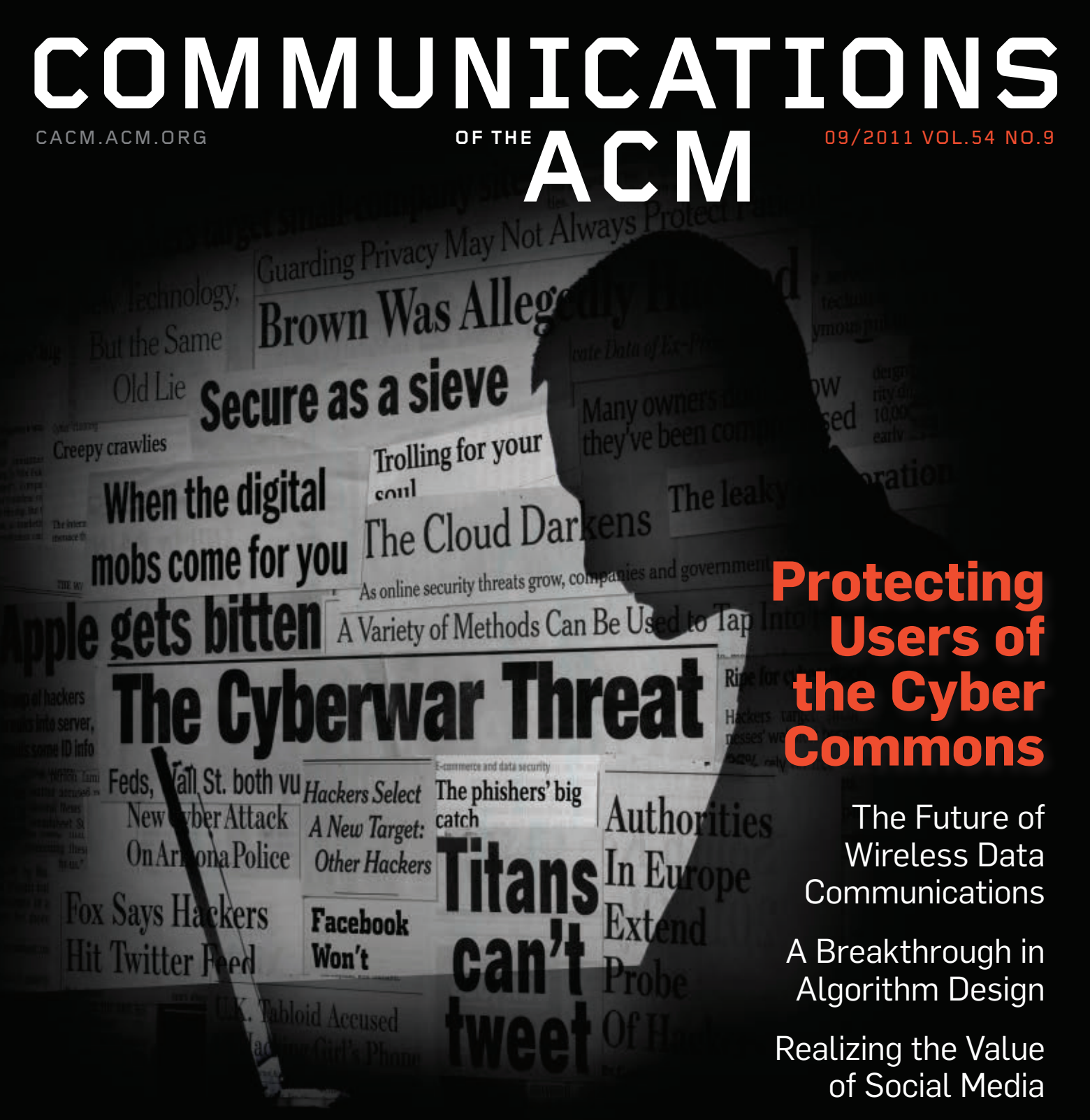
COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

09/2011 VOL.54 NO.9



Protecting Users of the Cyber Commons

The Future of Wireless Data Communications

A Breakthrough in Algorithm Design

Realizing the Value of Social Media

Abstracting Abstract Machines

Association for Computing Machinery

October 22–27, 2011

A SPLASH Conference

Hilton Portland & Executive Tower

Portland, Oregon USA

ONWARD! 2011

ACM Symposium on New Ideas in Programming and Reflections on Software

Chair
Robert Hirschfeld
Hasso-Plattner-Institut Potsdam, Germany
chair@onward-conference.org

Papers
Eelco Visser
Delft University of Technology, The Netherlands
papers@onward-conference.org

Workshops
Pascal Costanza
Vrije Universiteit Brussel, Belgium
workshops@onward-conference.org

Essays
David West
New Mexico Highlands University, USA
essays@onward-conference.org

Films
Bernd Bruegge
Technische Universität München, Germany
films@onward-conference.org

Association for Computing Machinery

http://onward-conference.org/

INTERACTIVE TABLETOPS AND SURFACES

november 13 - 16

2011 - Kobe - Japan

THE ACM INTERNATIONAL CONFERENCE ON INTERACTIVE TABLETOPS AND SURFACES

ACM ITS 2011

ACM ITS is the premiere venue for presenting research in the design and use of new and emerging tabletop and interactive surface technologies.

ITS 2011 will feature:

- Paper & Notes
- Demos
- Poster
- Doctoral Symposium
- Tutorials / Workshops

Conference Chairs

Jun Rekimoto
Hideki, Koike
Kentaro Fukuchi

Program Chairs

Yoshifumi Kitamura
Daniel Wigdor

www.its2011.jp

COMMUNICATIONS OF THE ACM

Departments

5 Editor's Letter
Are You Talking to Me?
By Moshe Y. Vardi

7 Letters to the Editor
Solved, for All Practical Purposes

9 In the Virtual Extension

10 BLOG@CACM
Jeannette M. Wing @ PCAST; Barbara Liskov Keynote
Jeannette M. Wing discusses her PCAST presentation about the importance of computer science and its impact. Valerie Barr shares highlights from Barbara Liskov's keynote at Grace Hopper.

27 Calendar

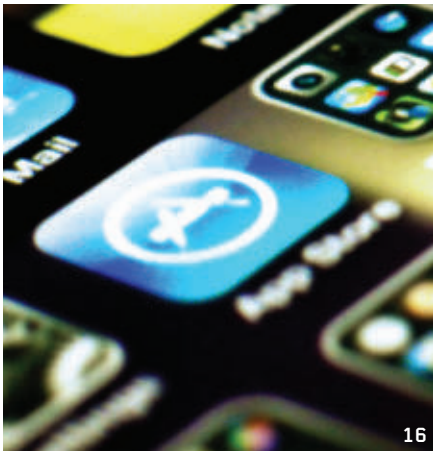
88 Careers

Last Byte

110 Puzzled
Solutions and Sources
By Peter Winkler

112 Q&A
Scaling Up
Eric Brewer talks about infrastructure, connectivity, and computing for developing nations.
By Leah Hoffmann

News



13 **A Breakthrough in Algorithm Design**
Computer scientists at Carnegie Mellon University have devised an algorithm that might be able to solve a certain class of linear systems much more quickly than today's fastest solvers.
By Kirk L. Kroeker

16 **Invasion of the Mobile Apps**
The market model pioneered by Apple and others is transforming the software world—and has profound implications for software companies and their customers.
By Gary Anthes

19 **Remaking American Medicine**
Developing an IT ecosystem for health could improve—and transform—the practice of medicine.
By Neil Savage

Viewpoints

22 Law and Technology
Remix Nation
Assessing the threat the anticircumvention provisions of the Digital Millennium Copyright Act pose for fair use.
By Rebecca Tushnet

25 Historical Reflections
In Praise of 'Wilkes, Wheeler, and Gill'
Reflections on the first textbook on programming.
By Martin Campbell-Kelly

28 Emerging Markets
Corporate Social Responsibility and Global IT Outsourcing
How to improve IT outsourcing relationships while doing good for society.
By Ron Babin, Steve Briggs, and Brian Nicholson

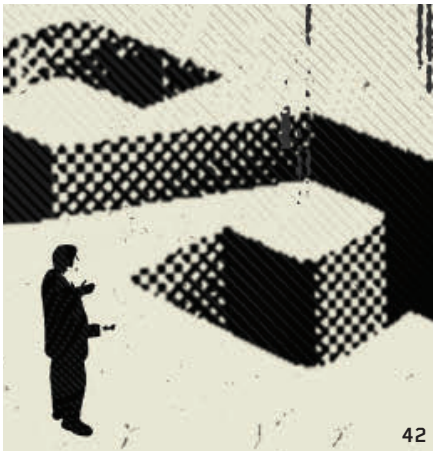
31 The Profession of IT
Managing Time, Part 2
Masterful time management means not just tracking of messages in your personal environment, but managing your coordination network with others.
By Peter J. Denning and Ritu Raj

34 Viewpoint
Realizing the Value of Social Media Requires Innovative Computing Research
How social media are expanding traditional research and development topics for computer and information scientists.
By Ben Shneiderman, Jennifer Preece, and Peter Pirolli

Association for Computing Machinery
Advancing Computing as a Science & Profession

PHOTOGRAPH BY DANIEL GO

Practice



38 **Arrogance in Business Planning**
Technology business plans that assume no competition—ever.
By Paul Vixie

42 **The Most Expensive One-Byte Mistake**
Did Ken, Dennis, and Brian choose wrong with NUL-terminated text strings?
By Poul-Henning Kamp

45 **ACM CTO Roundtable on Mobile Devices in the Enterprise**
Finding solutions as growth and fragmentation complicate mobile device support.
By Mache Creeger

Articles' development led by acmqueue
queue.acm.org

About the Cover:
Progress in protecting cyberspace has moved at a snail's pace, often adopting a fix-on-demand approach. This month's cover story introduces the notion of attacking the challenge with top-down *and* bottom-up processes designed to work together.

ILLUSTRATION BY GARY NEILL

Contributed Articles

54 **Protecting Users of the Cyber Commons**
Establish a global cyber "neighborhood watch" to enable users to take defensive action to protect their operations.
By Stephen J. Lukasik

62 **Realizing the Future of Wireless Data Communications**
Technologies exist to unlock radio spectrum as consumers need it.
By Craig Partridge

69 **Satisfiability Modulo Theories: Introduction and Applications**
vChecking the satisfiability of logical formulas, SMT solvers scale orders of magnitude beyond custom ad hoc solvers.
By Leonardo de Moura and Nikolaj Bjørner

VE **Calculating and Improving ROI in Software and System Programs**
The investment value of innovation follows from the technology's uncertain net present value and derived ROI calculations.
By Murray Cantor

VE **Crossing to the Dark Side: Examining Creators, Outcomes, and Inhibitors of Technostress**
Exploring the factors that may lead to the inability of professionals to adapt or cope with emerging IS in a healthy manner.
By Monideepa Tarafdar, Qiang Tu, T.S. Ragu-Nathan, and Bhanu S. Ragu-Nathan

Review Articles

78 **Quantitative Analysis of Real-Time Systems Using Priced Timed Automata**
Timed automata and their extensions allow for analysis of a wide range of performance and optimization problems.
By Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, and Nicolas Markey

Research Highlights

90 Technical Perspective
Making Browser Extensions Secure
By Christopher Kruegel

91 **Vetting Browser Extensions for Security Vulnerabilities with VEX**
By Sruthi Bandhakavi, Nandit Tiku, Wyatt Pittman, Samuel T. King, P. Madhusudan, and Marianne Winslett

100 Technical Perspective
Abstracting Abstract Machines
By Olivier Danvy and Jan Midtgaard

101 **Abstracting Abstract Machines: A Systematic Approach to Higher-Order Program Analysis**
By David Van Horn and Matthew Might

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.

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.

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 acmhelp@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

Association for Computing Machinery

Printed in the U.S.A.

DOI:10.1145/1995376.1995377

Moshe Y. Vardi

Are You Talking to Me?

I recently attended a rather theoretical computer-science conference, and sat, as is my habit, in the front row. The speaker was trying to convey the fine details of

a rather intricate mathematical construction. I was hopelessly lost. At that point I found the talk indistinguishable from Doug Zongker's celebrated "Chicken Chicken Chicken" talk presented at the 2007 AAAS Humor Session (http://www.youtube.com/watch?v=yL_-1d9OSdk). Looking behind me to see how other attendees were reacting to the highly dense presentation, I was greeted by a wall of laptop screens; people were busily reading their email.

At the business meeting that evening, I asked "How many people could follow 100% of 100% of the talks?" Silence. "80% of 80%?" One brave soul responded positively. It was only when I got to "50% of 50%" that about 50% of the participants raised their hands. Of course, this statistic should not be taken too seriously, but, nevertheless, I found it shocking! About 100 people are spending four days attending talks, and only 50% understand 50% of 50% the talks? What is the point of this futile exercise?

