delight in discovering hexadecimal machine code after years spent programming one-tape Turing machines.)

Still, these systems are arguably useful in some scenarios. In high-assurance contexts, we might be willing to pay a lot to know that a remotely deployed machine is executing correctly. In the streaming context, the verifier may not have space to compute locally, so we could use CMT¹⁸ to check the outputs are correct, in concert with Thaler's refinements⁴⁵ to make the prover truly inexpensive. Finally, data-parallel cloud computations (like MapReduce jobs) perfectly match the regimes in which the general-purpose schemes perform well: abundant CPU cycles for the prover and many instances of the same computation with different inputs.

Moreover, the gap separating the performance of the current research prototypes and plausible deployment in the cloud is a few orders of magnitude—which is certainly daunting, but, given the current pace of improvement, might be bridged in a few years.

More speculatively, if the machinery becomes truly low overhead, the effects will go far beyond verifying cloud computations: we will have new ways of building systems. In any situation in which one module performs a task for another, the delegating module will be able to check the answers. This could apply at the micro level (if the CPU could check the results of the GPU, this would expose hardware errors) and the macro level (distributed systems could be built under very different trust assumptions).

But even if none of this comes to pass, there are exciting intellectual currents here. Across computer systems, we are starting to see a new style of work: reducing sophisticated cryptography and other achievements of theoretical computer science to practice.^{28,35,38,49}

These developments are likely a product of our times: the preoccupation with strong security of various kinds, and the computers powerful enough to run previously “paper-only” algorithms. Whatever the cause, proof-based verifiable computation is an excellent example of this tendency: not only does it compose theoretical refinements with systems

techniques, it also raises research questions in other sub-disciplines of computer science. This cross-pollination is the best news of all.

Acknowledgments. We thank Sri-nath Setty, Justin Thaler, Riad Wahby, Alexis Gallagher, the anonymous *Communications* reviewers, Boaz Barak, William Blumberg, Oded Goldreich, Yuval Ishai and Guy Rothblum. C

References

- Arora, S., Lund, C., Motwani, R., Sudan, M. and Szegedy, M. Proof verification and the hardness of approximation problems. *JACM* 45, 3 (May 1998), 501–555. (Prelim. version FOCS 1992).
- Arora, S. and Safra, S. Probabilistic checking of proofs: A new characterization of NP. *JACM* 45, 1 (Jan. 1998), 70–122. (Prelim. version FOCS 1992).
- Babai, L. Trading group theory for randomness. In *Proceedings of STOC*, 1985.
- Babai, L., Fortnow, L., Levin, A. and Szegedy, M. Checking computations in polylogarithmic time. In *Proceedings of STOC*, 1991.
- Ben-Sasson, E. et al. Decentralized anonymous payments from Bitcoin. *IEEE Symposium on Security and Privacy*, 2014.
- Ben-Sasson, E., Chiesa, A., Genkin, D. and Tromer, E. Fast reductions from RAMs to delegatable succinct constraint satisfaction problems. In *Proceedings of ITCS*, Jan. 2013.
- Ben-Sasson, E., Chiesa, A., Genkin, D. and Tromer, E. SNARKs for C: Verifying program executions succinctly and in zero knowledge. In *Proceedings of CRYPTO*, Aug. 2013.
- Ben-Sasson, E., Chiesa, A., Tromer, E. and Virza, M. Scalable zero knowledge via cycles of elliptic curves. In *Proceedings of CRYPTO*, Aug. 2014.
- Ben-Sasson, E., Chiesa, A., Tromer, E. and Virza, M. Succinct non-interactive zero knowledge for a von Neumann architecture. *USENIX Security*, (Aug. 2014).
- Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M. and S. Vadhan, S. Robust PCPs of proximity, shorter PCPs and applications to coding. *SIAM J. on Comp.* 36, 4 (Dec. 2006), 889–974.
- Bitansky, N., Canetti, R., Chiesa, A. and Tromer, E. Recursive composition and bootstrapping for SNARKs and proof-carrying data. In *Proceedings of STOC*, June 2013.
- Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R. and Paneth, O. Succinct non-interactive arguments via linear interactive proofs. In *Proceedings of IACR TCC*, Mar. 2013.
- Brassard, G., Chaum, D. and Crépeau, C. Minimum disclosure proofs of knowledge. *J. Comp. and Sys. Sciences* 37, 2 (Oct. 1988), 156–189.
- Braun, B., Feldman, A.J., Ren, Z., Setty, S., Blumberg, A.J., and Walfish, M. Verifying computations with state. In *Proceedings of SOSP*, Nov. 2013.
- Canetti, R., Riva, B. and Rothblum, G. Practical delegation of computation using multiple servers. *ACM CCS*, 2011.
- Castro, M. and Liskov, B. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. on Comp. Sys.* 20, 4 (Nov. 2002), 398–461.
- Chung, K.M., Kalai, Y. and Vadhan, S. Improved delegation of computation using fully homomorphic encryption. In *Proceedings of CRYPTO*, 2010.
- Cormode, G., Mitzenmacher, M. and Thaler, J. Practical verifiable computation with streaming interactive proofs. In *Proceedings of ITCS*, 2012.
- Danezis, G., Fournet, C., Kohlweiss, M. and Parno, B. Pinocchio coin: Building zero coin from a succinct pairing-based proof system. In *Proceedings of the Workshop on Language Support for Privacy-enhancing Technologies*, Nov. 2013.
- Dinur, I. The PCP theorem by gap amplification. *JACM* 54, 3 (June 2007), 2:1–12:44.
- Gennaro, R., Gentry, C. and Parno, B. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Proceedings of CRYPTO*, 2010.
- Gennaro, R., Gentry, C. and Parno, B. and Raykova, M. Quadratic span programs and succinct NIZKs without PCPs. In *Proceedings of EUROCRYPT*, May 2013.
- Gentry, C. A fully homomorphic encryption scheme. PhD thesis, Stanford University, 2009.
- Goldreich, O. Probabilistic proof systems—A primer. *Foundations and Trends in Theoretical Computer Science* 3, 1 (2007), 1–91.
- Goldwasser, S., Kalai, Y.T. and Rothblum, G.N. Delegating computation: Interactive proofs for muggles. In *Proceedings of STOC*, May 2008.
- Goldwasser, S., Micali, S. and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM J. on Comp.* 18, 1 (1989), 186–208.
- Håstad, J. Some optimal inapproximability results. *JACM* 48, 4 (July 2001), 798–859. (Prelim. version STOC 1997).
- Huang, Y., Evans, D., Katz, J. and Malka, L. Faster secure two-party computation using garbled circuits. In *USENIX Security*, 2011.
- Ishai, Y., Kushilevitz, E., and Ostrovsky, R. Efficient arguments without short PCPs. In *Proceedings of the Conference on Computational Complexity (CCC)*, 2007.
- Kilian, J. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of STOC*, 1992.
- Kreuter, B., Shelat, A. and Shen, C.H. Billion-gate secure computation with malicious adversaries. *USENIX Security* (Aug. 2012).
- Lund, C., Fortnow, L., Karloff, H.J., and Nisan, N. Algebraic methods for interactive proof systems. *JACM* 39, 4 (1992), 859–888.
- Mahajan, P. et al. Depot: Cloud storage with minimal trust. *ACM Trans. on Comp. Sys.* 29, 4 (Dec. 2011).
- Malkhi, D. and Reiter, M. Byzantine quorum systems. *Distributed Computing* 11, 4 (Oct. 1998), 203–213. (Prelim. version *Proceedings of STOC* 1997).
- Narayan, A. and Haebleren, A. DJoin: Differentially private join queries over distributed databases. In *Proceedings of OSDI*, 2012.
- Parno, B., Gentry, C., Howell, J. and Raykova, M. Pinocchio: Nearly practical verifiable computation. *IEEE Symposium on Security and Privacy*, (May 2013).
- Parno, B., McCune, J.M. and Perrig, A. *Bootstrapping Trust in Modern Computers*. Springer, 2011.
- Popa, R.A., Redfield, C.M.S., Zeldovich, N. and Balakrishnan, H. CryptDB: Protecting confidentiality with encrypted query processing. In *Proceedings of SOSP*, 2011.
- Sadeghi, A.R., Schneider, T., and Winandy, M. Token-based cloud computing: Secure outsourcing of data and arbitrary computations with lower latency. In *Proceedings of TRUST*, 2010.
- Setty, S., Blumberg, A.J. and Walfish, M. Toward practical and unconditional verification of remote computations. In *Proceedings of HotOS*, May 2011.
- Setty, S., Braun, B., Vu, V., Blumberg, A.J., Parno, B. and Walfish, M. Resolving the conflict between generality and plausibility in verified computation. In *Proceedings of EuroSys*, Apr. 2013.
- Setty, S., McPherson, R., Blumberg, A.J., and Walfish, M. Making argument systems for outsourced computation practical (sometimes). In *Proceedings of NDSS*, 2012.
- Setty, S., Vu, V., Panpalia, N., Braun, B., Blumberg, A.J. and Walfish, M. Taking proof-based verified computation a few steps closer to practicality. *USENIX Security*, Aug. 2012.
- Sudan, M. Probabilistically checkable proofs. *Commun. ACM* 52, 3 (Mar. 2009), 76–84.
- Thaler, J. Time-optimal interactive proofs for circuit evaluation. In *Proceedings of CRYPTO*, Aug. 2013.
- Thaler, J., Roberts, M., Mitzenmacher, M. and Pfister, H. Verifiable computation with massively parallel interactive proofs. In *Proceedings of USENIX HotCloud Workshop*, (June 2012).
- Vu, V., Setty, S., Blumberg, A.J. and Walfish, M. A hybrid architecture for interactive verifiable computation. *IEEE Symposium on Security and Privacy*, (May 2013).
- Wahby, R.S., Setty, S., Ren, Z., Blumberg, A.J. and Walfish, M. Efficient RAM and control flow in verifiable outsourced computation. In *Proceedings of NDSS*, Feb. 2015.
- Wolinsky, D.I., Corrigan-Gibbs, H., Ford, B. and Johnson, A. Dissent in numbers: Making strong anonymity scale. In *Proceedings of OSDI*, 2012.

Michael Walfish (mwalfish@cs.nyu.edu) is an associate professor in the computer science department at New York University, New York City.

Andrew J. Blumberg (blumberg@math.utexas.edu) is an associate professor of mathematics at the University of Texas at Austin.

Copyright held by authors.

P. 86

**Technical
Perspective**
**The Equivalence
Problem for
Finite Automata**

By Thomas A. Henzinger
and Jean-François Raskin

P. 87

Hacking Nondeterminism with Induction and Coinduction

By Filippo Bonchi and Damien Pous



Watch the authors discuss
this work in this exclusive
Communications video.

Technical Perspective

The Equivalence Problem for Finite Automata

By Thomas A. Henzinger and Jean-François Raskin

FORMAL LANGUAGES AND automata are fundamental concepts in computer science. Pushdown automata form the theoretical basis for the parsing of programming languages. Finite automata provide natural data structures for manipulating infinite sets of values that are encoded by strings and generated by regular operations (concatenation, union, repetition). They provide elegant solutions in a wide variety of applications, including the design of sequential circuits, the modeling of protocols, natural language processing, and decision procedures for logical formalisms (remember the fundamental contributions by Rabin, Büchi, and many others).

Much of the power and elegance of automata comes from the natural ease with which they accommodate nondeterminism. The fundamental concept of nondeterminism—the ability of a computational engine to guess a path to a solution and then verify it—lies at the very heart of theoretical computer science. For finite automata, Rabin and Scott (1959) showed that nondeterminism does not add computational power, because every nondeterministic finite automaton (NFA) can be converted to an equivalent deterministic finite automaton (DFA) using the subset construction. However, since the subset construction may increase the number of automaton states exponentially, even simple problems about deterministic automata can become computationally difficult to solve if nondeterminism is involved.

One of the basic problems in automata theory is the equivalence problem: given two automata A and B , do they define the same language, that is, $L(A) = L(B)$. For DFA, the equivalence problem can be solved in linear time by the algorithm of Hopcroft and Karp (1971). For NFA, however, the minimization algorithm does not solve the equivalence problem, but

computes the stronger notion of bisimilarity between automata. Instead, the textbook algorithm for NFA equivalence checks language inclusion in both directions, which reduces to checking that both $\overline{L(A)} \cap L(B) = \emptyset$ and $L(A) \cap \overline{L(B)} = \emptyset$. The complementation steps are expensive for NFA: using the subset construction to determinize both input automata before complementing them causes, in the worst case, an exponential cost. Indeed, it is unlikely that there is a theoretically better solution, as the equivalence problem for NFA is PSpace-hard, which was shown by Meyer and Stockmeyer (1972).

As the equivalence problem is essential in many applications—from compilers to hardware and software verification—we need algorithms that avoid the worst-case complexity as often as possible. The exponential blow-up can be avoided in many cases by keeping the subset construction implicit. This can be done, for example, by using symbolic techniques such as binary decision diagrams for representing state sets. Filippo Bonchi and Damien Pous show us there is a better way.

The starting point of their solution is a merging of the Hopcroft-Karp DFA equivalence algorithm with subset constructions for the two input automata.