I am reminded of Lance Fortnow's pithy description of a computer-science conference as "a journal that meets at a hotel." Indeed, if the point of the conference is simply to score a prestigious publication, then attending the conference and giving a talk is just a hurdle that one must overcome as a condition of publication. As I pointed out in my May 2011 editorial, "Technology Has Social Consequences," many conferences eliminated face-to-face program-committee meetings in the late 1990s to save travel expenses and hassle. Why don't we take the next logical step and virtualize our conferences in the name of efficiency?

I am not serious, of course. I actually like conferences very much. I believe they are a critical component of the scientific enterprise. Science is a social undertaking. For most of us, our scientific social network is truly global. Meeting at conferences is the only way to maintain our links, learn what is happening, and tell others about our latest and greatest. While some of the activity of a conference happens in coffee breaks and hallways, its core activity takes place in the lecture halls, and this activity better be effective, which means the talks better be clear, informative, and interesting. Why is it then that we put so much attention on ensuring the quality of the papers, and so little attention on ensuring the quality of the talks?

There are many ways in which we can attempt to improve the quality of conference talks. Some of these measures are easy and obvious. For example, graduate students should be taught that preparing a good talk is quite different from, though equally important as, writing a good paper. They should never give a conference talk without some dry runs with brutally honest feedback from their advisor and fellow students. Also, for their first few conference talks, graduate students should be video-recorded. Many will be rather shocked when seeing and hearing themselves for the first time. This advice applies not only to graduate students. While students often make the rookie mistake of trying to tell the audience *everything* in their paper, rather than tell the audience *about* their paper, they are not the only ones giving poor talks.

Conferences should, in my opinion, take active measures to improve presentation quality. A radical proposal would be to require authors to submit not only papers but also video recordings of their talks. The quality of those presentations would be considered in making program decisions. Less radical a move is to require authors to send draft presentations before the conference, and receive feedback from their session chairs. It should also be relatively easy to augment conference-management systems with feedback pages where conference participants can give speakers anonymous feedback on their presentations. (That would give attendees something constructive to do during poor presentations!)

At some conferences, I have raised the issue of poor presentations, and encountered unwillingness by conference officials to take any concrete measure. I am told my proposals are "too intrusive," which is truly puzzling. We manage conference programs with an iron hand, often ruffling many feathers by (sometimes controversial) program decisions. Why are we suddenly "kinder and gentler" when it comes to presentation quality? If conferences are important, then we ought to treat them as more than "journals meeting at hotels" and make sure the time we spend attending them is well spent.

Moshe Y. Vardi, EDITOR-IN-CHIEF

Call for Nominations

The ACM Doctoral Dissertation Competition

Rules of the Competition

ACM established the Doctoral Dissertation Award program to recognize and encourage superior research and writing by doctoral candidates in computer science and engineering. These awards are presented annually at the ACM Awards Banquet.

Submissions

Nominations are limited to one per university or college, from any country, unless more than 10 Ph.D.'s are granted in one year, in which case two may be nominated.

Eligibility

Each nominated dissertation must have been accepted (successfully defended) by the department between October 2010 and September 2011. Exceptional dissertations completed in September 2010, but too late for submission last year will be considered. Only English language versions will be accepted. Please send a copy of the thesis in PDF format to emily.eng@acm.org.

Sponsorship

Each nomination shall be forwarded by the thesis advisor and must include the endorsement of the department head. A one-page summary of the significance of the dissertation written by the advisor must accompany the transmittal.

Deadline

Submissions must be received by **October 31, 2011** to qualify for consideration.

Publication Rights

Each nomination must be accompanied by an assignment to ACM by the author of exclusive publication rights. (Copyright reverts to author if not selected for publication.)

Publication

Winning dissertations will be published by ACM in the ACM Digital Library, not by Springer as previously noted.

Selection Procedure

Dissertations will be reviewed for technical depth and significance of the research contribution, potential impact on theory and practice, and quality of presentation. A committee of five individuals serving staggered five-year terms performs an initial screening to generate a short list, followed by an in-depth evaluation to determine the winning dissertation.

The selection committee will select the winning dissertation in early 2012.

Award

The Doctoral Dissertation Award is accompanied by a prize of $20,000 and the Honorable Mention Award is accompanied by a prize of $10,000. Financial sponsorship of the award is provided by Google.

For Submission Procedure

See http://awards.acm.org/html/dda.cfm

Association for Computing Machinery

DOI:10.1145/1995376.1995378

Solved, for All Practical Purposes

Moshe Y. Vardi's Editor's Letter "Solving the Unsolvable" (July 2011) raised an important point—that we should reconsider the meaning of unsolvability, especially in terms of its practical application. Even though a problem (such as the Halting Problem) may be theoretically unsolvable, we should, perhaps, still try to solve it.

The proof of undecidability is based on the possibility of self-application; that is, a program cannot look at itself and decide if it is itself stuck in a loop; from a practical point of view, this situation is not relevant. Why even write such a program? The proof does not say I cannot write a server program that looks at running applications to determine if any of them is in a loop.

The same reliance on self-application applies to the Post Correspondence Problem (PCP), a string-matching problem also theoretically unsolvable. The proof does not say PCP is undecidable for any practical problem, only for one using self-application. However, the proof does say if I try to simulate a Turing Machine program that looks to see if it is itself in a loop, then, as in the Halting Problem, PCP is theoretically unsolvable. But from a string-matching point of view, this potential insight about unsolvability is again hardly relevant to the programmer. Perhaps, for all cases of practical interest, PCP is indeed solvable.

The same point applies to the many other theorems that relate to the unsolvability of certain problems. It may be the problems are very difficult to solve; likewise, it may be very difficult to devise a solution for a reasonable sub-problem or solve a sub-problem in polynomial time. In any case, the question of unsolvability might simply be a red herring.

Henry Ledgard, Toledo, OH

Author's Response:

I do not agree that unsolvability is a "red herring" but a fundamental limit on computability. We do not have an algorithm for program termination. My point was we should take a sober view of unsolvability, recognizing that many unsolvable problems can, in practice, be solved.

Moshe Y. Vardi, Editor-in-Chief

To Program, Imagine All Contingencies

In his Viewpoint "Non-Myths About Programming" (July 2011), Mordechai Ben-Ari said programming requires logical thinking, which is certainly true, but to write a program that interacts with anything—API, device, UI—a programmer must also be able to imagine all contingencies and define appropriate responses. Such talent is orthogonal to following a theorem proof or manipulating algebraic expressions that would be needed for, say, a good grade in high school mathematics.

Tom Moran, Saratoga, CA

Author's Response:

I agree the definition of logical thinking should be as broad as possible. However, it is an empirical question whether success in high school mathematics predicts the logical thinking needed for programming. I conjecture that the correlation is positive (not 1.0, but certainly not 0.0, orthogonal) and thus a reasonable predictor for use by a guidance counselor.

Mordechai Ben-Ari, Rehovot, Israel

Where Privacy Ends

Besides being a great article on its subject, Stephen B. Wicker's "Cellular Telephony and the Question of Privacy" (July 2011) also identified a game-changing direction in privacy. Consider that the word "privacy" is oxymoronic when discussing radio transmission; by definition, a radio sends our stuff to places totally beyond our control or authority; think postcard rather than envelope. We can't give away something and still claim to own it and presume we can tell everyone else how to use it. To my knowledge, no legal precedent exists to empower a nail maker to decree all builders use its products only pointy-side down.

This is a trend (and fallacy) sanctified by the software industry (and others), claiming "It's mine, even when we have it." Absurd, of course, though it seems to function as the basis for everything from copyright law to digital privacy.

Utterances overheard at a distance are not private; neither are postcards, signs in the front yard, or a radio or wire-line signal. A government might wish to guarantee a certain right of privacy for some particular technology, except that such a guarantee would be a matter of contract law, not of practical expedience. The postal service guarantees privacy (within limits) as part of its service. The phone company does not. I know of no service that allows remote talking that also guarantees confidentiality. The guarantee is to try to ensure confidentiality, or good faith.

Our expectation of privacy ends when the communication leaves our point of control, save for specific guarantees from the final authority, in the U.S., the Federal Government.

What Wicker called "context information" cannot be made private by definition (or the service stops). Presuming protection of related content is just silly; A gives it to B, and B may now do whatever it wants with it or whatever it thinks it can get away with. Wrangling legalisms about what is permitted is the equivalent of rearranging deck chairs as the ship of privacy heads for the bottom.

David Byrd, Arlington, VA

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

Association for Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: AD10

You can join ACM in several easy ways:

Online
http://www.acm.org/join

Phone
+1-800-342-6626 (US & Canada)
+1-212-626-0500 (Global)

Fax
+1-212-944-1318

Or, complete this application and return with payment via postal mail

Special rates for residents of developing countries:
http://www.acm.org/membership/L2-3/

Special rates for members of sister societies:
http://www.acm.org/membership/dues.html

Please print clearly

Name

Address

City | State/Province | Postal code/Zip

Country | E-mail address

Area code & Daytime phone | Fax | Member number, if applicable

Purposes of ACM

ACM is dedicated to:
1) advancing the art, science, engineering, and application of information technology
2) fostering the open interchange of information to serve both professionals and the public
3) promoting the highest professional and ethics standards

I agree with the Purposes of ACM:

Signature

ACM Code of Ethics:
http://www.acm.org/serving/ethics.html

choose one membership option:

PROFESSIONAL MEMBERSHIP:

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

STUDENT MEMBERSHIP:

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

All new ACM members will receive an ACM membership card.
For more information, please visit us at www.acm.org

Professional membership dues include $40 toward a subscription to *Communications of the ACM*. Student membership dues include $15 toward a subscription to *XRDS*. Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

RETURN COMPLETED APPLICATION TO:

Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Questions? E-mail us at acmhelp@acm.org
Or call +1-800-342-6626 to speak to a live representative

Satisfaction Guaranteed!

payment:

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc. in US dollars or foreign currency at current exchange rate.

o Visa/MasterCard o American Express o Check/money order

o Professional Member Dues ($99 or $198) $ _____

o ACM Digital Library ($99) $ _____

o Student Member Dues ($19, $42, or $62) $ _____

Total Amount Due $ _____

Card # Expiration date

Signature

DOI:10.1145/1995376.1995379

In the Virtual Extension

To ensure the timely publication of articles, Communications *created the Virtual Extension (VE) to expand the page limitations of the print edition by bringing readers the same high-quality articles in an online-only format. VE articles undergo the same rigorous review process as those in the print edition and are accepted for publication on merit. The following synopses are from articles now available in their entirety to ACM members via the Digital Library.*

contributed article

DOI: 10.1145/1995376.1995403

Crossing to the Dark Side: Examining Creators, Outcomes, and Inhibitors of Technostress

Monideepa Tarafdar, Qiang Tu, T.S. Ragu-Nathan, and Bhanu S. Ragu-Nathan

Mike, a Fortune 100 senior-management executive, spends a good part of his annual vacation answering office email messages. He has trouble focusing on his family; he forgets things like dinner plans. Joanne, a university secretary, found it difficult to use a new student-management application. Daunted by the sheer multiplicity of its features, exhausted by repeated crashes, and unhappy at the lack of IT support, she took early retirement. Pat, a purchasing manager, is prompt at answering email and voicemail messages, and has received "responsive employee of the month" awards. But, every time she interrupts whatever she is doing to answer messages, it takes her about 15 minutes to refocus her full attention back to that task before another message comes along and the cycle start again. Paul, a manager at a livestock feed company, uses his 45-minute office commute time to email, text, or call-in instructions from his BlackBerry to fulfill last-minute customer orders, so that his commute is not "wasted." He dangerously juggles his phone while driving.

These vignettes illustrate an interesting and increasingly persistent dichotomy in the way that emerging information systems for work and collaboration are affecting professional users. One aspect of this dichotomy is that aided by workflow applications, mobile computing and communication devices, collaborative software, and computer networks, users can quickly and easily access information, work from anywhere, and share information and insights with colleagues in real time. But these same technologies can make them feel compulsive about being connected, forced to respond to work-related information in real time, trapped in almost habitual multitasking and left with little time to spend on sustained thinking and creative analysis. These latter outcomes constitute the phenomenon of "technostress."

Professionals experience technostress when they cannot adapt to or cope with information technologies in a healthy manner. This article reports on a study of IS users in an effort to understand the phenomenon of technostress, explaining why technostress is created; how it varies across individuals; what its adverse consequences are; and how organizations can reduce them.

contributed article

DOI: 10.1145/1995376.1995404

Calculating and Improving ROI in Software and System Programs

Murray Cantor

Constrained by a limited budget, most enterprises must apply unprecedented business discipline to the business function of software and system delivery across entire software and system life cycles. For this reason, the CIO, CTO, or VP of software or systems development may be under increased scrutiny from the corporate chief finance office (CFO). When conversing with the CFO, money talks, so only one of two sorts of conversations is possible: software and systems as cost center or software and systems as value-creation center. The second is more involved than the first since cost is easily measured and the value of the software and systems under development is difficult to measure. Also, the second conversation entails treating software and systems programs as investments and calculating the return on investment (ROI) along with the investment risk.

Most common development-program and portfolio-management practices in software and systems organizations do not support the second conversation; benefits and risk are usually based on qualitative scores, determined either through team consensus or weighted sum of scores from a questionnaire, while costs are assigned monetary value. Since quantitative and especially monetary measures are more persuasive than qualitative measures, it is no wonder that software and systems are often managed as a cost center, not as a business function contributing to overall enterprise value.

Creating enterprise value often requires innovation, but innovative programs are inherently risky, financially and technically. Almost by definition, innovative programs begin with incomplete information, resulting in uncertainty in both expected program costs and benefits. Reasoning about, justifying, and making trade-offs among programs with different risks and value requires determining the ROI in the programs. Here, I explore how to compute ROI given the inherent uncertainty of innovative programs.

BLOG@CACM

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

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

DOI:10.1145/1995376.1995380 http://cacm.acm.org/blogs/blog-cacm

Jeannette M. Wing @ PCAST; Barbara Liskov Keynote

Jeannette M. Wing discusses her PCAST presentation about the importance of computer science and its impact. Valerie Barr shares highlights from Barbara Liskov's keynote at Grace Hopper.

Jeannette M. Wing "Talking with PCAST"
http://cacm.acm.org/blogs/blog-cacm/98818
Sept. 15, 2010

I was honored to have the opportunity to talk with the President's Council of Advisors on Science and Technology (PCAST) on September 2, 2010, at the Keck Center in Washington, D.C. I opened with a 20-minute presentation, which was followed by a question and answer period. The topic of my session was networking and information technology, since PCAST is doing a review of the Networking and Information Technology Research and Development (NITRD) Program, but I chose to speak more broadly about the importance of computer science and its impact on our economy, society, and other science and engineering disciplines.

I told three stories—The Google Story, Model Checking, and Machine Learning—as a way to illustrate the importance of sustained federal funding of basic research in computer science, the rapid pace of innovation in our field, and the deep scientific contributions we offer besides our obvious technological ones. Using my three drivers of computing framework, Technology-Society-Science, I presented some trends for the future including: Big Data, Cell+Cloud, Cyber-Physical Systems, Socially Intelligent Computing, and emerging computing substrates under Technology; "A^7: Anywhere Anytime Affordable Access to Anything by Anyone Authorized" under Society; and questions like "What is computable?" (see "Five Deep Questions in Computing") under Science.

I emphasized the importance that advances in computer science have in addressing societal grand challenges such as sustainability/energy, health care, transportation, education, and security, thereby also placing our role in the context of the Obama Administration's priorities in science and engineering. I also made some specific recommendations for NITRD. Since most scientists think "high performance computing" when they think of computer science and since NITRD was created from the High-Performance Computing Act of 1991 (before browsers or search engines even existed!), I assumed PCAST understood our role in scientific computing and is cognizant of the trend toward exascale computing. Finally, I reminded PCAST that computer science is part of STEM and argued the importance of learning computer science concepts (aka "computational thinking") at the K–12 level.

The Q&A session was lively, starting off with questions on K–12 computer science education and the use of computing technology for learning. Education was clearly on PCAST's mind since that afternoon they were going to discuss a report they plan to release on K–12 STEM education. Other questions ranged from topics such as (paraphrasing) "From a physics perspective, is there a maximum volume of information we can have?" (a nice challenge question for the theoretical computer science community since I think it begs the question "What is information?") and "What is the seamy underbelly of the optimistic picture [I] painted?" (my answer: cybersecurity and privacy).

Besides trying to give PCAST a sense of computer science as a discipline, three of the most important messages I tried to convey to PCAST: 1) Advances in computer science help accelerate the pace of innovation and discovery in nearly all other fields; 2) Advances in computer science are needed to ad-

dress society's and our nation's grand challenges; and 3) Computer science has a rich intellectual agenda.

My slides are available in .pptx and .pdf formats: (http://www.cs.cmu.edu/afs/cs/usr/wing/www/talks/Wing-Sept-2-2010.pptx) and (http://www.cs.cmu.edu/afs/cs/usr/wing/www/talks/Wing-Sept-2-2010.pdf). Please see the Notes pages of my PowerPoint slides for my transcript.

Valerie Barr
"Barbara Liskov Keynote, Grace Hopper Conference"
http://cacm.acm.org/blogs/blog-cacm/99599
Oct. 2, 2010

Barbara Liskov, Institute Professor at the Massachusetts Institute of Technology (MIT), received the 2008 A.M. Turing Award for her innovations to designing and building computer systems and her achievements in programming language design that have made software more reliable and easier to maintain. Liskov opened her talk by commenting that receiving the Turing Award had given her an opportunity to reflect on her meandering career path and the work she has done.

Liskov grew up in San Francisco in the 1950s. She was interested in math and science, so she took lots of classes, but she didn't talk about it much because it wasn't cool for girls to like math and science. She then went to the University of California, Berkeley, and became a math major, despite being one of very few women in her classes. After her undergraduate work, Liskov didn't feel ready for graduate school, so she moved to Boston and was offered a job as a programmer at the MITRE Corp. She learned FORTRAN, and discovered she really liked programming. After a year, she moved to Harvard and worked on their language translation project. This was during the period of great optimism about artificial intelligence (AI). Liskov maintained a large program written in machine language, which was great training for becoming a computer scientist. Of course, it also gave her a great understanding of bad code, especially since it was self-modifying code.

Liskov eventually decided to go back to school because she wasn't learning fast enough. She went to Stanford, met John McCarthy, boldly asked him for support, and ended up working with him during her graduate studies. She was the only woman in her class, followed by Susan Graham who entered a year later. But it was a very supportive environment. Liskov eventually decided to switch out of AI after finishing her thesis because she had become more interested in computer systems.

Initially, Liskov could not find a job at an academic institution as hiring was done by the old boys' network. She went back to work at MITRE, this time as a researcher. Going to MITRE rather than into academia at that point enabled her to switch technical areas without the added pressure of being a new faculty member who had to think about standing for tenure in a relatively short period of time.

After providing the background information, Liskov talked about her technical work that ultimately led to the Turing Award. Much of her work was motivated by an interest in program methodology and the questions of how programs should be designed and how programs should be structured. So, after receiving the Turing Award, she went back and reread the old literature, discovering anew that there is great material in old papers and that her students were unaware of it. So, she is now pointing people to these papers and encouraging people to read them.

For example, three key papers she cited are:

- Edsger Dijkstra, "Go To Considered Harmful," *Communications of the ACM*, Vol. 11, No. 3, March 1968, pp. 147–148.
- Niklaus Wirth, "Program Development by Stepwise Refinement," *Communications of the ACM*, Vol. 14, No. 4, April 1971, pp. 221–227.
- David Parnas, "Information Distribution Aspects of Design Methodology," *IFIP Congress*, 1971.

In 1972, Liskov published "A Design Methodology for Reliable Software Systems." In this paper she presented the idea of a global state in which each partition owns a part of the global state. Modules completely encapsulate their portion of the global state. This paper was award winning, and Liskov was invited to apply for a position at MIT. She began there in the fall of 1972, one of 10 women out of a faculty of 1,000.

Liskov then began to look at how the partition ideas could be applied to building programs—Could you make programming methodology into something that regular programmers would use? And Liskov began to think about partitions as abstract data types. She looked at material on extensible languages and early material on hierarchical programming structures and inheritance. Her work on abstract data types was codified during the summer of 1973 for a conference in 1974. She basically proposed abstract data types (ADTs) as clusters with encapsulation, polymorphism, static type checking, and exception handling.

In the fall of 1973, Liskov decide to proceed with language design based on ADT work. With three grad students, she designed the CLU language. Her idea was that a programming language would allow her to figure out whether ADTs really work in practice, would allow her to get a precise definition of ADTs, and determine whether ADTs would impact performance. So, CLU has all these mechanisms—cluster, polymorphism, exception handling, and iterators.

Finally, Liskov presented the research challenges of interest to her now:

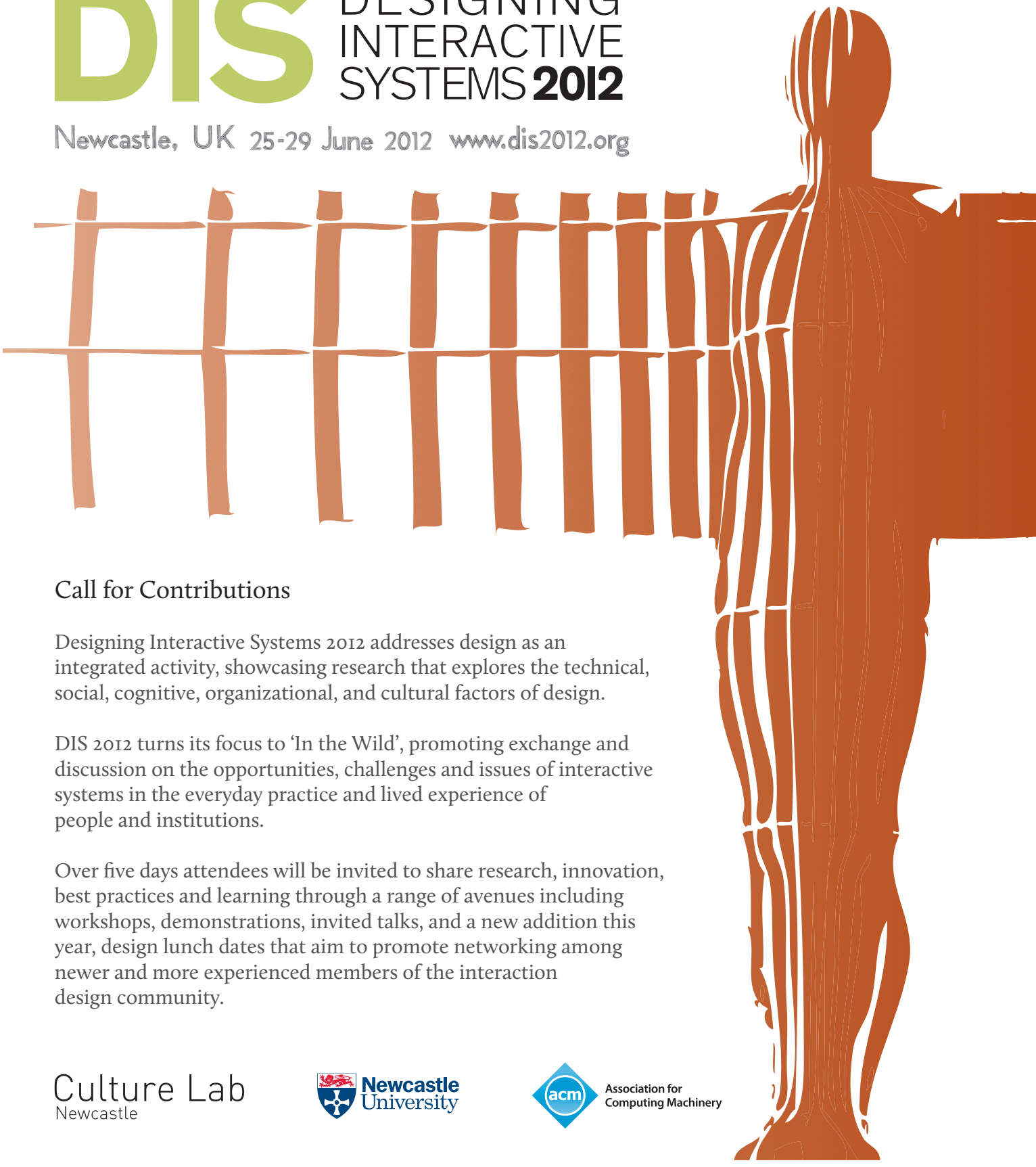
- new abstraction mechanism
- massively parallel computers—much to be explored and learned in this area;
- Internet computation—a rich set of problems; and
- storage and computation, semantics, reliability, availability, and security.

Liskov also discussed the ingredients that have to be in place in order to get an "ah hah" moment. You have to be working on a problem, but also have to be able to have "off time" so that the brain can work on the back burner. Finally, she exhorted the audience not to get too tired because then you aren't productive. C

Jeannette M. Wing is a professor at Carnegie Mellon University. **Valerie Barr** is the chair of the computer science department at Union College.