Much of the power and elegance of automata comes from the natural ease with which they accommodate nondeterminism.

This idea does not lead to an efficient algorithm per se, as it can be shown that the entire state spaces of the two subset constructions (or at least their reachable parts) may need to be explored to establish bisimilarity if the two input automata are equivalent. The contribution of Bonchi and Pous is to show that bisimilarity—that is, the existence of a bisimulation relation between states—can be proved without constructing the deterministic automata explicitly. They show that, instead, it suffices to compute a bisimulation up to congruence. It turns out that for computing such a bisimulation up to congruence, often only a small fraction of the subset spaces need to be explored. As you will see, the formalization of their idea is extremely elegant and leads to an algorithm that is efficient in practice.

The authors evaluate their algorithm on benchmarks. They explore how it relates to another class of promising recent approaches, called antichain methods, which also avoid explicit subset constructions for solving related problems, such as language inclusion for NFA over finite and infinite words and the solution of games played on graphs with imperfect information. In addition, they show how their construction can be improved by exploiting polynomial time computable simulation relations on NFA, an idea that was suggested by the antichain approach.

As this work vividly demonstrates, even classical, well-studied problems like NFA equivalence can still offer surprising research opportunities, and new ideas may lead to elegant algorithmic improvements of practical importance. 

Thomas A. Henzinger is president of the IST Austria (Institute of Science and Technology Austria).

Jean-François Raskin is a professor of computer science at the Université Libre de Bruxelles, Belgium.

Copyright held by authors.

Hacking Nondeterminism with Induction and Coinduction

By Filippo Bonchi and Damien Pous

Abstract

We introduce *bisimulation up to congruence* as a technique for proving language equivalence of nondeterministic finite automata. Exploiting this technique, we devise an optimization of the classic algorithm by Hopcroft and Karp.¹³ We compare our approach to the recently introduced antichain algorithms and we give concrete examples where we exponentially improve over antichains. Experimental results show significant improvements.

1. INTRODUCTION

Checking language equivalence of finite automata is a classic problem in computer science, with many applications in areas ranging from compilers to model checking.

Equivalence of deterministic finite automata (DFA) can be checked either via minimization¹² or through Hopcroft and Karp's algorithm,¹³ which exploits an instance of what is nowadays called a *coinduction proof principle*^{17, 20, 22}: two states are equivalent if and only if there exists a *bisimulation* relating them. In order to check the equivalence of two given states, Hopcroft and Karp's algorithm creates a relation containing them and tries to build a bisimulation by adding pairs of states to this relation: if it succeeds then the two states are equivalent, otherwise they are different.

On the one hand, minimization algorithms have the advantage of checking the equivalence of all the states at once, while Hopcroft and Karp's algorithm only checks a given pair of states. On the other hand, they have the disadvantage of needing the whole automata from the beginning, while Hopcroft and Karp's algorithm can be executed “on-the-fly,”⁸ on a lazy DFA whose transitions are computed on demand.

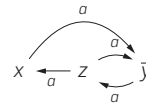
This difference is essential for our work and for other recently introduced algorithms based on *antichains*.^{1, 7, 25} Indeed, when starting from nondeterministic finite automata (NFA), determinization induces an exponential factor. In contrast, the algorithm we introduce in this work for checking equivalence of NFA (as well as those using antichains) usually does not build the whole deterministic automaton, but just a small part of it. We write “usually” because in few cases, the algorithm can still explore an exponential number of states.

Our algorithm is grounded on a simple observation on DFA obtained by determinizing an NFA: for all sets X and Y of states of the original NFA, the union (written $+$) of the language recognized by X (written $\llbracket X \rrbracket$) and the language recognized by Y ($\llbracket Y \rrbracket$) is equal to the language recognized by the union of X and Y ($\llbracket X + Y \rrbracket$). In symbols:

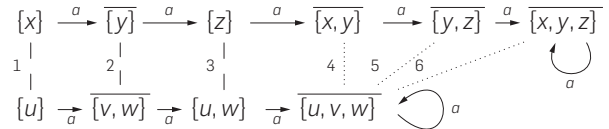
$$\llbracket X + Y \rrbracket = \llbracket X \rrbracket + \llbracket Y \rrbracket \quad (1)$$

This fact leads us to introduce a sound and complete proof technique for language equivalence, namely *bisimulation up to context*, that exploits both *induction* (on the operator $+$) and *coinduction*: if a bisimulation R relates the set of states X_1 with Y_1 and X_2 with Y_2 , then $\llbracket X_1 \rrbracket = \llbracket Y_1 \rrbracket$ and $\llbracket X_2 \rrbracket = \llbracket Y_2 \rrbracket$ and, by Equation (1), we can immediately conclude that $X_1 + X_2$ and $Y_1 + Y_2$ are language equivalent as well. Intuitively, bisimulations up to context are bisimulations which *do not need to relate* $X_1 + X_2$ with $Y_1 + Y_2$ when X_1 is already related with Y_1 and X_2 with Y_2 .

To illustrate this idea, let us check the equivalence of states x and u in the following NFA. (Final states are overlined, labeled edges represent transitions.)



The determinized automaton is depicted below.



Each state is a set of states of the NFA. Final states are overlined: they contain at least one final state of the NFA. The numbered lines show a relation which is a bisimulation containing x and u . Actually, this is the relation that is built by Hopcroft and Karp's algorithm (the numbers express the order in which pairs are added).

The dashed lines (numbered by 1, 2, 3) form a smaller relation which is not a bisimulation, but a bisimulation up to context: the equivalence of $\{x, y\}$ and $\{u, v, w\}$ is deduced from the fact that $\{x\}$ is related with $\{u\}$ and $\{y\}$ with $\{v, w\}$, without the need to further explore the automaton.

Bisimulations up-to, and in particular bisimulations up to context, have been introduced in the setting of concurrency theory^{17, 21} as a proof technique for bisimilarity of CCS or π -calculus processes. As far as we know, they have never been used for proving language equivalence of NFA.

Among these techniques one should also mention *bisimulation up to equivalence*, which, as we show in this paper, is implicitly used in Hopcroft and Karp's original

Extended Abstract, a full version of this paper is available in *Proceedings of POPL, 2013, ACM*.

algorithm. This technique can be explained by noting that not all bisimulations are equivalence relations: it might be the case that a bisimulation relates X with Y and Y with Z , but not X with Z . However, since $\llbracket X \rrbracket = \llbracket Y \rrbracket$ and $\llbracket Y \rrbracket = \llbracket Z \rrbracket$, we can immediately conclude that X and Z recognize the same language. Analogously to bisimulations up to context, a bisimulation up to equivalence does *not* need to relate X with Z when they are both related with some Y .

The techniques of up-to equivalence and up-to context can be combined, resulting in a powerful proof technique which we call *bisimulation up to congruence*. Our algorithm is in fact just an extension of Hopcroft and Karp's algorithm that attempts to build a bisimulation up to congruence instead of a bisimulation up to equivalence. An important property when using up to congruence is that we do not need to build the whole deterministic automata. For instance, in the above NFA, the algorithm stops after relating z with $u + w$ and does not build the remaining states. Despite their use of the up to equivalence, this is not the case with Hopcroft and Karp's algorithm, where all accessible subsets of the deterministic automata have to be visited at least once.

The ability of visiting only a small portion of the determinized automaton is also the key feature of the antichain algorithm²⁵ and its optimization exploiting similarity.^{1, 7} The two algorithms are designed to check *language inclusion* rather than equivalence and, for this reason, they do not exploit equational reasoning. As a consequence, the antichain algorithm usually needs to explore more states than ours. Moreover, we show how to integrate the optimization proposed in Abdulla et al.¹ and Doyen and Raskin⁷ in our setting, resulting in an even more efficient algorithm.

Outline

Section 2 recalls Hopcroft and Karp's algorithm for DFA, showing that it implicitly exploits bisimulation up to equivalence. Section 3 describes the novel algorithm, based on bisimulations up to congruence. We compare this algorithm with the antichain one in Section 4.

2. DETERMINISTIC AUTOMATA

A deterministic finite automaton (DFA) over the alphabet A is a triple (S, o, t) , where S is a finite set of states, $o: S \rightarrow 2$ is the output function, which determines if a state $x \in S$ is final ($o(x) = 1$) or not ($o(x) = 0$), and $t: S \rightarrow S^A$ is the transition function which returns, for each state x and for each letter $a \in A$, the next state $t_a(x)$. Any DFA induces a function $\llbracket \cdot \rrbracket$ mapping states to formal languages ($\mathcal{P}(A^*)$), defined by $\llbracket x \rrbracket(\epsilon) = o(x)$ for the empty word, and $\llbracket x \rrbracket(aw) = \llbracket t_a(x) \rrbracket(w)$ otherwise. For a state x , $\llbracket x \rrbracket$ is called the language accepted by x .

Throughout this paper, we consider a fixed automaton (S, o, t) and study the following problem: given two states x_1, x_2 in S , is it the case that they are language equivalent, that is, $\llbracket x_1 \rrbracket = \llbracket x_2 \rrbracket$? This problem generalizes the familiar problem of checking whether two automata accept the same language: just take the union of the two automata as the

automaton (S, o, t) , and determine whether their respective starting states are language equivalent.

2.1. Language equivalence via coinduction

We first define bisimulation. We make explicit the underlying notion of progression, which we need in the sequel.

DEFINITION 1 (PROGRESSION, BISIMULATION). *Given two relations $R, R' \subseteq S^2$ on states, R progresses to R' , denoted $R \rhd R'$, if whenever $x R y$ then*

1. $o(x) = o(y)$ and
2. *for all $a \in A$, $t_a(x) R' t_a(y)$.*

A bisimulation is a relation R such that $R \rhd R$.

As expected, bisimulation is a sound and complete proof technique for checking language equivalence of DFA:

PROPOSITION 1 (COINDUCTION). *Two states are language equivalent iff there exists a bisimulation that relates them.*

2.2. Naive algorithm

Figure 1 shows a naive version of Hopcroft and Karp's algorithm for checking language equivalence of the states x and y of a deterministic finite automaton (S, o, t) . Starting from x and y , the algorithm builds a relation R that, in case of success, is a bisimulation.

PROPOSITION 2. *For all $x, y \in S$, $x \sim y$ iff $\text{Naive}(x, y)$.*

PROOF. We first observe that if $\text{Naive}(x, y)$ returns true then the relation R that is built before arriving to step 4 is a bisimulation. Indeed, the following proposition is an invariant for the loop corresponding to step 3:

$$R \rhd R \cup \text{todo}$$

Since *todo* is empty at step 4, we have $R \rhd R$, that is, R is a bisimulation. By Proposition 1, $x \sim y$. On the other hand, $\text{Naive}(x, y)$ returns false as soon as it finds a word which is accepted by one state and not the other. $\square$

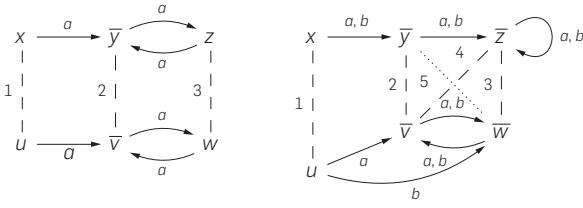
For example, consider the DFA with input alphabet $A = \{a\}$ in the left-hand side of Figure 2, and suppose we want to check that x and u are language equivalent.

Figure 1. Naive algorithm for checking the equivalence of states x and y of a DFA (S, o, t) . The code of $\text{HK}(x, y)$ is obtained by replacing the test in step 3.2 with $(x', y') \in e(R)$.

Naive(x, y)

```
(1)  $R$  is empty; todo is empty;
(2) insert  $(x, y)$  in todo;
(3) while todo is not empty do
  (3.1) extract  $(x', y')$  from todo;
  (3.2) if  $(x', y') \in R$  then continue;
  (3.3) if  $o(x') \neq o(y')$  then return false;
  (3.4) for all  $a \in A$ ,
    insert  $(t_a(x'), t_a(y'))$  in todo;
  (3.5) insert  $(x', y')$  in  $R$ ;
(4) return true;
```

Figure 2. Checking for DFA equivalence.



During the initialization, (x, u) is inserted in *todo*. At the first iteration, since $o(x) = 0 = o(u)$, (x, u) is inserted in R and (y, v) in *todo*. At the second iteration, since $o(y) = 1 = o(v)$, (y, v) is inserted in R and (z, w) in *todo*. At the third iteration, since $o(z) = 0 = o(w)$, (z, w) is inserted in R and (y, v) in *todo*. At the fourth iteration, since (y, v) is already in R , the algorithm does nothing. Since there are no more pairs to check in *todo*, the relation R is a bisimulation and the algorithm terminates returning true.

These iterations are concisely described by the numbered dashed lines in Figure 2. The line i means that the connected pair is inserted in R at iteration i . (In the sequel, when enumerating iterations, we ignore those where a pair from *todo* is already in R so that there is nothing to do.)

In the previous example, *todo* always contains at most one pair of states but, in general, it may contain several of them. We do not specify here how to choose the pair to extract in step 3.1; we discuss this point in Section 3.2.