DIS DESIGNING INTERACTIVE SYSTEMS 2012

Newcastle, UK 25-29 June 2012 www.dis2012.org



Call for Contributions

Designing Interactive Systems 2012 addresses design as an integrated activity, showcasing research that explores the technical, social, cognitive, organizational, and cultural factors of design.

DIS 2012 turns its focus to 'In the Wild', promoting exchange and discussion on the opportunities, challenges and issues of interactive systems in the everyday practice and lived experience of people and institutions.

Over five days attendees will be invited to share research, innovation, best practices and learning through a range of avenues including workshops, demonstrations, invited talks, and a new addition this year, design lunch dates that aim to promote networking among newer and more experienced members of the interaction design community.

Culture Lab Newcastle

Newcastle University

acm Association for Computing Machinery

Submission Deadlines

Full and Short Papers	Workshop Proposals	Doctoral Consortium and Demos
Friday 20th January 2012	Friday 9th December 2011	Wednesday 7th March 2012

Science | DOI:10.1145/1995376.1995382

Kirk L. Kroeker

A Breakthrough in Algorithm Design

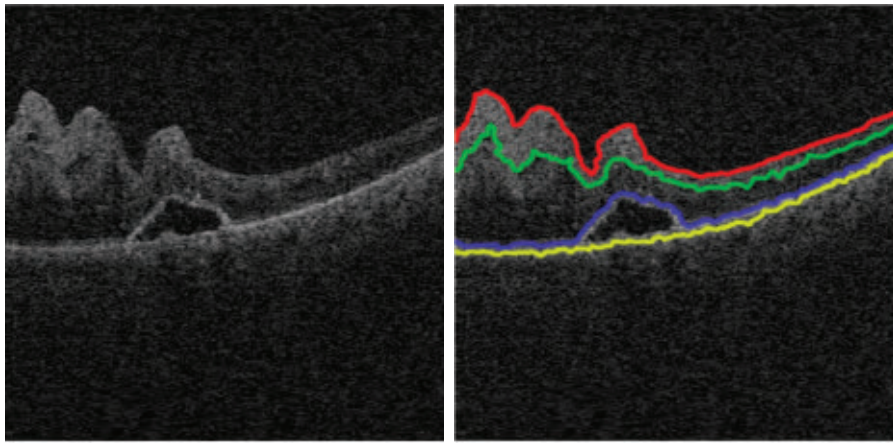
Computer scientists at Carnegie Mellon University have devised an algorithm that might be able to solve a certain class of linear systems much more quickly than today's fastest solvers.

Systems of linear equations are everywhere. They are used in telecommunications, transportation, manufacturing, and many other domains. The algorithms used to solve linear systems must be able to compute solutions to equations involving millions—or sometimes billions—of variables. Because calculating solutions for these systems is time-consuming on even the fastest computers, finding ways to accelerate these computations is an ongoing challenge for algorithm designers. Now, a group of computer scientists at Carnegie Mellon University (CMU) have devised an algorithm they say might be able to solve a certain class of linear systems much more quickly than today's fastest solvers.

The researchers say the algorithm, which applies to a class of problems known as symmetric and diagonally dominant (SDD) systems, not only has practical potential, but also is so fast it might soon be possible for a desktop PC to solve systems with one billion variables in just seconds. "The main point of the new algorithm is that it is guaranteed to work and to work quickly," says Gary L. Miller, a professor of computer science at CMU and a member of the three-person team that developed the new algorithm.

SDD systems, characterized by system matrices in which each diagonal element is larger than the sum of the absolute values of all the other elements in the corresponding row, are used for a wide range of purposes, from online recommendation systems to industrial simulations, materials modeling, and image processing. Algorithms used for this type of linear system fall into two broad classes: direct solvers, such as Gaussian elimination, and iterative solvers. In contrast to direct solvers, which compute exact solutions, iterative solvers produce a series of approximate solutions. Direct methods are usually memory-hungry, a limitation that makes iterative solvers, such as the kind developed by the CMU team, more effective for the large data sets generated by today's applications.

While iterative solvers eventually return satisfactory results, those results typically take a long time to produce because they require calculating many approximations. There have been hundreds of approaches to developing faster iterative solvers, but one method has proved to be the most effective and has become a guid-



A linear system designed to improve the quality of retinal image segmentation through the use of an iterative solver technique called spectral rounding, developed at Carnegie Mellon University and the University of Pittsburgh Medical Center. Conventional segmentation algorithms tend to fail in the presence of retinal abnormalities. On the left is the input image. On the right is the segmented image.

ing principle of sorts in this area of research. The idea is to solve computations on a massive linear system by quickly running computations on a sparser system that in some well-defined algebraic sense is similar to the larger one. The sparser system used to set up these computations for the larger system is called the preconditioner.

Producing the sparse matrix requires zeroing out some of the nonzero entries and increasing the weight of others in the larger matrix. "One key ingredient in the newer algorithms is the judicious use of randomization to determine which entries are zeroed out," explains Miller, who likens the CMU algorithm's process of sparsification to "flipping a biased coin" to determine the fate of an element in the system matrix. This sparsification process is designed to create a representation of the larger system matrix to generate the preconditioner that will guide later computations.

While finding a preconditioner might seem straightforward, finding a good one is not. A reliable method for finding a good preconditioner to accelerate computations on large system matrices is an ongoing challenge in math and computer science. Methods that rely on heuristics, for example, have been effective, but only to a limited extent. "Heuristic solvers are often guided by good intuition," says Richard Peng, a graduate student in the CMU computer science department and a member of the new algorithm team. "However, the critical missing pieces of understanding make them unreliable, especially with the large and complicated systems that we face today."

The Spielman-Teng Solver

The path to developing a more effective method than heuristics for finding a good preconditioner dates to the early 1990s and a series of ideas that suggested viewing SDD systems as combinatorial graphs. Research projects in spectral graph theory and numerical analysis developed these ideas in a string of new theories that, in 2004, emerged as what was widely considered to be a breakthrough proof by Daniel Spielman and Shang-Hua Teng. Spielman and Teng were able to prove that every SDD matrix has a good and discoverable preconditioner.

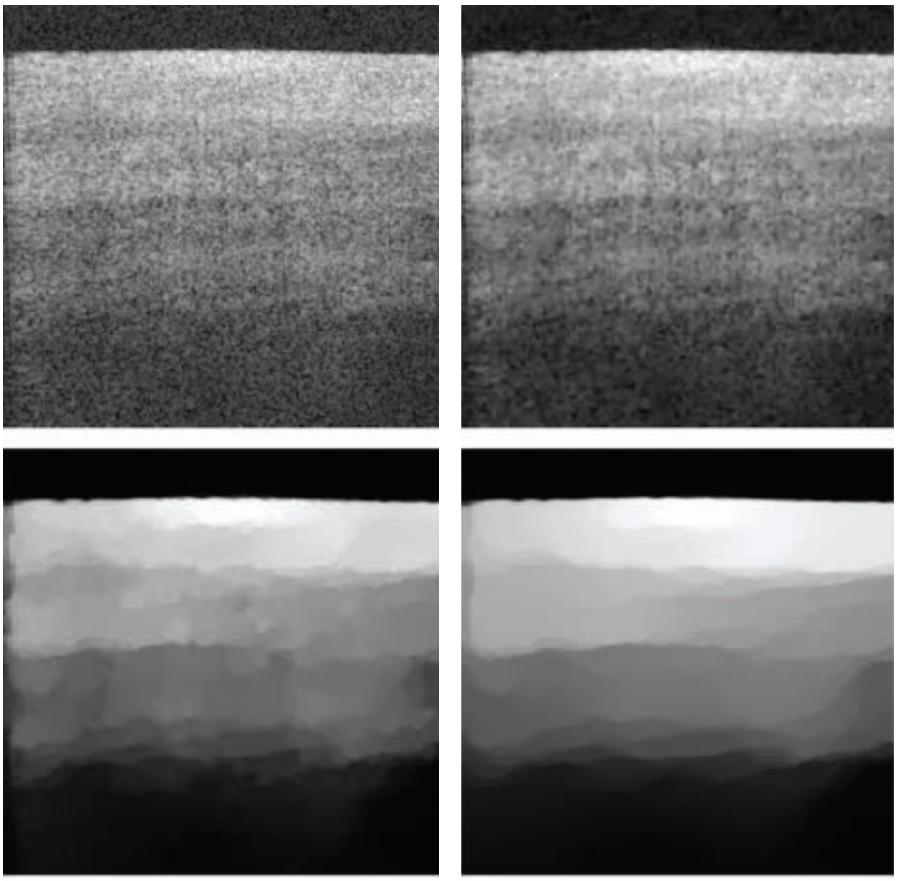
To put the idea into electrical terms, Spielman and Teng showed that for a given electrical network, there will be one that uses fewer resistors while having the same reliability and energy-consumption properties as the original. "The Spielman-Teng solver is asymptotically much faster than everything that was known before," says Peng. "It is faster than previous solvers for all systems larger than a fixed size, and that difference in speed increases as the system becomes larger."

Building on Spielman and Teng's work, the CMU team developed their new algorithm that, from a mathematical point of view, is more concise, taking only five pages to detail instead of 50. "It's nearly optimal," says Ioannis Koutis, the third member of the CMU team and now a professor of computer science at the University of Puerto Rico, Rio Piedras. "We know that we can't do much better, if that's possible at all."

Due to its simplicity, along with its promise of significant speed improvements over earlier algorithms, the new solver made headlines when it was introduced last October at the IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS). With an optimized implementation, the researchers say, the algorithm would be some 10 to 20 times faster than other solvers for current problems. (The technical details of the algorithm are in the FOCS paper; see I. Koutis, G.L. Miller, and R. Peng, "Approaching Optimality for Solving SDD Linear Systems.")

Spielman, a professor of applied mathematics and computer science at Yale University, says the CMU algorithm represents a significant improvement for solving SDD systems. "It is the first algorithm for this problem that is both provably fast in an asymptotic sense and that could be fast in practice," he says.

Spielman explains that when he and Teng created their initial approach to this problem in 2004, their algorithm was guaranteed to find solutions in near-linear time. However, this guaran-



An application designed to improve the quality of optical coherence tomography images for an automated cartilage health-assessment routine. The top images represent the input. The bottom images, enhanced with a linear system designed to smooth the optical coherence tomography images, show striations in the cartilage that are indicative of unhealthy tissue.

"The main point of the new algorithm is that it is guaranteed to work and to work quickly," says Gary L. Miller.

tee was only theoretical and for what he calls "impractically huge" system matrices. The new CMU solver, says Spielman, fixes this problem. "The resulting algorithm is theoretically sound, provably correct, and reasonable in practice," he says. "Now, they just need to optimize their implementations."

Spielman says he expects it will be another decade before understanding of the CMU algorithm fully matures, and that it remains to be seen whether the algorithm will be put to use by developers in the near term. "There are still many reasonable ways of varying their algorithm," Spielman says. "I expect particular applications will benefit from different optimizations."

For the algorithm to be useful in practice, these optimizations must be done so the new algorithm can accommodate the massive sets of data that are the norm for machine-learning problems, materials modeling, image processing, and other applications whose computational results often benefit from better input-data quality. One way to accommodate large amounts of input data while still achieving acceptable computation speed is, of course, to parallelize, but parallelization remains an ongoing challenge in itself for algorithm designers.

"People want better answers, but the algorithms cannot practically handle the larger data sets unless they are fast," says Spielman. "The best we can hope for is algorithms whose running time scales linearly with their input size."

Koutis, Miller, and Peng are working on such optimizations, including parallelization, and are testing their ideas in several practical implementations, such as a new approach to medical imaging developed with David Tolliver, also at CMU, and researchers at the University of Pittsburgh Medical Center. The idea is to use a technique called spectral rounding, driven by SDD systems, to improve the quality of retinal image segmentation. Koutis says that, so far, he and his colleagues have achieved imaging results whose quality, in many cases, has far exceeded that of previous methods.

"This medical imaging application highlights the utility of fast SDD solvers as a primitive operation in building more involved software systems," says Koutis, who predicts that an increasing number of researchers will realize they can formulate some of their problems to benefit from SDD solvers. "We're excited about the possibility that some of our ideas will have a positive impact in the future." C

Further Reading

Koutis, I., Miller, G.L., and Peng, R.
Approaching optimality for solving SDD linear systems, *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, Las Vegas, NV, Oct. 23–26, 2010.

Spielman, D.A. and Teng, S-H.
Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, Chicago, IL, June 13–16, 2004.

Koutis, I., Miller, G.L., and Tolliver, D.
Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing, *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I*, Las Vegas, NV, Nov. 30–Dec. 2, 2009.

Blelloch, G.E., Koutis, I., Miller, G.L., and Tangwongsan, K.
Hierarchical diagonal blocking and precision reduction applied to combinatorial multigrid, *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, New Orleans, LA, Nov. 13–19, 2010.

Teng, S.-H.
The Laplacian paradigm: Emerging algorithms for massive graphs. *Lecture Notes in Computer Science 6108*, Kratochvíl, J., Li, A., Fiala, J., and Kolman, P. (Eds.), Springer-Verlag, Berlin, Germany, 2010.

Based in Los Angeles, **Kirk L. Kroeker** is a freelance editor and writer specializing in science and technology.

Security

Small Companies Targeted

Cybercriminals have expanded their reach beyond large corporations and are increasingly attacking small businesses, according to *The Wall Street Journal*.

While break-ins at Sony and other well-known corporations have recently attracted widespread media attention, the boom in small business hacking has gone largely unnoticed.

Last year, the U.S. Secret Service and Verizon responded to a combined total of 761 data breaches, with 63% of them involving small businesses. In 2009, they responded to 141 data breaches, of which only 27% involved small businesses.

As small companies have grown increasingly reliant on computers in recent years, they have started to store more business-critical information online, including credit card information and other financial data. While large organizations usually employ rigorous security measures to safeguard sensitive data, small businesses' relative lack of IT sophistication has made them easy prey. The wide array of small business systems now in place has created ample opportunities for hackers to develop new techniques for compromising those systems.

In one common ploy, hackers pilfer money by gaining access to companies' online bank account login information. The *Journal* reports that Lease Duckwall of Abilene, KS, saw $63,000 disappear from his company's bank account when a hacker added nine fictitious employees to the company's payroll. By the time Duckwall spotted the discrepancy and notified his bank to freeze the accounts, the hackers had already withdrawn $22,000. To this day, Duckwall has no idea how the hackers gained access to his account information.

Small business hacking is becoming a "prolific problem," Dean Kinsman, a special agent in the Federal Bureau of Investigation's cyberdivision, told the *Journal*. "It's going to get much worse before it gets better."

—Alex Wright

Technology | DOI:10.1145/1995376.1995383 Gary Anthes

Invasion of the Mobile Apps

The market model pioneered by Apple and others is transforming the software world—and has profound implications for software companies and their customers.

SOFTWARE STARTUP SPRING Partners had 40,000 customers in March 2010. Fourteen months later, it had 1.6 million.

The 3,900% increase in business is far from unusual these days. The Charlestown, MA-based company is just one of thousands of mostly small, entrepreneurial firms that have bypassed traditional methods of development, marketing, and distribution in favor of the new online app stores run by Apple, Google, and a few other software and communications giants.

Market-research firm Gartner predicts that mobile app stores will serve 17.7 billion downloads this year, up 116% from an estimated 8.2 billion last year, and that application downloads will soar to 185 billion by 2014. Developers will see more than $15 billion in revenues in 2011 from their mobile online apps, both from download fees and advertising linked to the downloads, according to Gartner.

Today, a stroll through the app stores is a little like visiting an urban flea market, where there are first-rate products but where low-price goods of dubious value abound, and support is practically nonexistent. But suppliers to app stores say the sophistication, utility, and price of the software is increasing, crowding out the junk. As a result, the nascent business model can be seen as a warning to consumer software companies that today sell shrink-wrapped software and whose development cycles are often measured in years.

The financial models and philosophies of the mobile app companies vary widely. But they all cite the same benefits of the online stores: low operating costs for development, marketing, distribution, and support, and low capital requirements for getting into the game. What follows are mini-profiles of three very different companies—Spring Partners, a venture capital-backed startup; Instant Cocoa, a hobby turned two-person startup; and Nuance Communications, an established software company—that claim success at the online app stores.

Apps for the iPhone and other smartphones are soaring in popularity—and creating economic opportunities for countless app developers and software companies.

PHOTOGRAPH BY DANIEL GO

Spring Partners

In mid-2008 Spring Partners landed $5 million in venture capital and in January 2009 launched a free Web-based application called Springpad, a service for "saving anything you want to remember." Things you see online, such as a recipe or a book review, can be cataloged and saved in a personal database. In March 2010, Springpad for the iPhone was launched at the Apple App Store; in May 2010 it appeared at the Google Android Market; in June 2010 Spring Partners had it for the iPad; in December 2010, v2 of the Web app launched in the Google Chrome Web Store; and in May this year Spring Partners announced support for Google Android tablets and offline access through Google's Chrome browser.

All versions of Springpad are free. Spring Partners says its revenue, which it won't divulge, comes when users take action on something they have saved in the Springpad database, such as buying a book. It says 2%–3% of its users generate tiny slices of revenue that way each month.

Although online stores take a cut of the sales price, which is typically 30%, they are otherwise a free distribution mechanism for software developers, and that is perhaps the greatest enabler for small startups. But, says Jeff Janer, cofounder and CEO of Spring Partners, it can take substantial effort to get a product high enough in a store's rankings to keep it from getting lost among the competition. "We reorganized the company last summer so that everything we do, whether product development or business development or marketing, is focused on getting ranked as high as possible. We spend a fair amount of money on public relations."

Springpad was developed as a single service, then ported as a native

application to the various mobile and Web operating systems by using the application programming interfaces (APIs) and software development kits (SDKs) provided by the manufacturers. In addition, the company has built connectors to more than 100 services such as Facebook and Groupon. Eight of the company's 13 employees are developers, including Java programmers. Programmers with mobile application development skills are expensive and in short supply, says Janer.

But companies like Spring Partners can catch a break on the cost of computer resources by going, at least initially, to a cloud service. The company has no data center and uses Amazon's pay-as-you-go cloud service. "For our 1.6 million users," says Janer, "we have one person on staff to run our IT operations." Amazon's big outage in April knocked Spring Partners offline for 30 hours, which was painful, says Janer. Still, he says if Spring Partners goes in-house for processing, it will be based on the economies of scale for a larger company, not the risks of being based in the cloud.

Instant Cocoa

This Seattle-based startup is not so much a company as it is a hobby started by Eric Maland in his spare time while working full time at Google and then Twitter. But Maland, who is currently unemployed, says he's devoting his efforts to taking Instant Cocoa to a new level.

Several years ago while at Google, Maland took an Apple Mac programming class and for fun wrote a desktop application called Wordplay that would solve crossword puzzles. He put it on his Web site, free of charge. When Apple introduced the iPhone, he started developing for iOS. "I didn't actually have an iPhone," he explains. "I just downloaded [Apple's] SDK and wrote my first couple of apps in that." He spent a week writing pTerm, a simple SSH (Secure Shell protocol) client and terminal emulator for the iPhone, and he placed it at the Apple App Store.

Meanwhile, he met a woman named Eliza Block who had written a program for retrieving crossword puzzles and solving them, and the two of them launched Instant Cocoa and published her product, 2 Across, with pTerm, at the iPhone App Store.

"The first week or two the sales were mind boggling," Maland recalls. Four thousand people downloaded pTerm on the first day, and he says he now sells 1,500–2,000 copies a month at $4.99 each. He says 2 Across, at $5.99, has done almost as well. He says he enjoyed two big advantages: There was only one other SSH client at the App Store and it didn't support terminal emulation, and in 2008 he was an early publisher at the App Store.

And, he admits, "what I put out there in a week, I wouldn't put out today, because it would be embarrassing." He is now improving his products and says a month of development work is about right for products like his. Software can be written quickly for this market, he says, because so many

Advertising Revenue Keeps Growing

Many Internet-based companies offer goods and services to consumers for free based on an expectation of associated advertising revenue. The effects of advertising, however, are notoriously difficult to measure. "There's a lot of hype around mobile apps, but not necessarily a lot of dollars as the advertising models haven't been proven yet," says Aaron Masih, director of the mobile developer program at Nuance Communications.

But the model, at least in aggregate, seems promising. Gartner says the advertising revenue from mobile app stores has been modest, but is growing rapidly from $15 million in 2008 to $269 million in 2010 to an estimated $1.5 billion next year.

"There are many proven and profitable online ad models for search, content, referral, and user-generated community sites," says Jeff Janer, cofounder and CEO of Spring Partners. "In our case, where consumers are overtly signaling the products and services they're interested in, the brands view this signal as high-value lead generation and are willing to pay more than standard display ad rates to reach consumers who have expressed intent."

In Memoriam

Robert Morris, 1932–2011

Cryptographer and Unix operating system co-creator Robert Morris died June 26 in Lebanon, NH, at the age of 78 from complications of dementia. Morris was a pioneer in developing operating systems and computer security. He also purportedly played a role in one of the world's first cyberattacks during the 1991 Persian Gulf War.

Morris, who started his career as a researcher for AT&T's Bell Laboratories in the 1960s, initially focused on the development of compilers that could turn programming instructions into machine readable code. Later, he helped develop the Unix OS, which now resides in a growing spate of devices, including Apple's OS X, the iPhone, and Google's Android.

During the 1970s, Morris played an important role in the development of key computer security features, including encryption and password protection. He continued to explore cryptography, eventually unlocking an early German encryption system. From 1986 to 1994, Morris served as chief scientist for the NSA's National Computer Security Center.

Although his role in the 1991 Persian Gulf War remains classified, it has been widely reported that Morris helped launch cyberattacks against key government and military systems in Iraq. Experts have speculated that these attacks destroyed command and control systems before the actual assault was launched against Saddam Hussein's regime.

Morris gained international attention in 1988, after his son, Robert Tappan Morris, then a graduate student in computer science at Cornell University, wrote a computer worm that ultimately froze about 10% of the 50,000 computers then used on the Internet. The code was intended to be innocuous but spread because of a design flaw.

Morris retired in 1994. He is survived by his wife Anne Farlow Morris, a daughter, and two sons.

—Samuel Greengard

conventional items—documentation, help menus, and support—are not demanded by users.

When Block wrote 2 Across, she was a graduate student in philosophy and, although she had a background in math, she had no formal programming experience, Maland notes. "She sat down with the documentation for the SDK and just wrote it in something like six weeks," he says. 2 Across went on to win several notable awards from Apple, including "Best of iTunes: Puzzle Games" and "Staff Favorite."

Instant Cocoa has no formal user support system, says Maland, but he keeps up with email queries and participates in a pTerm Google Group. "A lot of the emails are, 'How do I do this or that,' but somewhere in there is a signal that tells you where you have not done a good job. You get a pretty good sense of what people want in the next version."

Nuance Communications

Unlike Spring Partners and Instant Cocoa, Nuance's history goes back decades. The Burlington, MA-based company has sold its Dragon speech products via retailers and resellers long before app stores existed. But now Nuance has begun offering a few products—most notably Dragon Dictation and Dragon Search—for Apple mobile devices. Nuance also recently began publishing an SDK and APIs to its core speech recognition software so that any developer may easily incorporate Nuance speech into its products.

When Apple introduced the iPhone, Eric Maland started developing for iOS. "I didn't actually have an iPhone," he says. "I just downloaded [Apple's] SDK and wrote my first couple of apps in that."

Nuance charges nothing for its online Dragon apps and derives no revenue from them, at least not directly. According to Aaron Masih, director of the Nuance Mobile Developer Program, the idea is "to prove to the marketplace that speech technology really works and is ready for prime time." A second objective is to learn how people use speech in mobile environments and to learn more about how people speak, especially in foreign languages. Third, it is to increase the brand recognition for the Nuance name and to encourage users to try the Dragon desktop products.

Masih says that as the online app stores become more flooded with offerings, a shakeout will occur, with the products of dubious quality and utility disappearing. The survivors will grow in complexity and will less likely be free, he says.

Maland at Instant Cocoa agrees that mobile apps will gradually move upscale. "It will be harder for the little guy with a good idea to get something in the store," he says. Maland also thinks the model will increasingly invade the desktop realm. "I can't imagine people going to Best Buy to buy software in five years," he says. C

Further Reading

Laudon, K. and Traver, C.G.
E-Commerce 2011 **(7th Ed.), Prentice Hall, Upper Saddle River, NJ, 2010.**

Lee, B.G., Lee, G.H., Shim, Y.H., and Choi, A.
Let developers run the app store by lowering the barrier-to-entry, *International Journal of Electronic Finance 4*, 3, July 2010.

Mahmoud, Q.H., and Popowicz, P.
Toward a framework for the discovery and acquisition of mobile applications, *Proceedings of the 2010 Ninth International Conference on Mobile Business*, Athens, Greece, June 27–29, 2010.

Stark, J.
***Building iPhone Apps with HTML, CSS, and JavaScript*, O'Reilly Media, Sebastopol, CA, 2010.**

Yarmosh, K.
***App Savvy: Turning Ideas Into iPad and iPhone Apps Customers Really Want*, O'Reilly Media, Sebastopol, CA, 2010.**

Gary Anthes is a technology writer and editor based in Arlington, VA.