2.3. Hopcroft and Karp's algorithm

The naive algorithm is quadratic: a new pair is added to R at each nontrivial iteration, and there are only n^2 such pairs, where $n = |S|$ is the number of states of the DFA. To make this algorithm (almost) linear, Hopcroft and Karp actually record a set of *equivalence classes* rather than a set of visited pairs. As a consequence, their algorithm may stop earlier it encounters a pair of states that is not already in R but belongs to its reflexive, symmetric, and transitive closure. For instance, in the right-hand side example from Figure 2, we can stop when we encounter the dotted pair (y, w) since these two states already belong to the same equivalence class according to the four previous pairs.

With this optimization, the produced relation R contains at most n pairs. Formally, ignoring the concrete data structure used to store equivalence classes, Hopcroft and Karp's algorithm consists in replacing step 3.2 in Figure 1 with

(3.2) if $(x', y') \in e(R)$ then continue;

where $e: \mathcal{P}(S^2) \rightarrow \mathcal{P}(S^2)$ is the function mapping each relation $R \subseteq S^2$ into its symmetric, reflexive, and transitive closure. We refer to this algorithm as HK.

2.4. Bisimulations up-to

We now show that the optimization used by Hopcroft and Karp corresponds to exploiting an “up-to technique.”

DEFINITION 2 (BISIMULATION UP-TO). Let $f: \mathcal{P}(S^2) \rightarrow \mathcal{P}(S^2)$ be a function on relations. A relation R is a bisimulation up to f if $R \mapsto f(R)$, i.e., if $x R y$, then

1. $o(x) = o(y)$ and
2. for all $a \in A$, $t_a(x) f(R) t_a(y)$.

With this definition, Hopcroft and Karp's algorithm just consists in trying to build a bisimulation up to e . To prove the correctness of the algorithm, it suffices to show that any bisimulation up to e is contained in a bisimulation. To this end, we have the notion of compatible function^{19,21}:

DEFINITION 3 (COMPATIBLE FUNCTION). A function $f: \mathcal{P}(S^2) \rightarrow \mathcal{P}(S^2)$ is compatible if it is monotone and it preserves progressions: for all $R, R' \subseteq S^2$,

$$R \mapsto R' \text{ entails } f(R) \mapsto f(R').$$

PROPOSITION 3. Let f be a compatible function. Any bisimulation up to f is contained in a bisimulation.

We could prove directly that e is a compatible function; we, however, take a detour to ease our correctness proof for the algorithm we propose in Section 3.

LEMMA 1. The following functions are compatible:

- id*: the identity function;
- $f \circ g$: the composition of compatible functions f and g ;
- $\cup F$: the pointwise union of an arbitrary family F of compatible functions: $\cup F(R) = \cup_{f \in F} f(R)$;
- f^ω : the (omega) iteration of a compatible function f , defined by $f^\omega = \cup_i f^i$, with $f^0 = \text{id}$ and $f^{i+1} = f \circ f^i$;
- r : the constant reflexive function: $r(_) = \{(x, x) \mid x \in S\}$;
- s : the converse function: $s(R) = \{(y, x) \mid x R y\}$;
- t : the squaring function: $t(R) = \{(x, z) \mid \exists y, x R y R z\}$.

Intuitively, given a relation R , $(s \cup \text{id})(R)$ is the symmetric closure of R , $(r \cup s \cup \text{id})(R)$ is its reflexive and symmetric closure, and $(r \cup s \cup t \cup \text{id})^\omega(R)$ is its symmetric, reflexive, and transitive closure: $e = (r \cup s \cup t \cup \text{id})^\omega$. Another way to understand this decomposition of e is to recall that $e(R)$ can be defined inductively by the following rules:

$$\frac{}{x e(R) x} r \quad \frac{x e(R) y}{y e(R) x} s \quad \frac{x e(R) y \quad y e(R) z}{x e(R) z} t \quad \frac{x R y}{x e(R) y} \text{id}$$

THEOREM 1. Any bisimulation up to e is contained in a bisimulation.

COROLLARY 1. For all $x, y \in S$, $x \sim y$ iff $\text{HK}(x, y)$.

PROOF. Same proof as for Proposition 2, by using the invariant $R \mapsto e(R) \cup \text{todo}$. We deduce that R is a bisimulation up to e after the loop. We conclude with Theorem 1 and Proposition 1. $\square$

Returning to the right-hand side example from Figure 2, Hopcroft and Karp's algorithm constructs the relation

$$R_{\text{HK}} = \{(x, u), (y, v), (z, w), (z, v)\}$$

which is not a bisimulation, but a bisimulation up to e : it contains the pair (x, u) , whose b -transitions lead to (y, w) , which is not in R_{HK} but in its equivalence closure, $e(R_{\text{HK}})$.

3. NONDETERMINISTIC AUTOMATA

We now move from DFA to nondeterministic automata (NFA). An NFA over the alphabet A is a triple (S, o, t) , where S is a finite set of states, $o: S \rightarrow 2$ is the output function, and $t: S \rightarrow \mathcal{P}(S)^A$ is the transition relation: it assigns to each state $x \in S$ and letter $a \in A$ a set of possible successors.

The *powerset construction* transforms any NFA (S, o, t) into the DFA $(\mathcal{P}(S), o^\sharp, t^\sharp)$ where $o^\sharp: \mathcal{P}(S) \rightarrow 2$ and $t^\sharp: \mathcal{P}(S) \rightarrow \mathcal{P}(S)^A$ are defined for all $X \in \mathcal{P}(S)$ and $a \in A$ as follows:

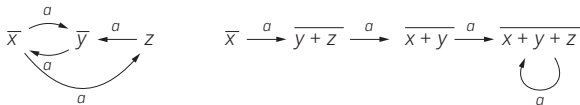
$$o^\sharp(X) = \begin{cases} o(x) & \text{if } X = \{x\} \text{ with } x \in S \\ 0 & \text{if } X = \emptyset \\ o^\sharp(X_1) + o^\sharp(X_2) & \text{if } X = X_1 + X_2 \end{cases}$$

$$t_a^\sharp(X) = \begin{cases} t_a(x) & \text{if } X = \{x\} \text{ with } x \in S \\ 0 & \text{if } X = \emptyset \\ t_a^\sharp(X_1) + t_a^\sharp(X_2) & \text{if } X = X_1 + X_2 \end{cases}$$

(Here we use the symbol “+” to denote both set-theoretic union and Boolean or; similarly, we use 0 to denote both the empty set and the Boolean “false.”) Observe that in $(\mathcal{P}(S), o^\sharp, t^\sharp)$, the states form a semilattice $(\mathcal{P}(S), +, 0)$, and $o^\sharp$ and $t^\sharp$ are, by definition, semilattices homomorphisms. These properties are fundamental for the up-to technique we are going to introduce. In order to stress the difference with generic DFA, which usually do not carry this structure, we use the following definition.

DEFINITION 4. A determinized NFA is a DFA $(\mathcal{P}(S), o^\sharp, t^\sharp)$ obtained via the powerset construction of some NFA (S, o, t) .

Hereafter, we use a new notation for representing states of determinized NFA: in place of the singleton $\{x\}$, we just write x and, in place of $\{x_1, \dots, x_n\}$, we write $x_1 + \dots + x_n$. Consider for instance the NFA (S, o, t) depicted below (left) and part of the determinized NFA $(\mathcal{P}(S), o^\sharp, t^\sharp)$ (right).



In the determinized NFA, x makes one single a -transition into $y + z$. This state is final: $o^\sharp(y + z) = o^\sharp(y) + o^\sharp(z) = o(y) + o(z) = 1 + 0 = 1$; it makes an a -transition into $t_a^\sharp(y + z) = t_a^\sharp(y) + t_a^\sharp(z) = t_a(y) + t_a(z) = x + y$.

Algorithms for NFA can be obtained by computing the determinized NFA on-the-fly⁸: starting from the algorithms for DFA (Figure 1), it suffices to work with sets of states, and to inline the powerset construction. The corresponding code is given in Figure 3. The naive algorithm (Naive) does not use any up to technique, Hopcroft and Karp’s algorithm (HK) reasons up to equivalence in step 3.2.

3.1. Bisimulation up to congruence

The semilattice structure $(\mathcal{P}(S), +, 0)$ carried by determinized NFA makes it possible to introduce a new up-to technique, which is not available with plain DFA: *up to congruence*. This technique relies on the following function.

DEFINITION 5 (CONGRUENCE CLOSURE). Let $u: \mathcal{P}(\mathcal{P}(S)^2) \rightarrow \mathcal{P}(\mathcal{P}(S)^2)$ be the function on relations on sets of states defined for all $R \subseteq \mathcal{P}(S)^2$ as:

$$u(R) = \{(X_1 + X_2, Y_1 + Y_2) \mid X_1 R Y_1 \text{ and } X_2 R Y_2\}$$

The function $c = (r \cup s \cup t \cup u \cup \text{id})^\omega$ is called the congruence closure function.

Intuitively, $c(R)$ is the smallest equivalence relation which is closed with respect to $+$ and which includes R . It could alternatively be defined inductively using the rules r , s , t , and id from the previous section, and the following one:

$$\frac{X_1 c(R) Y_1 \quad X_2 c(R) Y_2}{X_1 + X_2 c(R) Y_1 + Y_2} u$$

DEFINITION 6 (BISIMULATION UP TO CONGRUENCE). A bisimulation up to congruence for an NFA (S, o, t) is a relation $R \subseteq \mathcal{P}(S)^2$, such that whenever $X R Y$ then

1. $o^\sharp(X) = o^\sharp(Y)$ and
2. for all $a \in A$, $t_a^\sharp(X) c(R) t_a^\sharp(Y)$.

LEMMA 2. The function u is compatible.

THEOREM 2. Any bisimulation up to congruence is contained in a bisimulation.

We already gave in the Introduction section an example of *bisimulation up to context*, which is a particular case of bisimulation up to congruence (up to context means up to $(r \cup u \cup \text{id})^\omega$, without closing under s and t).

Figure 4 shows a more involved example illustrating the use of all ingredients of the congruence closure function (c). The relation R expressed by the dashed numbered lines (formally $R = \{(x, u), (y + z, u)\}$) is

Figure 3. On-the-fly naive algorithm, for checking the equivalence of sets of states X and Y of an NFA (S, o, t) . $\text{HK}(X, Y)$ is obtained by replacing the test in step 3.2 with $(X', Y') \in e(R)$, and $\text{HKC}(X, Y)$ is obtained by replacing it with $(X', Y') \in c(R \cup \text{todo})$.

Naive (X, Y)

- (1) R is empty; *todo* is empty;
- (2) insert (X, Y) in *todo*;
- (3) while *todo* is not empty do
 - (3.1) extract (X', Y') from *todo*;
 - (3.2) if $(X', Y') \in R$ then continue;
 - (3.3) if $o^\sharp(X') \neq o^\sharp(Y')$ then return false;
 - (3.4) for all $a \in A$,
 - insert $(t_a^\sharp(X'), t_a^\sharp(Y'))$ in *todo*;
 - (3.5) insert (X', Y') in R ;
- (4) return true;

neither a bisimulation nor a bisimulation up to e since $y + z \xrightarrow{a} x + y$ and $u \xrightarrow{a} u$, but $(x + y, u) \notin e(R)$. However, R is a bisimulation up to congruence. Indeed, we have $(x + y, u) \in c(R)$:

$$\begin{array}{ll} x + y \ c(R) \ u + y & ((x, u) \in R) \\ c(R) \ y + z + y & ((y + z, u) \in R) \\ = y + z & \\ c(R) \ u & ((y + z, u) \in R) \end{array}$$

In contrast, we need four pairs to get a bisimulation up to equivalence containing (x, u) : this is the relation depicted with both dashed and dotted lines in Figure 4.

Note that we can deduce many other equations from R ; in fact, $c(R)$ defines the following partition of sets of states: $\{0\}$, $\{y\}$, $\{z\}$, $\{x, u, x + y, x + z\}$, and the 9 remaining subsets.

3.2. Optimized algorithm for NFA

The optimized algorithm, called HKC in the sequel, relies on up to congruence: step 3.2 from Figure 3 becomes

(3.2) **if** $(X', Y') \in c(R \cup \text{todo})$ **then** **continue**;

Observe that we use $c(R \cup \text{todo})$ rather than $c(R)$: this allows us to skip more pairs, and this is safe since all pairs in *todo* will eventually be processed.

COROLLARY 2. For all $X, Y \in \mathcal{P}(S)$, $X \sim Y$ iff $\text{HKC}(X, Y)$.

PROOF. Same proof as for Proposition 2, by using the invariant $R \mapsto c(R \cup \text{todo})$ for the loop. We deduce that R is a bisimulation up to congruence after the loop. We conclude with Theorem 2 and Proposition 1. $\square$

The most important point about these three algorithms is that they compute the states of the determinized NFA lazily. This means that only *accessible* states need to be computed, which is of practical importance since the determinized NFA can be exponentially large. In case of a negative answer, the three algorithms stop even before all accessible states have been explored; otherwise, if a bisimulation (possibly up-to) is found, it depends on the algorithm:

- With **Naive**, all accessible states need to be visited, by definition of bisimulation.
- With **HK**, the only case where some accessible states can be avoided is when a pair (X, X) is encountered:

the algorithm skips this pair so that the successors of X are not necessarily computed (this situation never happens when starting with disjoint automata). In the other cases where a pair (X, Y) is skipped, X and Y are necessarily already related with some other states in R , so that their successors will eventually be explored.