ACM Member News

To understand and use large datasets, science must continue to make advances in the study of hybrid algorithms that combine symbolic and numeric computations, says Zhi Lihong, professor at the Academy of Mathematics and System Sciences at the Chinese Academy of Sciences.

One of leading researchers in this field, Zhi recently won a Chinese Female Scientists Award earlier this year. While her research may sound esoteric to the general public, Zhi stresses that her work means little if hybrid symbolic-numeric computations are not applied to real-world purposes.

"Hybrid symbolic-numeric computation is becoming more and more important in solving problems in various areas of engineering, robotics, biology, and signal theory," Zhi says. "It is also important in information technology, cryptology, coding theory, and other uses. These require exact and/or certified algorithmic solutions."

That is the point behind the hybrid approach, to get the best of both worlds by taking advantage of the speedier numerical approach with the less error-prone symbolic methods.

This has essentially served as Zhi's career focus. As a graduate student, she sought to develop a new computer algebra system for China, a noble but unsuccessful effort that helped shape Zhi's determination to keep "real" application value foremost in her work.

Zhi takes pride in contributing to the elevation of computer science in China. In previous decades, she says, few Chinese papers were accepted by the International Symposium on Symbolic and Algebraic Computation. In 2011, seven papers from Zhi's lab were accepted. This is another sign, she says, that Chinese research is now on par with that from Europe, the U.S., or other countries.

—Dennis McCafferty

Society | DOI:10.1145/1995376.1995384 Neil Savage

Remaking American Medicine

Developing an IT ecosystem for health could improve —and transform—the practice of medicine.

In these days when so much of life seems to take place on a Web site or over a smartphone, health care is still remarkably lacking when it comes to information technology. Of course billing is done mostly by computer, and in the past few years the electronic writing of prescriptions has soared. But most medical records are still on paper, and even those in digital form are not easily shared between doctors or readily accessible to patients. Data that could aid in understanding a patient—activity patterns or dietary habits—aren't captured. Patterns that might indicate a problem with a drug or suggest a better method of treatment aren't noticed.

A special commission, the U.S. President's Council of Advisors on Science and Technology (PCAST), issued a report last December calling for the creation of an information technology infrastructure for health care in the U.S. Such an IT ecosystem starts with the widespread adoption of electronic health records. But it could go beyond that to devices that collect data about how people live their lives or offer them feedback for making healthy choices. It could include individual databases that gather information relevant to health from a wide variety of sources, and collections of aggregated, anonymized data to aid public-health decisions or supplement clinical trials.

Converting health records to electronic form is a major federal goal. The stimulus package of 2009 provides at least $20 billion over the next five years to promote the adoption of electronic health records, with doctors and hospitals qualifying for extra Medicare and Medicaid payments if they make "meaningful use" of such records. The PCAST report found that nearly 80% of doctors lack even rudimentary digital records. "Of those who do use electronic systems, most do not make full use of their potential functionality," the report states. "The sharing of health information electronically remains the exception rather than the rule."

With Health Buddy, a patient's medical condition can be monitored on a continuous basis without requiring visits to a physician or hospital.

The report recommends that the government promote a universal exchange language for health-care data, based on metadata-tagged elements. Done right, that would allow large organizations to keep their current systems but share data with others that are now incompatible. It would also let innovators create new programs that could run on top of those systems, as well as new mobile apps for consumers that could feed data into a personal health record.

Shareable, accessible digital records could improve the quality of care in a number of ways. If a patient from Boston, for instance, is rushed to an emergency room in Seattle, doctors could immediately find out her allergies, what medications she's on, or a recent surgery that might be contributing to her medical condition. A computer might alert a doctor to potential drug interactions, or send a reminder to follow-up on a lab test. Gordon Schiff, associate professor of medicine at Harvard Medical School and a doctor at Boston's Brigham and Women's Hospital, envisions something like a wiki, "where you sort of continuously evolve a description of a patient. You don't have to start from scratch every time."

Such a system has the potential to reduce diagnostic errors, argues Schiff, by providing a more thorough

PHOTOGRAPH COURTESY OF ROBERT BOSCH HEALTHCARE

patient history, highlighting potential problems, and making it easier for doctors to share their diagnostic thinking. Although he doesn't think it will happen soon, someday computers may even guide doctors toward a diagnosis. Schiff has invited the creators of IBM's Watson, the machine that beat a duo of *Jeopardy* champions earlier this year, to address that possibility at the Diagnostic Error in Medicine conference he's co-chairing in October.

But health information need not be limited to doctor's visits and lab tests. A second PCAST report, "Designing a Digital Future," focusing on networking and information technology, was released a week after the health IT report. It envisions a more comprehensive, lifelong record that includes not only treatment history but also a genetic profile, psychological characteristics, behavior patterns, and exposures to risks that might be relevant to health. While such a record could benefit individual patients, it could provide even greater value when stripped of personally identifying information, combined with similar records, and subjected to data mining algorithms.

It would, for instance, create a sort of extended clinical trial for approved drugs, says Susan Graham, a computer science professor at the University of California, Berkeley, and a member of the report's working group. Today's drug trials stop with the approval of a medication, "yet while people are taking these drugs there's an accumulation of experience about what the side effects are and what the potential benefits are," Graham says. The healthcare group Kaiser Permanente has already demonstrated such a benefit; electronic records for its 8.6 million members helped identify the link between the painkiller Vioxx and an increased risk of heart attacks.

With an entire nation's health records at their disposal, computers might also find early warnings of epidemics or identify which treatment approaches work best.

With an entire nation's health records at their disposal, computers might also find early warnings of epidemics or identify which treatment approaches work best. Graham points out that only major diseases that affect millions of people tend to be studied. A huge database could provide valuable insights into less common disorders. "It's only possible if all of the information on which that kind of insight is based is, number one, electronic, and number two, available," she says.

At-Home Monitoring

Health records could also be fed by devices that collect information about people as they go about their lives. The U.S. Veterans Administration (VA) system already uses the Health Buddy, an electronic device that plugs into a home phone line or Ethernet socket. Each day patients answer a series of questions tailored to their particular medical conditions, asking, for example, whether they have taken their medications or about their glucose levels. Answers are sent to the VA and flagged if they show warning signs.

"Versions of that will be in every home, or at least every home where there's a health condition that could be supported by that," says Molly Joel

Predictions

Ten Disruptive Technologies

The next decade will bring 10 technological changes that will transform the world, says Dave Evans, Cisco's chief futurist. In his opinion, they are:

▸ *The Internet Of Things.* Evans predicts the number of Internet-connected "things" will reach 50 billion—more than six devices for every person on Earth—by 2020.

▸ *The Zettaflood Is Coming.* This year the world is creating 1.2 zettabytes of unique data, mainly as a result of high-definition video. Evans expects 91% of Internet data will be video by 2015.

▸ *Wisdom Of The Cloud.* Evans says that, by 2020, one-third of all data will live in the cloud. "Already, the cloud is powerful enough to help us communicate through real-time language translation," he notes.

▸ *The Next Net.* Evans describes his home as an example of the speed of network improvements. Today he has 38 always-on connections and more than 50Mbps of bandwidth. By 2021, he expects the speed to his home will increase by 3 million times.

▸ *The World Is Flat And So Is Your Technology.* With always-on connectivity, social networking has the power to change cultures, as with the Egyptian revolution. A smaller world also means faster information dissemination. "The dissemination and consumption of events are going from 'near time' to 'real time,' " he says.

▸ *The Power Of Power.* Evans believes that solar alone can meet the world's energy needs. To address today's global demand for energy, 25 solar super sites—each consisting of 36 square miles—could be erected in just three years.

▸ *It's All About You.* More items will move from physical to virtual. Today, we download e-books and movies rather than buy books and DVDs. A technology called 3D printing will allow us to instantly manufacture many physical items.

▸ *The Next Dimension.* Virtual humans will be added to the work force. By 2025, says Evans, the robot population will surpass the number of humans in the developed world. By 2035, robots could completely replace humans in the workforce.

▸ *Another Family Tree.* In the next 10 years, Evans believes medical technologies will grow vastly more sophisticated as computing power becomes available in smaller forms. Devices such as nanobots and the ability to grow replacement organs from our own tissues will be the norm.

▸ *You...Only Better.* Taking the medical technology idea to the next level, healthy humans will be given the tools to augment themselves. While their early use will be to repair unhealthy tissue or fix the consequences of brain injury, eventually designer enhancements will be available.

—*Paul Hyman*

Coye, head of the UCLA Innovates Healthcare initiative at the University of California, Los Angeles. "We will know what your blood pressure is every morning at 8 o'clock, or how it varies during the day, instead of every six or eight months when you go to the doctor."

Such increased monitoring could catch potential problems earlier, perhaps leading to more effective treatment or outright prevention of some conditions. It could also reduce costs. The VA estimates its in-home monitoring saves thousands of dollars per patient by reducing doctors' visits and nursing home care.

The growth of the "Internet of Things," in which now-discrete devices are networked, could provide both monitoring and feedback, suggests Isaac Kohane, professor of pediatrics and of health sciences and technology at Harvard Medical School and director of informatics at Boston's Children's Hospital. Your refrigerator, for instance, might offer suggestions to help you adhere to your diet, or the motion sensor in a gaming system could be used to guide physical therapy. "Pretty much everything we're doing today could have a sensor," Kohane says. "Your scale could have an IP address."

With electronic patient-monitoring devices, "we will know what your blood pressure is every morning at 8 o'clock, or how it varies during the day, instead of every six or eight months when you go to the doctor," notes Molly Joel Coye.

There's already a package of sensors that many people carry around with them every day: their smartphone. "People are walking around with devices that make it much easier to capture in-the-moment data," says Deborah Estrin, director of the Center for Embedded Network Sensing at UCLA. Analyzing patterns of a smartphone's GPS traces could reveal changes in a person's behavior, perhaps signaling, for example, a bout of depression or an increased risk of suicide.

Estrin is a proponent of developing an open architecture for mHealth, the practice of using mobile communication devices for monitoring patient health. A patient telling a cellphone app about symptoms or pain levels will be more accurate about how he's feeling right now than trying to recall these details in a visit to the clinic days or weeks later, she says. Existing apps already help people keep track of diet and exercise, for instance, but if they could feed the information back into a permanent health record available to the doctor, they could offer much greater benefit.

Protecting Patient Privacy

If all this is to work, strong privacy protections will be important. Latanya Sweeney, professor of computer science at Carnegie Mellon University, says data should be segmented and in the control of the patient. This way, a patient could share information about an HIV test only with her primary-care doctor while letting everybody know about her allergies. There also should be a way to track who sees patient data to help prevent abuse, Sweeney says. If a bank, for instance, is buying information about a customer's cancer risk and using it to adjust their credit scores, a patient ought to know. Sweeney worries that a lack of privacy incentives in the health-care initiative will produce a backlash.

Even with patient names stripped away, it's possible to cross-correlate data and expose private information, the way some researchers have used public data to deduce an individual's Social Security number. On the other hand, Graham points out, preventing all such correlations could mean missing connections and patterns that might improve patients' health. Reaching the right level of data protection, Graham says, is both a technical challenge and a policy issue.

Sweeney sees a lot of value in developing an IT ecosystem, but is skeptical about how quickly it will develop. "For me, the excitement is in the sharing level, but we're not there," she says. "We're not apt to get there in 2015."

Computer scientists will have to work with doctors to figure out what is technically feasible and how IT can fit into the practice of medicine, says Graham. The capture of information in clinical settings has to fit into the workflow, so providers don't find it burdensome. And they will have to guide the policy makers who will make the regulatory and financial decisions.

"It really needs to be interdisciplinary," Graham says. "This is not just a computer science topic." C

Further Reading

Coye, M.J., Haselkorn, A, and DeMello, S.
Remote patient management: Technology-enabled innovation and evolving business models for chronic disease care, *Health Affairs 28*, 1, Jan.–Feb. 2009.

Graham S., Estrin D., Horvitz, E., Kohane, I, Mynatt, E., and Sim, I.
Information technology research challenges for healthcare: From discovery to delivery, Computing Community Consortium, May 25, 2010.

PCAST
***Realizing the Full Potential of Health Information Technology to Improve Healthcare for Americans: The Path Forward*, The White House, Office of Science and Technology Policy, Dec. 8, 2010.**

PCAST
***Designing a Digital Future: Federally Funded Research and Development in Networking and Information Technology*, The White House, Office of Science and Technology Policy, Dec. 16, 2010.**

Schiff, G.D. and Bates, D.W.
Can electronic clinical documentation help prevent diagnostic errors? *New England Journal of Medicine 362*, 12, March 25, 2010.

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

DOI:10.1145/1995376.1995385

Rebecca Tushnet

Law and Technology
Remix Nation

Assessing the threat the anticircumvention provisions of the Digital Millennium Copyright Act pose for fair use.