- With **HKC**, accessible states are often skipped. For a simple example, let us execute HKC on the NFA from Figure 4. After two iterations, $R = \{(x, u), (y + z, u)\}$. Since $x + y \ c(R) \ u$, the algorithm stops without building the states $x + y$ and $x + y + z$. Similarly, in the example from the Introduction section, HKC does not construct the four states corresponding to pairs 4, 5, and 6.

This ability of HKC to ignore parts of the determinized NFA can bring an exponential speedup. For an example, consider the family of NFA in Figure 5, where n is an arbitrary natural number. Taken together, the states x and y are equivalent to z : they recognize the language $(a + b)^*(a + b)^{n+1}$. Alone, x recognizes the language $(a + b)^*a(a + b)^n$, which is known for having a minimal DFA with 2^n states.

Therefore, checking $x + y \sim z$ via minimization (as in Hopcroft¹²) requires exponential time, and the same holds for **Naive** and **HK** since all accessible states must be visited. This is not the case with **HKC**, which requires only polynomial time in this example. Indeed, $\text{HKC}(x + y, z)$ builds the relation

$$\begin{aligned} R' = & \{(x + y, z)\} \\ & \cup \{(x + Y_i + y_{i+1}, Z_{i+1}) \mid i < n\} \\ & \cup \{(x + Y_i + x_{i+1}, Z_{i+1}) \mid i < n\}, \end{aligned}$$

where $Y_i = y + y_1 + \dots + y_i$ and $Z_i = z + z_1 + \dots + z_i$. R' only contains $2n + 1$ pairs and is a bisimulation up to congruence. To see this, consider the pair $(x + y + x_1 + y_2, Z_2)$ obtained from $(x + y, z)$ after reading the word ba . Although this pair does not belong to R' , it belongs to its congruence closure:

$$\begin{array}{ll} x + y + x_1 + y_2 \ c(R') \ Z_1 + y_2 & (x + y + x_1 \ R' \ Z_1) \\ c(R') \ x + y + y_1 + y_2 & (x + y + y_1 \ R' \ Z_1) \\ c(R') \ Z_2 & (x + y + y_1 + y_2 \ R' \ Z_2) \end{array}$$

REMARK 1. In the above derivation, the use of transitivity is crucial: R' is a bisimulation up to congruence, but not a bisimulation up to context. In fact, there exists no bisimulation up to context of linear size proving $x + y \sim z$.

Figure 4. A bisimulation up to congruence.

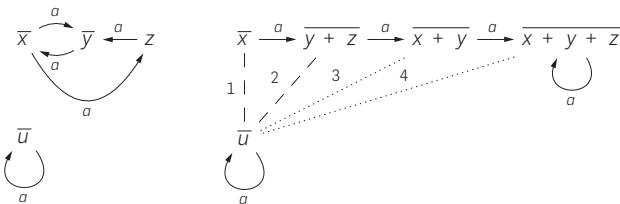
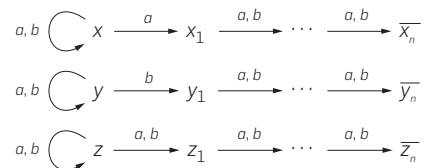


Figure 5. Family of examples where HKC exponentially improves over AC and HK; we have $x + y \sim z$.



We now discuss the exploration strategy, that is, how to choose the pair to extract from the set *todo* in step 3.1. When looking for a counterexample, such a strategy has a large influence: a good heuristic can help in reaching it directly, while a bad one might lead to explore exponentially many pairs first. In contrast, the strategy does not impact much looking for an equivalence proof (when the algorithm eventually returns true). Actually, one can prove that the number of steps performed by Naive and HK in such a case does not depend on the strategy. This is not the case with HKC: the strategy can induce some differences. However, we experimentally observed that breadth-first and depth-first strategies usually behave similarly on random automata. This behavior is due to the fact that we check congruence w.r.t. $R \cup \text{todo}$ rather than just R (step 3.2): with this optimization, the example above is handled in polynomial time whatever the chosen strategy. In contrast, without this small optimization, it requires exponential time with a depth-first strategy.

3.3. Computing the congruence closure

For the optimized algorithm to be effective, we need a way to check whether some pairs belong to the congruence closure of a given relation (step 3.2). We present a simple solution based on set rewriting; the key idea is to look at each pair (X, Y) in a relation R as a pair of rewriting rules:

$$X \rightarrow X + Y \qquad Y \rightarrow X + Y,$$

which can be used to compute normal forms for sets of states. Indeed, by idempotence, $XR Y$ entails $Xc(R)X + Y$.

DEFINITION 7. Let $R \subseteq \mathcal{P}(S)^2$ be a relation on sets of states. We define $\sim_R \subseteq \mathcal{P}(S)^2$ as the smallest irreflexive relation that satisfies the following rules:

$$\frac{XRY}{X \rightsquigarrow_R X+Y} \quad \frac{XRY}{Y \rightsquigarrow_R X+Y} \quad \frac{Z \rightsquigarrow_R Z'}{U+Z \rightsquigarrow_R U+Z'}$$

LEMMA 3. *For all relations R , $\leadsto_R$ is confluent and normalizing.*

In the sequel, we denote by $X \downarrow_R$ the normal form of a set X w.r.t. $\sim_R$. Intuitively, the normal form of a set is the largest set of its equivalence class. Recalling the example from Figure 4, the common normal form of $x + y$ and u can be computed as follows (R is the relation $\{(x, u), (y + z, u)\}$):

$$\begin{array}{ccccccc}
 x+y & & & & & & u \\
 \searrow & & & & & & \swarrow \\
 & x+y+u & & & x+u & & \\
 & \searrow & & & \swarrow & & \\
 & & x+y+z+u & & & &
 \end{array}$$

THEOREM 3. *For all relations R , and for all $X, Y \in \mathcal{P}(S)$, we have $X \downarrow_R = Y \downarrow_R$ iff $(X, Y) \in c(R)$.*

We actually have $X \downarrow_R = Y \downarrow_R$ iff $X \subseteq Y \downarrow_R$ and $Y \subseteq X \downarrow_R$, so that the normal forms of X and Y do not necessarily need to be fully computed in practice. Still, the worst-case complexity of this subalgorithm is quadratic in the size of the relation R

(assuming we count the number of operations on sets: unions and inclusion tests).

Note that many algorithms were proposed in the literature to compute the congruence closure of a relation (see, e.g., Nelson and Oppen,¹⁸ Shostak,²³ and Bachmair et al.²). However, they usually consider uninterpreted symbols or associative and commutative symbols, but not associative, commutative, and idempotent symbols, which is what we need here.

3.4. Using HKC for checking language inclusion

For NFA, language inclusion can be reduced to language equivalence: the semantics function $[-]$ is a semilattice homomorphism, so that for all sets of states X, Y , $[X + Y] = [Y]$ iff $[X] + [Y] = [Y]$ iff $[X] \subseteq [Y]$. Therefore, it suffices to run $\text{HKC}(X + Y, Y)$ to check the inclusion $[X] \subseteq [Y]$.

In such a situation, all pairs that are eventually manipulated by HKC have the shape $(X' + Y', Y')$ for some sets X', Y' . Step 3. 2 of HKC can thus be simplified. First, the pairs in the current relation only have to be used to rewrite from right to left. Second, the following lemma shows that we do not necessarily need to compute normal forms:

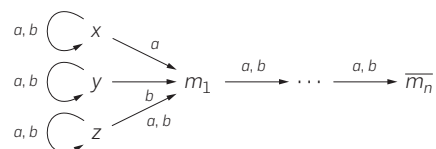
LEMMA 4. For all sets X, Y and for all relations R , we have $X + Y \subseteq c(R) Y$ iff $X \subseteq Y \downarrow_R$.

At this point, the reader might wonder whether checking the two inclusions separately is more convenient than checking the equivalence directly. This is not the case: checking the equivalence directly actually allows one to skip some pairs that cannot be skipped when reasoning by double inclusion. As an example, consider the DFA on the right of Figure 2. The relation computed by $\text{HKC}(x, u)$ contains only four pairs (because the fifth one follows from transitivity). Instead, the relations built by $\text{HKC}(x, x + u)$ and $\text{HKC}(u + x, u)$ would both contain five pairs: transitivity cannot be used since our relations are now oriented (from $y \leq v$, $z \leq v$, and $z \leq w$, we cannot deduce $y \leq w$). Figure 5 shows another example, where we get an exponential factor by checking the equivalence directly rather than through the two inclusions: transitivity, which is crucial to keep the relation computed by $\text{HKC}(x + y, z)$ small (see Remark 1), cannot be used when checking the two inclusions separately.

In a sense, the behavior of the coinduction proof method here is similar to that of standard proofs by induction, where one often has to strengthen the induction predicate to get a (nicer) proof.

3.5. Exploiting similarity

Looking at the example in Figure 5, a natural idea would be to first quotient the automaton by graph isomorphism. By doing so, one would merge the states x_i, y_i, z_i , and one would obtain the following automaton, for which checking $x + y \sim z$ is much easier.



As shown in Abdulla et al.¹ and Doyen and Raskin⁷ for antichain algorithms, one can do better, by exploiting any preorder contained in language inclusion. Hereafter, we show how this idea can be embedded in HKC, resulting in an even stronger algorithm.

For the sake of clarity, we fix the preorder to be *similarity*,¹⁷ which can be computed in quadratic time.¹⁰

DEFINITION 8 (SIMILARITY). *Similarity is the largest relation on states $\preceq \subseteq S^2$ such that $x \preceq y$ entails:*

1. $o(x) \leq o(y)$ and
2. for all $a \in A, x' \in S$ such that $x \xrightarrow{a} x'$, there exists some y' such that $y \xrightarrow{a} y'$ and $x' \preceq y'$.

To exploit similarity pairs in HKC, it suffices to notice that for any similarity pair $x \preceq y$, we have $x + y \sim y$. Let $\overline{\preceq}$ denote the relation $\{(x + y, y) \mid x \preceq y\}$, let r' denote the constant-to- $\overline{\preceq}$ function, and let $c' = (r' \cup s \cup t \cup u \cup id)^o$. Accordingly, we call HKC' the algorithm obtained from HKC (Figure 3) by replacing $(X, Y) \in c(R \cup todo)$ with $(X, Y) \in c'(R \cup todo)$ in step 3.2. The latter test can be reduced to rewriting thanks to Theorem 3 and the following lemma.

LEMMA 5. *For all relations R , $c'(R) = c(R \cup \overline{\preceq})$.*

THEOREM 4. *Any bisimulation up to c' is contained in a bisimulation.*

COROLLARY 3. *For all sets X, Y , $X \sim Y$ iff $HKC'(X, Y)$.*

4. ANTICHAIN ALGORITHMS

Even though the problem of deciding NFA equivalence is PSPACE-complete,¹⁶ neither HKC nor HKC' are in PSPACE: both of them keep track of the states they explored in the determinized NFA, and there can be exponentially many such states. This also holds for HK and for the more recent antichain algorithm²⁵ (called AC in the following) and its optimization (AC') exploiting similarity.^{1,7}

The latter algorithms can be explained in terms of coinductive proof techniques: we establish in Bonchi and Pous⁴ that they actually construct bisimulations up to context, that is, bisimulations up to congruence for which one does not exploit *symmetry* and *transitivity*.

Theoretical comparison. We compared the various algorithms in details in Bonchi and Pous.⁴ Their relationship is summarized in Figure 6, where an arrow $X \rightarrow Y$ means that

(a) Y can explore exponentially fewer states than X and (b) Y can mimic X , that is, the coinductive proof technique underlying Y is at least as powerful as the one of X .

In the general case, AC needs to explore much more states than HKC: the use of transitivity, which is missing in AC, allows HKC to drastically prune the exploration. For instance, to check $x + y \sim z$ in Figure 5, HKC only needs a linear number of states (see Remark 1), while AC needs exponentially many states. In contrast, in the special case where one checks for the inclusion of disjoint automata, HKC and AC exhibit the same behavior. Indeed, HKC cannot make use of transitivity in such a situation, as explained in Section 3.4. Things change when comparing HKC' and AC': even for checking inclusion of disjoint automata, AC' cannot always mimic HKC': the use of similarity tends to virtually merge states, so that HKC' can use the up-to transitivity technique which AC' lack.

Experimental comparison. The theoretical relationships drawn in Figure 6 are substantially confirmed by an empirical evaluation of the performance of the algorithms. Here, we only give a brief overview; see Bonchi and Pous⁴ for a complete description of those experiments.

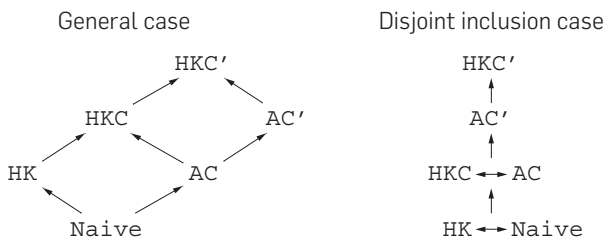
We compared our OCaml implementation⁴ for HK, HKC, and HKC', and the libvata C++ library¹⁴ for AC and AC'. We use a breadth-first exploration strategy: we represent the set *todo* from Figure 3 as a FIFO queue. As mentioned at the end of Section 3.2, considering a depth-first strategy here does not alter the behavior of HKC in a noticeable way.

We performed experiments using both random automata and a set of automata arising from model-checking problems.

- *Random automata.* We used Tabakov and Vardi's model²⁴ to generate 1000 random NFA with two letters and a given number of states. We executed all algorithms on these NFA, and we measured the number of processed pairs, that is, the number of required iterations (like HKC, AC is a loop inside which pairs are processed). We observe that HKC improves over AC by one order of magnitude, and AC improves over HK by two orders of magnitude. Using up-to similarity (HKC' and AC') does not improve much; in fact, similarity is almost the identity relation on such random automata. The corresponding distributions for HK, HKC, and AC are plotted in Figure 7, for automata with 100 states. Note that while HKC only improves by one order of magnitude over AC when considering the average case, it improves by several orders of magnitude when considering the worst cases.

- *Model-checking automata.* Abdulla et al.^{1,7} used automata sequences arising from regular model-checking experiments⁵ to compare their algorithm (AC') against AC. We reused these sequences to test HKC' against AC' in a concrete scenario. For all those sequences, we checked the inclusions of all consecutive pairs, in both directions. The timings are given in Table 1, where we report the median values (50%), the last deciles (90%), the last percentiles (99%), and

Figure 6. Relationships among the algorithms.



the maximum values (100%). We distinguish between the experiments for which a counterexample was found, and those for which the inclusion did hold. For HKC' and AC', we display the time required to compute similarity on a separate line: this preliminary step is shared by the two algorithms. As expected, HKC and AC roughly behave the same: we test inclusions of disjoint automata. HKC' is however quite faster than AC': up-to transitivity can be exploited, thanks to similarity pairs. Also note that over the 546 positive answers, 368 are obtained immediately by similarity.

5. CONCLUSION

Our implementation of HKC is available online,⁴ together with proofs mechanized in the Coq proof assistant and an interactive applet making it possible to test the presented algorithms online, on user-provided examples.

Several notions analogous to bisimulations up to congruence can be found in the literature. For instance, *self-bisimulations*^{6, 11} have been used to obtain decidability and complexity results about context-free processes. The main difference with bisimulation up to congruence is that self-bisimulations are proof techniques for bisimilarity rather than language equivalence. Other approaches that are independent from the equivalence (like bisimilarity or language) are shown in Lenisa,¹⁵ Bartels,³ and Pous.¹⁹ These papers

propose very general frameworks into which our up to congruence technique fits as a very special case. However, to our knowledge, bisimulation up to congruence has never been proposed before as a technique for proving language equivalence of NFA.

We conclude with directions for future work.

Complexity. The presented algorithms, as well as those based on antichains, have exponential complexity in the worst case while they behave rather well in practice. For instance, in Figure 7, one can notice that over a thousand random automata, very few require to explore a large amount of pairs. This suggests that an accurate analysis of the average complexity might be promising. An inherent problem comes from the difficulty to characterize the average shape of determinized NFA.²⁴ To avoid this problem, with HKC, we could try to focus on the properties of congruence relations. For instance, given a number of states, how long can be a sequence of (incrementally independent) pairs of sets of states whose congruence closure collapses into the full relation? (This number is an upper-bound for the size of the relations produced by HKC.) One can find ad hoc examples where this number is exponential, but we suspect it to be rather small in average.

Model checking. The experiments summarized in Table 1 show the efficiency of our approach for regular model checking using automata on finite words. As in the case of antichains, our approach extends to automata on finite trees. We plan to implement such a generalization and link it with tools performing regular tree model-checking.

In order to face other model-checking problems, it would be useful to extend up-to techniques to automata on infinite words, or trees. Unfortunately, the determinization of these automata (the so-called Safra's construction) does not seem suitable for exploiting neither antichains nor up to congruence. However, for some problems like LTL realizability⁹ that can be solved without prior determinization (the so-called Safraless approaches), antichains have been crucial in obtaining efficient procedures. We leave as future work to explore whether up-to techniques could further improve such procedures.

Acknowledgments

This work was partially funded by the PiCoq (ANR-10-BLAN-0305) and PACE (ANR-12IS02001) projects. 

Figure 7. Distributions of the number of processed pairs, for 1000 experiments with random NFA.

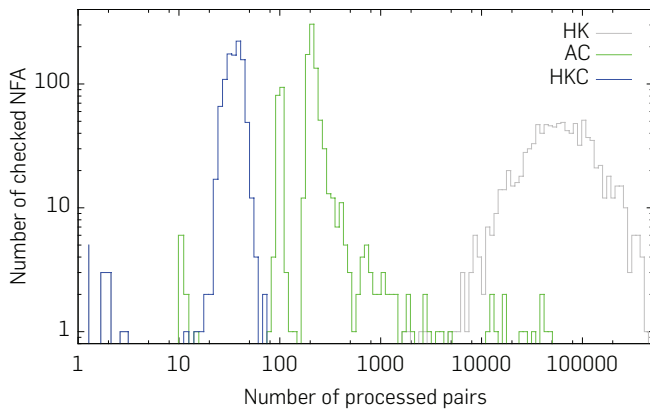


Table 1. Timings, in seconds, for language inclusion of disjoint NFA generated from model checking.

Algorithm	Inclusions (546 pairs)				Counterexamples (518 pairs)			
	50%	90%	99%	100%	50%	90%	99%	100%
AC	0.036	0.860	4.981	5.084	0.009	0.094	1.412	2.887
HKC	0.049	0.798	6.494	6.762	0.000	0.014	0.916	2.685
sim_time	0.039	0.185	0.574	0.618	0.038	0.193	0.577	0.593
AC'—sim_time	0.013	0.167	1.326	1.480	0.012	0.107	1.047	1.134
HKC'—sim_time	0.000	0.034	0.224	0.345	0.001	0.005	0.025	0.383

References

1. Abdulla, P.A., Chen, Y.F., Holik, L., Mayr, R., and Vojnar, T. When simulation meets antichains. In *TACAS*, J. Esparza and R. Majumdar, eds. Volume 6015 of *Lecture Notes in Computer Science* (2010). Springer, 158–174.
2. Bachmair, L., Ramakrishnan, I.V., Tiwari, A., and Vigneron, L. Congruence closure modulo associativity and commutativity. In *FroCoS*, H. Kirchner and C. Ringeissen, eds. Volume 1794 of *Lecture Notes in Computer Science* (2000). Springer, 245–259.
3. Bartels, F. Generalised coinduction. *Math. Struct. Comp. Sci.* 13, 2 (2003), 321–348.
4. Bonchi, F. and Pous, D. Extended version of this abstract, with omitted proofs, and web appendix for this work. <http://hal.linria.fr/hal-00639716/> and <http://perso.ens-lyon.fr/damien.pous/hknt>, 2012.
5. Bouajjani, A., Habermehl, P., and Vojnar, T. Abstract regular model checking. In *CAV*, R. Alur and D. Peled, eds. Volume 3114 of *Lecture Notes in Computer Science* (2004). Springer.
6. Caucal, D. Graphes canoniques de graphes algébriques. *ITA* 24 (1990), 339–352.
7. Doyen, L. and Raskin, J.F. Antichain algorithms for finite automata. In *TACAS*, J. Esparza and R. Majumdar, eds. Volume 6015 of *Lecture Notes in Computer Science* (2010). Springer.
8. Fernandez, J.C., Mounier, L., Jard, C., and Jéron, T. On-the-fly verification of finite transition systems. *Formal Meth. Syst. Design* 1, 2/3 (1992), 251–273.
9. Filiot, E., Jin, N., and Raskin, J.F. An antichain algorithm for LTL realizability. In *CAV*, A. Bouajjani and O. Maler, eds. Volume 5643 of *Lecture Notes in Computer Science* (2009). Springer, 263–277.
10. Henzinger, M.R., Henzinger, T.A., and Kopke, P.W. Computing simulations on finite and infinite graphs. In *Proceedings of 36th Annual Symposium on Foundations of Computer Science* (Milwaukee, WI, October 23–25, 1995). IEEE Computer Society Press.
11. Hirshfeld, Y., Jerrum, M., and Møller, F. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *TCS* 158, 1&2 (1996), 143–159.
12. Hopcroft, J.E. An $n \log n$ algorithm for minimizing in a finite automaton. In *International Symposium of Theory of Machines and Computations*. Academic Press, 1971, 189–196.
13. Hopcroft, J.E. and Karp, R.M. A linear algorithm for testing equivalence of finite automata. Technical Report 114. Cornell University, December 1971.
14. Lengál, O., Simáček, J., and Vojnar, T. Vata: A library for efficient manipulation of non-deterministic tree automata. In *TACAS*, C. Flanagan and B. König, eds. Volume 7214 of *Lecture Notes in Computer Science* (2012). Springer, 79–94.
15. Lenisa, M. From set-theoretic coinduction to coalgebraic coinduction: Some results, some problems. *ENTCS* 19 (1999), 2–22.
16. Meyer, A. and Stockmeyer, L.J. Word problems requiring exponential time. In *STOC*. ACM, 1973, 1–9.
17. Milner, R. *Communication and Concurrency*. Prentice Hall, 1989.
18. Nelson, G. and Oppen, D.C. Fast decision procedures based on congruence closure. *J. ACM* 27, 2 (1980), 356–364.
19. Pous, D. Complete lattices and up-to techniques. In *APLAS*, Z. Shao, ed. Volume 4807 of *Lecture Notes in Computer Science* (2007). Springer, 351–366.
20. Rutten, J. Automata and coinduction (an exercise in coalgebra). In *CONCUR*, D. Sangiorgi and R. de Simone, eds. Volume 1466 of *Lecture Notes in Computer Science* (1998). Springer, 194–218.
21. Sangiorgi, D. On the bisimulation proof method. *Math. Struct. Comp. Sci.* 8 (1998), 447–479.
22. Sangiorgi, D. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2011.
23. Shostak, R.E. Deciding combinations of theories. *J. ACM* 31, 1 (1984), 1–12.
24. Tabakov, D. and Vardi, M. Experimental evaluation of classical automata constructions. In *LPAR*, G. Sutcliffe and A. Voronkov, eds. Volume 3835 of *Lecture Notes in Computer Science* (2005). Springer, 396–411.
25. Wulf, M.D., Doyen, L., Henzinger, T.A., and Raskin, J.F. Antichains: A new algorithm for checking universality of finite automata. In *CAV*, T. Ball and R.B. Jones, eds. Volume 4144 of *Lecture Notes in Computer Science* (2006). Springer, 17–30.

Filippo Bonchi and Damien Pous

([filippo.bonchi, damien.pous]@ens-lyon.fr),
CNRS, ENS Lyon, LIP, Université de Lyon,
UMR 5668, France.



Watch the authors discuss
this work in this exclusive
Communications video.

© 2015 ACM 0001-0782/15/02 \$15.00

World-Renowned Journals from ACM

ACM publishes over 50 magazines and journals that cover an array of established as well as emerging areas of the computing field. IT professionals worldwide depend on ACM's publications to keep them abreast of the latest technological developments and industry news in a timely, comprehensive manner of the highest quality and integrity. For a complete listing of ACM's leading magazines & journals, including our renowned Transaction Series, please visit the ACM publications homepage: www.acm.org/pubs.

ACM Transactions on Interactive Intelligent Systems



ACM Transactions on Interactive Intelligent Systems (TIIS). This quarterly journal publishes papers on research encompassing the design, realization, or evaluation of interactive systems incorporating some form of machine intelligence.

ACM Transactions on Computation Theory



ACM Transactions on Computation Theory (ToCT). This quarterly peer-reviewed journal has an emphasis on computational complexity, foundations of cryptography and other computation-based topics in theoretical computer science.

PLEASE CONTACT ACM MEMBER SERVICES TO PLACE AN ORDER
Phone: 1.800.342.6626 (U.S. and Canada)
+1.212.626.0500 (Global)
Fax: +1.212.944.1318
(Hours: 8:30am–4:30pm, Eastern Time)
Email: acmhelp@acm.org
Mail: ACM Member Services
General Post Office
PO Box 30777
New York, NY 10087-0777 USA



Association for
Computing Machinery

Advancing Computing as a Science & Profession

www.acm.org/pubs

Are you looking for your next IT job?

Do you need Career Advice?

The **ACM Career & Job Center** offers ACM members a host of career-enhancing benefits:

- A **highly targeted focus** on job opportunities in the computing industry
- **Access to hundreds** of industry job postings
- Resume posting **keeping you connected** to the employment market while letting you maintain full control over your confidential information
- **Job Alert system** that notifies you of new opportunities matching your criteria
- **Career coaching** and guidance available from trained experts dedicated to your success
- **Free access** to a content library of the best career articles compiled from hundreds of sources, and much more!