IMAGINE A PERSON who decides to make a *Downfall* video, using a scene of Hitler receiving bad news to mock some current event. Assuming this is her first attempt at a remix, she might do some searches to figure out the best way to go about it. She will easily find guides online showing her how to use various software programs—many of the alternatives are free—to rip clips from a DVD and import them to her video editing program to create her remix.

Asked about copyright issues, she might say that what she is doing is a fair use allowed by copyright even without the owner's permission: it is noncommercial, uses only a portion of the movie she is remixing, offers new meaning that cannot be found in the original, and does not interfere with any market the copyright owner wants to participate in. And she would be right.

The only problem is that, until recently (and potentially starting again in 2012), U.S. law made her *method* of remixing illegal under the anticircumvention provisions of the Digital Millennium Copyright Act (DMCA). Circumventing the "access controls" of a commercial DVD—the code that tells it to work only on a licensed player that does not allow any copying, no matter how minimal—was unlawful regardless of whether the purpose was to make a fair use. To make matters worse, the DMCA applied only to particular ways of getting those fair use clips: someone who set up a separate camera to film the screen on which the DVD was playing would not be violating the DMCA, even if he filmed the whole movie. Though the film studios touted this as an alternative to circumvention, they also pressured the federal government and many states to enact laws making using a camcorder in a theater illegal, so that one woman was jailed for two days for filming her sister's birthday party, which involved a trip to see the blockbuster *The Twilight Saga: New Moon*.[1]

The Digital Millennium Copyright Act created a trap for the unwary.

Moreover, the Copyright Office has also stated that a person who used screen capture software to record a DVD's output as it played would not be subject to DMCA liability (though major copyright owners are not prepared to agree with that conclusion—they say that using screen capture *might* violate the DMCA). Under this bizarre system, only using the standard, widely available programs like DVD Decrypter for making clips would break the law, even if the output of the camcorder version and the screen capture version looked the same as the decrypted version.

The Digital Literacy Test and the Digital Poll Tax

The DMCA created a trap for the unwary. Indeed, someone who downloaded a full unencrypted movie from an unauthorized source might be better off, legally speaking, than someone who circumvented the controls on a DVD she had paid for to get 30 seconds' worth of clips, because at least the former would be able to argue that fair use justified her conduct. Historically, the literacy test required prospective voters to interpret an often arcane

ILLUSTRATION BY GLUEKIT. PHOTOGRAPH BY LEFT EYED PHOTOGRAPHY / SHUTTERSTOCK.COM

provision of the law, asking questions irrelevant to the capacity to vote. Under the DMCA, fair users needed to understand that a digital file created in one way is illegal, while a digital file of the same movie created in another way is legal. Yet the issue of how to define and identify a circumvention technology has no relation to artistry or to fair use—nor even to deterring copyright infringement, given the alternatives discussed previously.

Then the digital poll tax kicked in: remixers were supposed to use a camcorder or screen capture software, both of which often produce degraded results. We do not usually tell artists they have to use bad materials to make their creative works, even in the name of protecting previous artists. Visual quality can be especially vital to cultural critics. If pop culture has luscious imagery, and critics have to speak in hard-to-watch forms, their already-marginal work is further hampered by looking incompetent. Ironically, camcorders and screen captures can work for making first-generation copies that are good enough to watch—and thus passably satisfying for true pirates—but not good enough to survive the multiple generations of digital manipulation and editing often involved in a remix, since each iteration involves some image degradation just as it would in analog editing. For example, screen capture tends to produce dropped frames, making time editing all but impossible. Thus, the DMCA hits hardest at transformative, critical uses by people interested in conforming with the law, and does the least damage to pure copiers.

The poll tax also came in the literal financial expense of using the camcorder setup recommended by major copyright owners for making clips: hundreds of dollars on a separate camera, a tripod for stability, a perfectly dark room to prevent light pollution, and a large TV. In combination, the qualitative and financial burdens imposed by compliance with anticircumvention law erected profound barriers to effective use of video clips, for anyone who managed to learn about them.

The DMCA hits hardest at transformative, critical uses by people interested in conforming with the law.

None of this was difficult to predict when the DMCA was enacted, and from the beginning critics denounced its effects on fair use. Courts, however, considered the structural disadvantages created by the DMCA too hypothetical and general to justify any limits on the scope of the law.[a]

Rulemaking as Safety Valve

This legal regime had particularly damaging effects on members of marginalized groups who are already likely to have limited resources and to be uncertain about expressing themselves. There is a narrow avenue for relief: the DMCA provides for a triennial rulemaking procedure allowing the Librarian of Congress to create temporary exemptions to the ban on circumventing access controls where

a *Universal City Studios, Inc. v. Corley*, 273 F.3d 429 (2nd Cir. 2001).

the ban is harming noninfringing uses of copyrighted works. Although the Librarian initially accepted only extremely limited proposals, leaving most fair uses unprotected, in 2006 it allowed media studies and film professors to circumvent DVD encryption to use clips in teaching. Building on this exemption, representatives from the Organization for Transformative Works (OTW)—on whose legal committee I serve—testified in the most recent DMCA proceedings on behalf of noncommercial remix artists, supporting an exemption proposed by the Electronic Frontier Foundation.

Fair use remixes abound online, and we submitted many examples. For nonlawyers, American University's Center for Social Media has developed a set of best practices for fair use in an online video, offering comprehensible rules that require good judgment, but not a lawyer's services, to apply.[2]

One reason so many laypeople are dismissive of copyright law is because it is counterintuitive and arcane, resulting in seeming unfairness and futility; the anticircumvention provisions are a good example of that. While they encourage disrespect from some people, incomprehensible rules also deter risk-averse remixers who are vaguely aware of the DMCA from making fair uses. Even the ones who continue may find themselves unable to assert fair use defenses for fear of DMCA liability. Some remixers have received takedown notices and wanted to make fair use claims so their works could be restored, but decided they could not because they were unsure about the method they used to capture the clips.

Hiding the (Legal) Wiring

The solution, as a British government report put it, is to "hid[e] the wiring"—to simplify copyright law so that it comes into better alignment with ordinary logic.[4] Fortunately, the Copyright Office agreed with these arguments, at least in part, in its most recent rulemaking. The rulemaking allowed circumvention to access content on DVDs "when circumvention is accomplished solely in order to accomplish the incorporation of short portions of motion pictures into new works for the purpose of criticism or comment, and where the person engaging in circumvention believes and has reasonable grounds for believing that circumvention is necessary to fulfill the purpose of the use" for certain educational uses by professors and film students, documentary filmmaking, and noncommercial videos.[5] Notably, that last option not only covers most YouTube remixes, but also most educational uses, even those not allowed by the first, limited educational exemption. As long as they reasonably believe that circumvention is necessary—and given the expense and flaws of the alternatives, it will routinely be necessary—noncommercial video artists can remix at will.

There are several lessons from the battle to keep fair use from being eliminated via technological means.

The creativity of remix culture comes from many far-flung individuals, some of whom invent or reinvent remix for themselves without even knowing about other remixers and others of whom work within existing communities, aware in varying degrees of the artistic traditions they are updating, continuing, and disrupting. But when it comes to dealing with the effects of law on creativity, individual creators need organized representation. Otherwise, as copyright policymaking has repeatedly shown, their interests will simply be ignored. Henry Jenkins, a leading scholar on the interaction of corporate and individual creativity in the digital age, argues that media fandom, from which many remixes derive, is "the experimental prototype, the testing ground for the way media and culture industries are going to operate in the future."[3] If so, then without further activism, "testing ground" might be a far-too-apt metaphor, with the copyright industries trying out their best new heavy ordnance—technological and legal—on individual remixers.

There are several lessons from the battle to keep fair use from being eliminated via technological means. The rulemaking process of the DMCA is far from a panacea. Among other things, exemptions will be lost if advocates do not show up to argue for them every three years, or if the Copyright Office changes its mind about the value of particular uses. Also, distribution of circumvention technology remains unlawful, even though people entitled to an exemption are unlikely to be able to accomplish circumvention on their own and even though the copyright industries admit that this ban has failed. Regardless, since it is easy to find circumvention technology and not unlawful to possess it, people entitled to circumvent can easily find the means to do so, but this remaining barrier is a reminder of the costs of poorly thought-out lawmaking.

The U.S. has successfully pressured many of its trading partners to adopt U.S.-style anticircumvention provisions, generally without U.S.-style limitations and exemptions. The U.S. experience with DMCA overkill demonstrates that the DMCA as written is not right for anyone, and that other countries should be wary of copying a law that suppresses artists and educators. Laws will be made with or without the input of those who understand what technology enables (and threatens); the challenge is to ensure that we do not, in aiming at commercial pirates, hit the fans and critics who are trying to participate in a cultural conversation instead. C

References

1. Bell, A. Charges against accused "The Twilight Saga: New Moon" "Pirate" dropped, examiner.com, (Dec. 11, 2009); http://www.examiner.com/x-4908-Twilight-Examiner~y2009m12d11-Charges-against-accused--The-Twilight-Saga-New-Moon-pirate-dropped
2. Center for Social Media. Code of Best Practices in Fair Use for Online Video; http://www.centerforsocialmedia.org/sites/default/files/online_best_practices_in_fair_use.pdf
3. Jenkins, H. Afterword: The Future of Fandom. In J. Gray, C. Sandvoss and C.L. Harrington, Eds., *Fandom: Identities and Communities in a Mediated World*. New York University Press, New York, 2007, 357–364.
4. U.K. Intellectual Prop. Office, © The Way Ahead: A Strategy for Copyright in the Digital Age (2009); http://www.ipo.gov.uk/c-strategy-digitalage.pdf
5. U.S. Copyright Office, Rulemaking on Exemptions from Prohibition on Circumvention of Technological Measures that Control Access to Copyrighted Works; http://www.copyright.gov/1201/

Rebecca Tushnet (rlt26@law.georgetown.edu) is a law professor at the Georgetown University Law Center, Washington, D.C.

DOI:10.1145/1995376.1995386

Martin Campbell-Kelly

Historical Reflections

In Praise of 'Wilkes, Wheeler, and Gill'

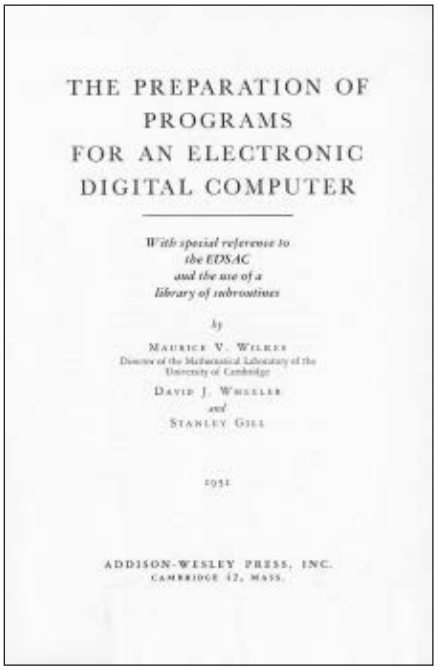
Reflections on the first textbook on programming.

Sixty years ago, in spring 1951, Maurice Wilkes, David Wheeler, and Stanley Gill produced the first textbook on programming: *The Preparation of Programs for an Electronic Digital Computer*.[2] It was a publication that spearheaded the software revolution.

The guiding light behind the book was Maurice Wilkes, who died last November at the great age of 97 years old. He was best known as head of the computer laboratory at Cambridge University, though he did a great deal more. His interest in computing long predated the modern digital computer. In 1937, he became assistant director of a newly established computing laboratory at Cambridge University, but development was cut short when Britain declared war on Germany in September 1939. The computing facilities were taken over by the military and Wilkes left Cambridge to join the scientific war effort. He worked on radar and operations research, which turned out to be an ideal background for the dawning of the computer age. In 1946, he returned to Cambridge with the mission of rebuilding the computer laboratory.

Maurice Wilkes and EDSAC

In May 1946, Wilkes got his first glimpse of John von Neumann's famous *EDVAC Report* of June 1945, which laid out the design of the electronic stored program computer. It was brought to the laboratory by a visitor who took it away the following day. Wilkes had no copying facilities, so he stayed up half the night reading it. He recognized it at once as "the real thing" and never looked back. The following summer he attended the summer school organized by the Moore School of Electrical Engineering, University of Pennsylvania, where the designers of the ENIAC and EDVAC unveiled the inner workings of the new electronic computers. Returning to England on the *Queen Mary*, Wilkes began to sketch out the design of a machine he called the EDSAC, for Electronic Delay Storage Automatic Calculator. The name was consciously chosen in homage to the EDVAC, on which it was directly based.