Visit **ACM's Career & Job Center** at:
<http://jobs.acm.org>



Association for
Computing Machinery

Advancing Computing as a Science & Profession

The **ACM Career & Job Center** is the perfect place to begin searching for your next employment opportunity!

Visit today at **<http://jobs.acm.org>**

California Institute of Technology (Caltech)

Tenure-track faculty position

The Computing and Mathematical Sciences (CMS) Department at Caltech invites applications for a tenure-track faculty position. Our department is a unique environment where innovative, interdisciplinary, and foundational research is conducted in a collegial atmosphere. We are looking for candidates who have demonstrated exceptional promise through novel research with strong potential connections to natural, information, and engineering sciences. Research areas of particular interest include applied mathematics, computational science, as well as computing. A commitment to high-quality teaching and mentoring is expected.

The initial appointment at the assistant-professor level is for four years and is contingent upon the completion of a Ph.D. degree in Applied Mathematics, Computer Science, or related field. Exceptionally well-qualified applicants may also be considered at the full professor level.

To ensure the fullest consideration, applicants are encouraged to have all their application materials on file by December 28, 2014. For a list of documents required and full instructions on how to apply on-line, please visit <http://www.cms.caltech.edu/search>. Questions about the application process may be directed to: search@cms.caltech.edu.

Caltech is an Equal Opportunity/Affirmative Action Employer. Women, minorities, veterans, and disabled persons are encouraged to apply.

Florida Atlantic University (FAU)

Department of Computer and Electrical Engineering & Computer Science (CEECS)

Tenure-Track Assistant Professor Positions

The Department of Computer and Electrical Engineering & Computer Science (CEECS) at Florida Atlantic University (FAU) invites applications for multiple Tenure-Track Assistant Professor Positions. Priority research and teaching areas for the positions include Big Data and Data Analytics, Cyber Security, and related areas. Selected candidates for these positions are expected to start May 2015.

The applicant must have earned a doctorate degree in Computer Engineering, Electrical Engineering, Computer Science, or a closely related field, by the time of expected employment. The successful candidate must show potential for developing a strong research program with emphasis on competitive external funding and for excellence in teaching at the undergraduate and graduate levels. Excellent communications skills, both verbal and written, as judged by faculty and students, are essential. Competitive start-up packages will be available for these positions.

FAU, which has over 30,000 students, is one

of the 12 state universities in Florida. The CEECS department is located at the FAU Boca Raton campus. The department offers B.S., M.S. and Ph.D. degrees in Computer Science, Computer Engineering and Electrical Engineering, and M.S. degrees in Bioengineering. It has over 660 undergraduate, 140 M.S. and 80 Ph.D. students. Joint programs and collaborations exist and are encouraged between the CEECS faculty and the internationally recognized research organizations Scripps Florida, Max Planck Florida and the Harbor Branch Oceanographic Institute, all located on FAU campuses.

Applications must be made by completing the Faculty Application Form available on-line through the Office of Human Resources: <https://jobs.fau.edu> and apply to position 978786. Candidates must upload a complete application package consisting of a cover letter, statements of both teaching and research goals, a detailed CV, copy of official transcript, and names, addresses, phone numbers and email addresses of at least three references. The selected candidates will be required to pass the university's background check.

Interviews are expected to start in February 2015; applicants are strongly urged to apply as soon as possible. For any further assistance please e-mail to ceecs_search@eng.fau.edu. Florida Atlantic University is an Equal Opportunity/Equal Access institution. All minorities and members of underrepresented groups are encouraged to apply. Individuals with disabilities requiring accommodation, call 561-297-3057. TTY/TDD 1-800-955-8771

Florida State University

Department of Computer Science

Tenure-Track Assistant Professor

The Department of Computer Science at the Florida State University invites applications for one tenure-track Assistant Professor position to begin August 2015. The position is 9-mo, full-time, tenure-track, and benefits eligible. We are seeking outstanding applicants with strengths in the areas of Big Data and Digital Forensics. Outstanding applicants specializing in other research areas will also be considered. Applicants should hold a PhD in Computer Science or closely related field, and have excellent research and teaching accomplishments or potential. The department offers degrees at the BS, MS, and PhD levels. The department is an NSA Center of Academic Excellence in Information Assurance Education (CAE/IAE) and Research (CAE-R).

FSU is classified as a Carnegie Research I university. Its primary role is to serve as a center for advanced graduate and professional studies while emphasizing research and providing excellence in undergraduate education. Further information can be found at <http://www.cs.fsu.edu>

Screening will begin January 1, 2015 and

will continue until the position is filled. Please apply online with curriculum vitae, statements of teaching and research philosophy, and the names of five references, at <http://www.cs.fsu.edu/positions/apply.html>

Questions can be e-mailed to Prof. Xiuwen Liu, Faculty Search Committee Chair, recruitment@cs.fsu.edu.

Equal Employment Opportunity

An Equal Opportunity/Access/Affirmative Action/Pro Disabled & Veteran Employer committed to enhancing the diversity of its faculty and students. Individuals from traditionally underrepresented groups are encouraged to apply.

FSU's Equal Opportunity Statement can be viewed at: http://www.hr.fsu.edu/PDF/Publications/diversity/EEO_Statement.pdf

Fordham University

Department of Computer and Information Science

Assistant Professor, Cybersecurity Specialization

Fordham University invites applications for a tenure track position of Assistant Professor in the Department of Computer and Information Science, cybersecurity specialization. For complete position description, see <http://www.cis.fordham.edu/openings.html>. Electronic applications may be submitted to Interfolio Scholar Services: apply.interfolio.com/25961.

Indiana University – Purdue University Fort Wayne, Indiana

Department of Computer Science

Assistant Professor

The Dept. of Computer Science invites applications to fill two faculty positions for Assistant Professor (Tenure-Track) beginning 8/17/15. Outstanding candidates must have strong expertise in Software Engineering. Candidates specializing in Mobile and Embedded Systems, Cyber Security, Cloud Computing, Graphics & Visualization, or Bioinformatics in addition to some Software Engineering experiences will be considered.

Requirements include a Ph.D. in Computer Science or closely related field; and an exceptional record of research that supports the teaching and research missions of the Department; excellent communication and interpersonal skills; and an interest in working with students and collaborating with the business community.

Required duties include teaching undergraduate and graduate level courses in Computer Science, academic advising, and a strong pursuit of scholarly endeavors. Service to and engagement with the University, Department, and community is also required.

Please submit a letter of application address-

NATIONAL RESEARCH FOUNDATION
PRIME MINISTER'S OFFICE
SINGAPORE

Singapore NRF Fellowship 2016

The Singapore National Research Foundation (NRF) invites outstanding researchers in the early stage of research careers to apply for the Singapore NRF Fellowship.

The Singapore NRF Fellowship offers:

- **A five-year research grant of up to SGD 3 million (approximately USD 2.4 million).**
- **Freedom to lead ground-breaking research in any discipline of science and technology in a host institution of choice in Singapore.**

The NRF Fellowship is open to all researchers (PhD holders) who are ready to lead independent research in Singapore.

Apply online at the following web-link before
25 February 2015, 3 PM (GMT+8):

https://rita.nrf.gov.sg/AboutUs/NRF_Initiatives/nrff2016/default.aspx

For further queries, email us at
NRF_Fellowship@nrf.gov.sg

About the National Research Foundation

The National Research Foundation (NRF) was set up on 1 January 2006 as a department within the Prime Minister's Office, Singapore. NRF sets the national direction for policies, plans and strategies for research, innovation and enterprise. It funds strategic initiatives and builds research and development (R&D) capabilities. The NRF aims to transform Singapore into a vibrant R&D hub and a magnet for excellence in science and technology, contributing to a knowledge-intensive innovation-driven society, and building a better future for Singaporeans. For more details, please refer to **www.nrf.gov.sg**

ing qualifications, your CV, teaching and research statements, an example of your teaching experience or effectiveness, and a 1-2 page teaching philosophy. Include names and contact information for three references. All materials must be submitted electronically in .pdf format to Lisa Davenport, Secretary for the Department of Computer Science at davenport@ipfw.edu. All candidates who are interviewed should prepare a 45-60 minute instructional student presentation.

Review of applications will begin immediately and will continue until the positions are filled. Please see an extended ad www.ipfw.edu/vcaa/employment/. IPFW is an EEO/AA employer. All individuals, including minorities, women, individuals with disabilities, and protected veterans are encouraged to apply.

New Jersey Institute of Technology **College of Computing Sciences at NJIT** *Assistant, Associate, or Full Professor in Cyber Security*

New Jersey Institute of Technology invites applications for two tenured/tenure track positions beginning Fall 2015. The candidates should work on cyber security.

Specifically, the first position will focus on software security, with expertise in areas that include: software assurance/verification/validation/certification, static and dynamic software analysis, malware analysis, security and privacy of mobile systems and apps.

The second position will focus on network and systems security, with expertise in areas that include: networked systems and cloud computing security, security of critical infrastructure systems and emerging technologies such as IoT and wearable devices, application of machine learning techniques in network/system security.

Applicants must have a Ph.D. by summer 2015 in a relevant discipline, and should have an excellent academic record, exceptional potential for world-class research, and a commitment to both undergraduate and graduate education. The successful candidate will contribute to and enhance existing research and educational programs that relate to the cyber security as well as participate in our planned Cyber Security Center. Prior work experience or research collaboration with government/industry and a strong record of recent sponsored research represent a plus.

NJIT is committed to building a diverse faculty and strongly encourage applications from women candidates.

To Apply:

1. Go to njit.jobs, click on Search Postings and then enter the following posting numbers: 0602424 for the Secure Software position, and 0602426 for the Network and Systems Security Position.

2. Create your application, and upload your cover letter, CV, Research Statement, and Teaching Statement on that site. The CV must include at least three names along with contact information for references.

The applications will be evaluated as they are received and accepted until the positions are filled.

Contact: cs-faculty-search@njit.edu.

To build a diverse workforce, NJIT encourages

applications from individuals with disabilities, minorities, veterans and women. EEO employer.

NEW JERSEY INSTITUTE OF TECHNOLOGY
University Heights, Newark, NJ 07102-1982

Purdue University Tenure-Track/Tenured Faculty Positions

The Department of Computer Science at Purdue University is entering a phase of significant growth, as part of a university-wide Purdue Moves initiative. Applications are solicited for tenure-track and tenured positions at the Assistant, Associate and Full Professor levels. Outstanding candidates in all areas of computer science will be considered. Review of applications and candidate interviews will begin early in October 2014, and will continue until the positions are filled.

The Department of Computer Science offers a stimulating and nurturing academic environment with active research programs in most areas of computer science. Information about the department and a description of open positions are available at <http://www.cs.purdue.edu>.

Applicants should hold a PhD in Computer Science, or related discipline, be committed to excellence in teaching, and have demonstrated excellence in research. Successful candidates will be expected to conduct research in their fields of expertise, teach courses in computer science, and participate in other department and university activities. Salary and benefits are competitive, and Purdue is a dual career friendly employer. Applicants are strongly encouraged to apply online

at <https://hiring.science.purdue.edu>. Alternatively, hardcopy applications can be sent to: Faculty Search Chair, Department of Computer Science, 305 N. University Street, Purdue University, West Lafayette, IN 47907. A background check will be required for employment. Purdue University is an EEO/AA employer fully committed to achieving a diverse workforce. All individuals, including minorities, women, individuals with disabilities, and protected veterans are encouraged to apply.

Qatar University Associate/Full Research Professor in Cyber Security

Qatar University invites applications for research faculty positions at all levels with an anticipated starting date before September 2015. Candidates will cultivate and lead research projects at the KINDI Center for Computing Research in the area of Cyber Security. Qatar University offers competitive benefits package including a 3-year renewable contract, tax free salary, free furnished accommodation, and more. Apply by posting your application on: <https://careers.qu.edu.qa> "Under College of Engineering".

Queens College CUNY Assistant to Full Professor Data Science

Ph.D. with significant research experience in the applying or doing research in the area of data science and "big data" related to problems arising in

the fields of population, demography, linguistics, economics, sociology or other areas. Review begins 1/15/15. Open until filled.

For more information on the position and instructions on how to apply, please visit the Queens College Human Resources website and **click on Job ID 10668**. <http://www.qc.cuny.edu/HR/Pages/JobListings.aspx>

Skidmore College Visiting Assistant Professor/Lecturer

The Skidmore College Department of Mathematics and Computer Science seeks a qualified full-time computer science instructor for Fall 2015 and Spring 2016. The courses have yet to be determined.

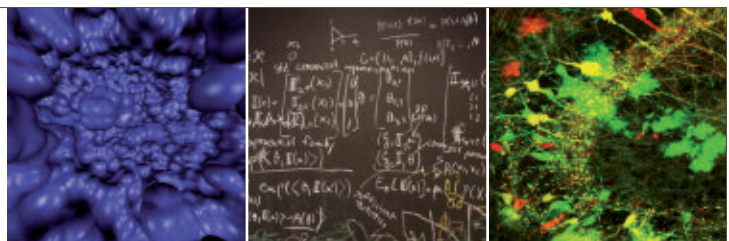
Minimum qualifications: MA or MS in Computer Science. Preferred qualification's: PhD in Computer Science and Teaching experience.

Review of applications begins immediately and will continue until the position is filled.

To learn more about and apply for this position please visit us online at: <https://careers.skidmore.edu/applicants/Central?quickFind=56115>

South Dakota State University Department of Electrical Engineering and Computer Science Brookings, South Dakota Assistant Professor of Computer Science

This is a 9-month renewable tenure track position; open August 22, 2015. An earned Ph.D. in



ISTFELLOW: Call for Postdoctoral Fellows

Are you a talented, dynamic, and motivated scientist looking for an opportunity to conduct research in the fields of **BIOLOGY, COMPUTER SCIENCE, MATHEMATICS, PHYSICS, or NEUROSCIENCE** at a young, thriving institution that fosters scientific excellence and interdisciplinary collaboration?

Apply to the ISTFellow program. Deadlines March 15 and September 15

www.ist.ac.at/istfellow



 Co-funded by
the European Union

IST AUSTRIA
Institute of Science and Technology

Computer Science or a closely related field is required by start date. Successful candidates must have a strong commitment to academic and research excellence. Candidate should possess excellent and effective written, oral, and interpersonal skills. Primary responsibilities will be to teach in the various areas of computer science, to participate widely in the CS curriculum, and to conduct research in the areas of big data, computer security, or machine learning, and related areas. To apply, visit <https://YourFuture.sdbor.edu>, search for posting number 0006832, and follow the electronic application process. For questions on the electronic employment process, contact SDSU Human Resources at (605) 688-4128. For questions on the position, contact Dr. Manki Min, Search Chair, at (605) 688-6269 or manki.min@sdsu.edu. SDSU is an AA/EEO employer. **APPLICATION DEADLINE:** Open until filled. Review process starts Feb. 20, 2015.

The State University of New York at Buffalo

Department of Computer Science and Engineering
Lecturer Position Available

The State University of New York at Buffalo Department of Computer Science and Engineering invites candidates to apply for a non-tenure track lecturer position beginning in the 2015-2016 academic year. We invite candidates from all areas of computer science and computer engineering who have a passion for teaching to apply.

The department has a strong commitment to hiring and retaining a lecturer for this career-oriented position, renewable for an unlimited number of 3-year terms. Lecturers are eligible for the in-house titles of Teaching Assistant Professor, Teaching Associate Professor and Teaching Professor.

Applicants should have a PhD degree in computer science, computer engineering, or a related field, by August 15, 2015. The ability to teach at all levels of the undergraduate curriculum is essential, as is a potential for excellence in teaching, service and mentoring. A background in computer science education, a commitment to K-12 outreach, and addressing the recruitment and retention of underrepresented students are definite assets.

Duties include teaching and development of undergraduate Computer Science and Computer Engineering courses (with an emphasis on lower-division), advising undergraduate students, as well as participation in department and university governance (service). Contribution to research is encouraged but not required.

Review of applications will begin on January 15, 2015, but will continue until the position is filled. **Applications must be submitted electronically via <http://www.ubjobs.buffalo.edu/>. Please use posting number 1400806 to apply.** The University at Buffalo is an Equal Opportunity Employer.

The Department, School and University

Housed in the School of Engineering and Applied Sciences, the Computer Science and Engineering department offers both BA and BS degrees

in Computer Science and a BS in Computer Engineering (accredited by the Engineering Accreditation Commission of ABET), a combined 5-year BS/MS program, a minor in Computer Science, and two joint programs (BA/MBA and Computational Physics).

The department has 34 tenured and tenure-track faculty and 4 teaching faculty, approximately 640 undergraduate majors, 570 masters students, and 150 PhD students. Fifteen faculty have been hired in the last five years. Eight faculty are NSF CAREER award recipients. Our faculty are active in interdisciplinary programs and centers devoted to biometrics, bioinformatics, biomedical computing, cognitive science, document analysis and recognition, high performance computing, information assurance and cyber security, and computational and data science and engineering.

The State University of New York at Buffalo (UB) is New York's largest and most comprehensive public university, with approximately 20,000 undergraduate students and 10,000 graduate students.

City and Region

Buffalo is the second largest city in New York state, and was rated the 10th best place to raise a family in America by Forbes magazine in 2010 due to its short commutes and affordability. Located in scenic Western New York, Buffalo is near the world-famous Niagara Falls, the Finger Lakes, and the Niagara Wine Trail. The city is renowned for its architecture and features excellent museums, dining, cultural attractions, and several professional sports teams, a revitalized downtown



上海科技大学
ShanghaiTech University

TENURE-TRACK AND TENURED POSITIONS IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

The newly launched **ShanghaiTech University** invites highly qualified candidates to fill multiple tenure-track/tenured faculty positions as its core team in the School of Information Science and Technology (SIST). Candidates should have exceptional academic records or demonstrate strong potential in cutting-edge research areas of information science and technology. They must be fluent in English. Overseas academic connection or background is highly desired.

ShanghaiTech is built as a world-class research university for training future generations of scientists, entrepreneurs, and technological leaders. Located in Zhangjiang High-Tech Park in the cosmopolitan Shanghai, ShanghaiTech is ready to trail-blaze a new education system in China. Besides establishing and maintaining a world-class research profile, faculty candidates are also expected to contribute substantially to graduate and undergraduate education within the school.

Academic Disciplines: We seek candidates in all cutting edge areas of information science and technology. Our recruitment focus includes, but is not limited to: computer architecture and technologies, nano-scale electronics, high speed and RF circuits, intelligent and integrated signal processing systems, computational foundations, big data, data mining, visualization, computer vision, bio-computing, smart energy/power devices and systems, next-generation networking, as well as inter-disciplinary areas involving information science and technology.

Compensation and Benefits: Salary and startup funds are highly competitive, commensurate with experience and academic accomplishment. We also offer a comprehensive benefit package to employees and eligible dependents, including housing benefits. All regular ShanghaiTech faculty members will be within its new tenure-track system commensurate with international practice for performance evaluation and promotion.

Qualifications:

- A detailed research plan and demonstrated record/potentials;
- Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, or related field)
- A minimum relevant research experience of 4 years.

Applications: Submit (in English, PDF version) a cover letter, a 2-page research plan, a CV plus copies of 3 most significant publications, and names of three referees to: sist@shanghaitech.edu.cn (until positions are filled). For more information, visit <http://www.shanghaitech.edu.cn>.

Deadline: February 28, 2015

UNLV

Big Data

Full-Time, Tenure-Track or Tenure Faculty position at the Assistant/Associate/Full Professor level in the area of Big Data in the Department of Computer Science (15257).

The Department of Computer Science at the University of Nevada, Las Vegas invites applications for a position in Big Data commencing Fall 2015. The candidates are expected to have an extensive research background in core areas associated with Big Data. More specifically, the applicants should have established records in one or more areas of machine learning, scalable computing, distributed/parallel computing, and database modeling/visualization of Big Data.

Applicants at assistant or associate level must have a Ph.D. in Computer Science from an accredited college or university. The professor rank is open to candidates with a Ph.D. in Computer Science or related fields from an accredited college or university with substantial history of published work and research funding. All applicants regardless of rank must demonstrate a strong software development background.

A complete job description with application details may be obtained by visiting <http://jobs.unlv.edu/>. For assistance with UNLV's online applicant portal, contact UNLV Employment Services at (702) 895-2894 or hrsearch@unlv.edu.

UNLV is an Equal Opportunity/Affirmative Action Educator and Employer Committed to Achieving Excellence Through Diversity.

waterfront as well as a growing local tech and start-up community. Buffalo is home to Z80, a start-up incubator, and 43 North, the world's largest business plan competition.

Texas A&M University - Central Texas
Assistant Professor - Computer Information Systems

TERM: 9 months/Tenure Track Teaches a variety of undergraduate and/or graduate courses in CIS and CS. Earned doctorate in IS or related areas such as CS. Recent Graduates or nearing completion of the doctorate are encouraged to apply. Apply: https://www.tamuctjobs.com/applicants/jsp/shared/Welcome_css.jsp

University of Central Florida CRCV
UCF Center for Research in Computer Vision
Assistant Professor

CRCV is looking for multiple tenure-track faculty members in the Computer Vision area. Of particular interest are candidates with a strong track record of publications. CRCV will offer competitive salaries and start-up packages, along with a generous benefits package offered to employees at UCF.

Faculty hired at CRCV will be tenured in the Electrical Engineering & Computer Science department and will be required to teach a maximum of two courses per academic year and are expected to bring in substantial external research funding. In addition, Center faculty are expected to have a vigorous program of graduate student mentoring and are encouraged to involve undergraduates in their research.

Applicants must have a Ph.D. in an area appropriate to Computer Vision by the start of the appointment and a strong commitment to academic activities, including teaching, scholarly publications and sponsored research. Preferred applicants should have an exceptional record of scholarly research. In addition, successful candidates must be strongly effective teachers.

To submit an application, please go to: <http://www.jobswithucf.com/postings/34681>

Applicants must submit all required documents at the time of application which includes the following: Research Statement; Teaching Statement; Curriculum Vitae; and a list of at least three references with address, phone numbers and email address.

Applicants for this position will also be considered for position numbers 38406 and 37361.

UCF is an Equal Opportunity/Affirmative Action employer. Women and minorities are particularly encouraged to apply.

University of Houston Clear Lake
Assistant Professor of Computer Science or Computer Information Systems

The University of Houston-Clear Lake CS and CIS programs invite applications for two tenure-track Assistant Professor positions to begin August 2015. A Ph.D. in CS or CIS, or closely related field is required. Applications accepted online only at <https://jobs.uhcl.edu/postings/9077.AA/EOE>.

University of Illinois at Chicago
Department of Computer Science
Non-tenure Track Full Time Teaching Faculty

The Computer Science Department at the University of Illinois at Chicago is seeking one or more full-time, non-tenure track teaching faculty members beginning Fall 2015. The department is committed to effective teaching, and candidates would be working alongside five full-time teaching faculty with over 75 years of combined teaching experience and 10 awards for excellence in teaching. Content areas of interest include introductory programming/data structures, theory/algorithms, artificial intelligence, computer systems, and software design. The teaching load is three undergraduate courses per semester, with a possibility of teaching at the graduate level if desired. Candidates must hold a master's degree or higher in Computer Science or a related field, and have demonstrated evidence of effective teaching.

The University of Illinois at Chicago (UIC) is ranked in the top-5 best US universities under 50 years old (Times Higher Education), and one of the top-10 most diverse universities in the US (US News and World Report). UIC's hometown of Chicago epitomizes the modern, livable, vibrant city. Located on the shore of Lake Michigan, it offers an outstanding array of cultural and culinary experiences. As the birthplace of the modern skyscraper, Chicago boasts one of the world's tallest and densest skylines, combined with an 8100-acre park system and extensive public transit and biking networks. Its airport is the second busiest in the world, with frequent non-stop flights to most major cities. Yet the cost of living, whether in a high-rise downtown or a house on a tree-lined street in one of the nation's finest school districts, is surprisingly low.

Applications are submitted online at <https://jobs.uic.edu/>. In the online application, please include your curriculum vitae, the names and addresses of at least three references, a statement providing evidence of effective teaching, and a separate statement describing your past experience in activities that promote diversity and inclusion and/or plans to make future contributions. Applicants needing additional information may contact Professor Joe Hummel, Search Committee Chair, jhummel2@uic.edu.

For fullest consideration, please apply by January 15, 2015. We will continue to accept and process applications until the positions are filled. UIC is an equal opportunity and affirmative action employer with a strong institutional commitment to the achievement of excellence and diversity among its faculty, staff, and student body. Women and minority applicants, veterans and persons with disabilities are encouraged to apply, as are candidates with experience with or willingness to engage in activities that contribute to diversity and inclusion.

University of Illinois at Chicago
Department of Computer Science
Faculty - Tenure Track - Computer Science

The Computer Science Department at the University of Illinois at Chicago invites applications in all areas of Computer Science for multiple tenure-track positions at the rank of Assistant Pro-

fessor (exceptional candidates at other ranks will also be considered). We are looking to fill:

- (a) One position in Big Data, where our focus ranges from data management and analytics to visualization and applications involving large volumes of data.
- (b) Two positions in Computer Systems, where we are looking for candidates whose work is experimental and related to one or more of the following topics: operating systems, networking, distributed computing, mobile systems, programming languages and compilers, security, software engineering, and other broadly related areas.
- (c) One position for which candidates from all other areas will be considered.

The University of Illinois at Chicago (UIC) ranks among the nation's top 50 universities in federal research funding and is ranked 4th best U.S. University under 50 years old. The Computer Science department has 24 tenure-track faculty representing major areas of computer science, and offers BS, MS and PhD degrees. Our faculty includes ten NSF CAREER award recipients. We have annual research expenditures of \$8.4M, primarily federally funded. UIC is an excellent place



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Rates: \$325.00 for six lines of text, 40 characters per line. \$32.50 for each additional line after the first six. The MINIMUM is six lines.

Deadlines: 20th of the month/2 months prior to issue date. For latest deadline info, please contact:

acmm mediasales@acm.org

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://jobs.acm.org>

Ads are listed for a period of 30 days.

For More Information Contact:

**ACM Media Sales,
at 212-626-0686 or
acmm mediasales@acm.org**

for interdisciplinary work—with the largest medical school in the country and faculty engage in several cross-departmental collaborations with faculty from health sciences, social sciences and humanities, urban planning, and the business school. UIC has an advanced networking infrastructure in place for data-intensive scientific research that is well-connected regionally, nationally and internationally. UIC also has strong collaborations with Argonne National Laboratory and the National Center for Supercomputing Applications, with UIC faculty members able to apply for time on their high-performance supercomputing systems.

Chicago epitomizes the modern, livable, vibrant city. Located on the shore of Lake Michigan, it offers an outstanding array of cultural and culinary experiences. As the birthplace of the modern skyscraper, Chicago boasts one of the world's tallest and densest skylines, combined with an 8100-acre park system and extensive public transit and biking networks. It's airport is the second busiest in the world. Yet the cost of living, whether in a 99th floor condominium downtown or on a tree-lined street in one of the nation's finest school districts, is surprisingly low. **Applications must be submitted at <https://jobs.uic.edu/>.** Please include a curriculum vitae, teaching and research statements, and names and addresses of at least three references in the online application. Applicants needing additional information may contact the Faculty Search Chair at search@cs.uic.edu. The University of Illinois is an Equal Opportunity, Affirmative Action employer. Minorities, women, veterans and individuals with disabilities are encouraged to apply.

University of Tartu Professor of Data Management and Analytics

The Institute of Computer Science of **University of Tartu** invites applications for the position of: **Full Professor of Data Management and Analytics**.

The successful candidate will have a solid and sustained research track record in the fields of data management, data analytics or data mining, including publications in top venues; a demonstrated record of excellence in teaching and student supervision; a recognized record of academic leadership; and a long-term research and teaching vision.

University of Tartu is the leading higher education and research centre in Estonia, with more than 16000 students and 1800 academic staff. It is the highest ranked university in the Baltic States according to both QS World University rankings and THE ranking. University of Tartu's Institute of Computer Science hosts 600 Bachelors and Masters students and around 50 doctoral students. The institute is home to internationally recognized research groups in the fields of software engineering, distributed and cloud computing, bioinformatics and computational neuroscience, cryptography, programming languages and systems, and language technology. The institute delivers Bachelors, Masters and PhD programs in Computer Science, as well as joint specialized Masters in software engineering, cyber-security and security and mobile computing, in cooperation with other leading universities in Estonia and Scandinavia. The institute has a strong international orientation: over 40% of graduate

students and a quarter of academic and research staff members are international. Graduate teaching in the institute is in English.

The duties of a professor include research and research leadership, student supervision, graduate and undergraduate teaching in English or Estonian (128 academic hours per year) as well as teaching coordination and academic leadership. The newly appointed professor will be expected to create a world-class research group in their field of specialty and to solidify and expand the existing teaching capacity in this field.

The appointment will be permanent. Gross salary is 5000 euros per month. Estonia applies a flat income tax of 20% on salaries and provides public health insurance for employees. Other benefits include 56 days of annual leave and a sabbatical semester per 5-years period of appointment. Relocation support will be provided if applicable. In addition, a seed funding package shall be negotiated. Besides access to EU funding instruments, Estonia has a merit-based national research funding system enabling high-performing scholars to create sustainable research groups.

The position is permanent. The starting date is negotiable between the second half of 2015 and first half of 2016.

The position is funded by the Estonian IT Academy programme.

The deadline for applications is 31 March 2015. Information about the application procedure and employment conditions at University of Tartu can be found at <http://www.ut.ee/en/employment>.

Apply URL: <http://www.ut.ee/en/2317443/data-management-and-analytics-professor>

University of Texas at Dallas Tenure Track Positions in Computer Science

The Department of Computer Science of The University of Texas at Dallas invites applications from outstanding applicants for **multiple tenure track positions in Computer Science**. Candidates in all areas of Computer Science will be considered though the department is particularly interested in areas of machine learning, information retrieval, software engineering, data science, cyber security and computer science theory. Candidates must have a PhD degree in Computer Science, Software Engineering, Computer Engineering or equivalent. The positions are open for applicants at all ranks. Candidates for senior positions must have a distinguished research, publication, teaching and service record, and demonstrated leadership ability in developing and expanding (funded) research programs. An endowed chair may be available for highly qualified senior candidates. Junior candidates must show outstanding promise.

The Department offers BS, MS, and PhD degrees both in Computer Science and Software Engineering, as well as in interdisciplinary fields of Telecom Engineering and Computer Engineering. Currently the Department has a total of 47 tenure-track faculty members and 23 senior lecturers. The department is housed in a spacious 150,000 square feet facility and has excellent computing equipment and support. The department houses a number of centers and institutes, particularly, in areas of net centric software, cyber

security and human language technology.

The University is located in the most attractive suburbs of the Dallas metropolitan area. There are over 800 high-tech companies within few miles of the campus, including Texas Instruments, Alcatel, Ericsson, Hewlett-Packard, AT&T, Fujitsu, Raytheon, Rockwell Collins, Cisco, etc. Almost all the country's leading telecommunication's companies have major research and development facilities in our neighborhood. Opportunities for joint university-industry research projects are excellent. The Department received more than \$27 Million in new research funding in the last three years. The University and the State of Texas are also making considerable investment in commercialization of technology developed in University labs: a new start-up business incubation center was opened in September 2011.

The search committee will begin evaluating applications on January 15th. Applications received on or before January 31st will get highest preference. Indication of gender and ethnicity for affirmative action statistical purposes is requested as part of the application. For more information contact Dr. Gopal Gupta, Department Head, at gupta@utdallas.edu or send e-mail to cs-search@utdallas.edu or view the Internet Web page at <http://cs.utdallas.edu>.

Applicants should provide the following information: (1) resume, (2) statement of research and teaching interests, and (3) full contact information for three, or more, professional references via the ONLINE APPLICATION FORM available at: <http://go.utdallas.edu/pcx141118>.

EOE/AA

York University Department of Electrical Engineering and Computer Science Assistant or Associate Lecturer

The Department of Electrical Engineering and Computer Science (EECS) York University is seeking an outstanding candidate for an alternate-stream tenure-track position at the **Assistant or Associate Lecturer** level to teach relevant core areas of engineering and play a leading role in developing and assessing curriculum as a Graduate Attributes Coordinator. While outstanding candidates in all areas of EECS will be considered, we are especially interested in those with strong abilities to develop and teach courses in systems areas to complement the Department's existing strengths. Systems areas include, but are not limited to: computer architecture, operating systems, embedded systems and allied areas. Priority will be given to candidates licensed as Professional Engineers in Canada. Complete applications must be received by 15 March 2015. **Full job description and application details are available at: <http://lassonde.yorku.ca/new-faculty/>.** York University is an Affirmative Action (AA) employer and strongly values diversity, including gender and sexual diversity, within its community. The AA Program, which applies to Aboriginal people, visible minorities, people with disabilities, and women, can be found at www.yorku.ca/acadjobs or by calling the AA office at 416-736-5713. All qualified candidates are encouraged to apply; however, Canadian citizens and Permanent Residents will be given priority.



2015

ACM INTERNATIONAL CONFERENCE
ON INTERACTIVE EXPERIENCES FOR
TELEVISION AND ONLINE VIDEO

3-5 JUNE, 2015
BRUSSELS, BELGIUM

Paper Submissions by
12 January 2015

Work in Progress, Demos,
DC, & Industrial
Submissions by
2 March 2015

Welcoming Submissions on
Content Production
Systems & Infrastructures
Devices & Interaction Techniques
Experience Design & Evaluation
Media Studies

Data Science & Recommendations
Business Models & Marketing
Innovative Concepts & Media Art



Association for
Computing Machinery



iMinds

TVX2015.COM

INFO@TVX2015.COM



DOI:10.1145/2699303

Dennis Shasha

Upstart Puzzles

Take Your Seats

A POPULAR LOGIC game involves figuring out an arrangement of people sitting around a circular table based on hints about, say, their relationships. Here, we aim to determine the smallest number of hints sufficient to specify an arrangement unambiguously. For example, suppose we must seat Alice, Bob, Carol, Sybil, Ted, and Zoe. If we are allowed hints only of the form X is one to the right of Y , it would seem four hints are necessary. But suppose we can include hints that still refer to two people, or “binary hints,” but in which X can be farther away from Y .

Suppose we have just three hints for the six people: Ted is two seats to the right of Carol; Sybil is two to the right of Zoe; and Bob is three to the right of Ted (see Figure 1 for this table arrangement). We see that we need only three hints to “fix” the relative locations of six people.

However, if we now bring Jack and Jill into the picture, for a total of eight

people, then we might ask how many binary hints we would need to fix the arrangement. Consider these five hints: Carol is three seats to the right of Jill; Alice is six to the right of Bob; Ted is four to the right of Zoe; Jill is six to the right of Zoe; and Carol is six to the right of Sybil. What arrangement would these hints produce?

Solution. Alice, Jill, Bob, Zoe, Carol, Jack, Sybil, and Ted. So we used five hints to fix the arrangement of eight people around a circular table.

Getting even more ambitious, suppose we add Christophe and Marie, giving us 10 people, and want the ordering to be like this: Christophe, Jack, Jill, Bob, Marie, Carol, Ted, Zoe, Alice, and Sybil (see Figure 2). Can you formulate seven hints that will fix this arrangement? Can you do it with fewer than seven? Here is one solution using seven hints: Alice is seven seats to the right of Jack; Jack is nine to the right of Jill;

Christophe is seven to the right of Bob; Christophe is six to the right of Marie; Bob is eight to the right of Carol; Ted is eight to the right of Alice; and Ted is seven to the right of Sybil.

Here are the upstart challenges, easier, I suspect, than the upstart challenge from my last (Nov. 2014) or next (May 2015) column: Is there an algorithm for finding $n-3$ binary hints to fix an arrangement of n people around a table for n of at least six? Is that algorithm “tight,” so it is impossible to do better?

Solutions to this and to other upstart challenges are at <http://cs.nyu.edu/cs/faculty/shasha/papers/cacmpuzzles.html>. □

All are invited to submit solutions and prospective upstart-style puzzles for future columns to upstartpuzzles@cacm.acm.org

Dennis Shasha (dennisshasha@yahoo.com) is a professor of computer science in the Computer Science Department of the Courant Institute at New York University, New York, as well as the chronicler of his good friend the omniheurist Dr. Ecco.

Copyright held by author.

Figure 1. Seating arrangement specified by the hints.

All participant rotations are permitted, so fixing the arrangement is unique up to rotation.



Figure 2. Find seven binary hints that will fix this arrangement.





Glasgow June 22-25

ACM Creativity and Cognition 2015 will serve as a premier forum for presenting the world's best new research investigating computing's impact on human creativity in a broad range of disciplines including the arts, design, science, and engineering.

Creativity and Cognition will be hosted by The Glasgow School of Art and the City of Glasgow College. The 2015 conference theme is Computers | Arts | Data. The theme will serve as the basis for a curated art exhibition, as well as for research presentations.



Call for Papers, Posters and Demonstrations

Papers submission deadline: 6th January

Posters submission deadline: 6th March

Demonstrations submission deadline: 6th March 2015

Call for Workshops

Deadline for submission: 6th March 2015

We invite Workshops to be presented on the day preceding the full conference

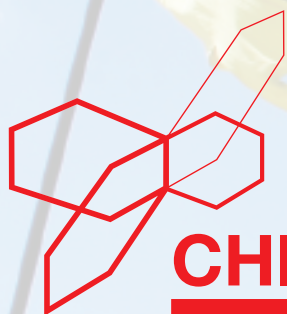
Call for Artworks

Deadline for submission: 6th March 2015

We are calling for proposals for artworks, music, performances and installations to be presented in conjunction with the conference.

Computers

+ Art + Data



CHI 2015
CROSSINGS
SEOUL • KOREA

18-23 APRIL 2015

REGISTER NOW

Come to CHI 2015, the premier international forum for human-computer interaction (HCI). This year's program has more technical content than ever before with breakthrough insights on the impact of human-centered innovation. The conference will take place in the COEX at the heart of Seoul's Gangnam district. Being held in Asia allows this year's conference to showcase groundbreaking content from across the region with special symposia for Chinese, Japanese and ASEAN insights and innovations. You will also hear fresh perspectives from exciting Asian speakers along with special presentations from speakers of diverse backgrounds and viewpoints.

Discounted early registration rates are available until 6 Mar 2015.

Late registration continues until 17 Apr 2015.

See website for registration and up-to-date details on the entire technical program:



LOU YONGQI

Dean of the College of
Design and Innovation ,
Tongji University,
Shanghai, China



**DONGHOON
CHANG**

Executive Vice
President, UX Center,
Samsung Electronics,
Korea



DAVID MIN

Senior Research
Fellow, Software
Research Center, LG
Electronics, Korea



PSY

Singer, songwriter,
record producer and
view-count breaker