THE PREPARATION OF PROGRAMS FOR AN ELECTRONIC DIGITAL COMPUTER

With special reference to the EDSAC and the use of a library of subroutines

by

Maurice V. Wilkes
Director of the Mathematical Laboratory of the University of Cambridge

David J. Wheeler
and
Stanley Gill

1951

ADDISON-WESLEY PRESS, INC.
CAMBRIDGE 42, MASS.

The title page from "WWG."

From the beginning, Wilkes was more interested in having a computer for practical use than in having one of the highest technological performance. To this end he kept the EDSAC simple—conservative in electronics and straightforward in design. The machine sprang to life on May 6, 1949. It was quickly put into operational use and it was the first computer in the world to provide a practical computing service. EDSAC was Cambridge University's principal computing resource until it was replaced by EDSAC 2 in 1958.

Within about six weeks, Wilkes made one of the most far-reaching discoveries of the computer age: that getting programs right was more difficult than it looked. As he subsequently recalled, it was while he was developing his very first application program that "the realization came over me with full force that a good part of the remainder of my life was going to be spent in finding the errors in my own programs." Wilkes decided that making the programming process less error prone would be a good project for his research student David Wheeler.

David Wheeler

Wheeler was a brilliant student. He had graduated in mathematics in 1948 as a "wrangler"—the University's argot for the top mathematicians of a cohort. While he was an undergraduate his interest in computing was piqued by the EDSAC that was then under construction and, in his own words, he pestered

The EDSAC subroutine library was kept in the steel cabinet on the left in the image. Library tapes were copied mechanically onto the user's program tape and then returned to the cabinet.

Wilkes to become involved. Wilkes put a soldering iron in his hand and Wheeler was hooked. In October 1948, he became Wilkes's research student.

Like computer groups everywhere, the laboratory had a copy of the *Planning and Coding* reports produced in 1947–1948 by Herman Goldstine and John von Neumann at the Institute for Advanced Study, Princeton University.[1] These seminal reports contained many key ideas, including that of a subroutine library. Most programs, it was observed, made use of common operations (such as computing a square root or integrating a differential equation). Using library subroutines not only reduced the amount of original code in a program but also cut down on the number of errors.

Goldstine and von Neumann did not come up with a particularly effective way of incorporating subroutines in a program, however. This enabled Wheeler to make his mark by devising a brilliant, practical solution known as the "initial orders." The initial orders were something of a cross between a bootstrap loader and an assembly routine. Programs, which were punched on paper tape, consisted of a main program and a sequence of subroutines copied from the subroutine library. The library was kept in a small steel cabinet containing master copies of the different subroutines (there were eventually over 100 different subroutines.) Subroutines helped, but there were still plenty of errors in programs.

In order to debug a program, a user would sit at the EDSAC's control desk and obey the program manually one instruction at a time, observing the state of the memory on a monitor display. This process—known at Cambridge as "peeping"—was time consuming and prohibitively wasteful for such a scarce resource. Wheeler's solution, which was evocatively called a "post-mortem routine," was later known more prosaically as a memory dump. The post-mortem routine consisted of a small program that could be loaded into the top end of memory in the event that a program did not run as expected, which printed out a region of memory so it could be studied at leisure away from the machine.

Stanley Gill

A different debugging idea was invented by Stanley Gill, another of Wilkes' research students. This was the so-called interpretive checking routine. If a program was difficult to debug, then the program would be executed, not by the hardware circuits of the EDSAC, but by Gill's interpretive routine. This behaved exactly like the real machine except that it printed diagnostic information while doing so. This idea was adopted, or reinvented, at many places and became known as a program trace. It earned Gill his place as one of the triumvirate of authors of *The Preparation of Programs*.

With these programming aids it was no longer necessary for programmers to run their own programs; a machine operator took over and greatly increased

the productivity of the machine. In the summer of 1950, the group wrote a report (with British spelling) *The Preparation of Programmes for the EDSAC*. Wilkes sent copies to everyone he thought might be interested, so that by the fall most computer groups had access to the report. Eventually the report found its way to the publisher Addison-Wesley in Cambridge, MA, who printed it in book form essentially without change, apart from some introductory front matter. This first textbook on programming, *The Preparation of Programs for an Electronic Digital Computer* was often known simply as "WWG" after the initials of its authors, Wilkes, Wheeler, and Gill.

WWG was highly influential and was essential reading for everyone thinking about programming in the early 1950s. For example, Wheeler's scheme of organizing subroutines was picked up by IBM and used in its first computer, the model 701 launched in 1953. Japan's first electronic computer TAC (Tokyo Automatic Computer), a joint development between Toshiba and Tokyo University, was designed with an identical instruction code to the EDSAC so that it could use the subroutine library printed in WWG. There was no direct contact with Cambridge University, and Wilkes did not learn of the TAC until many years later. If one goes back far enough, it seems that most programming systems have a distant ancestry in WWG.

And what became of Wilkes, Wheeler, and Gill themselves? Wilkes of course needs little explanation. He remained at the computer laboratory until his retirement in 1980. During that time he served both academic and public roles with distinction. His best-known scientific contribution was the invention of microprogramming. In the public sphere he was the founding president of the British Computer Society (a sister organization of ACM) and a prime mover behind IFIP. His many honors included the ACM Turing Award in 1967, the Kyoto Prize in 1992, and he was conferred with a knighthood in 2000.

Wheeler was a man of a completely different mold to Wilkes, and that is perhaps why they complemented one another so well. Wheeler, too, remained at the computer laboratory all his life, apart from occasional sabbatical leaves at other universities. Alongside Wilkes, he was a cornerstone of the laboratory. Yet it is difficult to think of anyone less career-minded than Wheeler. He was notoriously reluctant to publish; his motivation came from the research itself. When he was elected to the Royal Society in 1981, he must have had one of the shortest publication lists ever. He retired in 1994, but as an emeritus professor he still came in to the university most days; he died in 2004, while riding his bicycle to the laboratory on a cold day.

If one goes back far enough, it seems that most programming systems have a distant ancestry in WWG.

Today, not many people have heard of Stanley Gill because he died at a relatively young age. Gill was a mover and shaker, and would never have been content to spend his working life in one institution. In 1957, he left the laboratory to take on the important role of head of computer research at Ferranti, then one of Britain's primary manufacturers of mainframe computers. He next became professor of computing science at Imperial College, University of London. In 1970, he was appointed to a senior position with PA Management Consultants, a role of massive opportunity because computing services were just beginning to take off in the U.K. Gill should have been in the thick of it, helping to reshape the computer industry. That promise was cut tragically short when he died in 1975. C

References

1. Goldstine, H.G. and von Neumann, J. *Planning and Coding of Problems for an Electronic Computing Instrument*, 1947–1948.
2. Wilkes, M.V., Wheeler, D.J., and Gill, S. *The Preparation of Programs for an Electronic Digital Computer*, Addison-Wesley, 1951.

Martin Campbell-Kelly (M.Campbell-Kelly@warwick.ac.uk) is a professor in the Department of Computer Science at the University of Warwick, where he specializes in the history of computing.

Calendar of Events

September 18–24
ACM SIGPLAN International Conference on Functional Programming,
Tokyo, Japan,
Sponsored: SIGPLAN,
Contact: Manuel M. T. Chakravarty,
Email: chak@cse.unsw.edu.au

September 19–22
ACM Symposium on Document Engineering,
Mountain View, CA,
Sponsored: SIGWEB,
Contact: Matthew Robert Britt Hardy,
Email: mahardy@adobe.com

September 19–21
The 30th International Conference on Computer Safety, Reliability, and Security,
Naples, Italy,
Contact: Flammini Francesco,
Email: Francesco.flammini@ansaldo-sts.com

September 21–23
9th International Conference on Computational Methods in Systems Biology,
Paris, France,
Contact: Fages Francois,
Email: francois.fages@inria.fr

September 26–27
Multimedia and Security,
Buffalo, NY,
Sponsored: SIGMM,
Contact: Chad David Heitzenrater,
Email: heitzenrater@alum.rit.edu

September 26–30
The 3rd Extreme Conference on Communication – The Amazon Expedition,
Manaus, Brazil,
Contact: Pan Hui,
Email: pan.hui@cl.cam.ac.uk

September 28–30
Academic MindTrek 2011,
Tampere, Finland,
Contact: Artur R. Lugmayr,
Email: artur.lugmayr@tut.fi

September 20–October 1
Information Security Curriculum,
Kennesaw, Georgia,
Contact: Michael E. Whitman,
Email: mwhitman@kennesaw.edu

DOI:10.1145/1995376.1995387

Ron Babin, Steve Briggs, and Brian Nicholson

Emerging Markets
Corporate Social Responsibility and Global IT Outsourcing

How to improve IT outsourcing relationships while doing good for society.

GLOBAL IT OUTSOURCING (GITO) is an increasingly accepted business tactic, and continues to grow at healthy rates. In 2008, the value of the global ITO market was estimated at between $220 to $250 billion. The estimate for 2009–2014 is that ITO will grow by 6%–9% per annum.[8]

Today, corporate social responsibility (CSR) is a priority item on the agenda of almost every business organization. Not surprisingly, leading GITO providers have embraced it, and ongoing research at the University of Manchester suggests that some buyers and providers of GITO are gaining competitive advantage from the implementation of CSR projects.[1,3]

Elkington[4] describes how CSR can be integrated into every aspect of social, political, and economic activity, creating "win-win-win strategies...to simultaneously benefit the company, its customers, and the environment." Elkington describes the "triple bottom line" where people, planet, and profits are all considerations in evaluating company performance. He suggests that "successful companies will have little option but to get involved in this rapidly emerging area." Emerson[5] similarly describes "a significant rise in the number of mainstream corporate CEOs discussing the social and environmental performance of their firms." He concludes that "life is not driven strictly by either social or financial realities...we may use financial resources to expand and sustain the core value of organization, community, and individual."

GITO buyers increasingly expect providers to deliver CSR capability as both parties react to global standards and employee expectations. Most major GITO providers (including the major Indian providers) have therefore striven to meet emerging global CSR standards and many have produced elaborate CSR documents that can be downloaded from their Web sites.

From our research we found that GITO CSR projects can be divided into three categories:

Buyers and providers who collaborate on CSR initiatives create strong business value in the outsourcing relationship and create social value for the communities in which they operate.

- Philanthropy and good citizenship, for example, making donations to worthy causes, such as the work of the Indian IT outsourcing firm Infosys' Foundation;
- Compliance with standards such as the Global Reporting Initiative (GRI), which is increasingly a standard requirement in outsourcing requests for proposal; and
- Collaborative CSR activity where the outsourcing buyer and provider work together to achieve shared societal goals and also strengthen their relationship, thus "doing well by doing good." Most leading GITO providers participate in the first two CSR categories: they make contributions to good causes and comply with global CSR standards. A few providers and buyers are engaging in collaborative CSR activity and are able to report some interesting benefits.

The third category is relatively new to GITO relationships, and our research has found within it some novel developments. Porter and Kramer refer to this type of CSR activity as "strategic CSR," since companies attain "greater competitiveness through corporate strategy by advancing social conditions."[7] In our research we found that buyers and providers who collaborate on CSR initiatives create strong business value in the outsourcing relationship, and create social value for the communities in which they operate. Porter and Kramer refer to this as "creating shared value"[6] through strategic CSR.

Collaborative CSR in GITO

GITO providers and buyers who gain advantage from collaborative CSR are proactive in embracing CSR projects that will enhance their existing services or even create new capabilities in their portfolio. One particular example is a joint project by U.K.-based Cooperative Financial Services (CFS) and Steria, a French GITO provider with centers in the U.K. and India. CFS and Steria have collaborated on several CSR projects. A CFS executive told us that as a GITO client they expect of their outsourcing relationships a "common set of values, to build trust with like-minded organizations where we share something in common." This has been realized in the relationship with Steria.

Working together, CFS and Steria support a number of schools in India through the provision of library equipment, computers, sporting equipment, and other funded services to allow students in poor communities to be educated in a well-equipped school. Executives from both companies visit the schools regularly and meet with the teachers, students, and administrators. The local Steria staff visit and work at the schools, mentoring children and contributing budgeted time to the teaching and maintenance of the school project equipment. In parallel, Steria and CFS staff collaborate on social projects in the U.K., contributing staff time jointly to supporting local projects such as community shelters and schools. This is one example of what could be done collaboratively but it is not unique or exclusive; Steria has implemented similar collaborative CSR models with other clients besides CFS.

Overall, collaboration on CSR projects has built a greater level of trust and mutual understanding between CFS and Steria. This enhanced trust and understanding creates work force-related benefits. It contributes to reduced attrition, improves staff retention, and improves productivity through improved team morale.

Collaborative CSR benefits all parties: the buyer gains a more productive provider; the provider gains a more loyal and effective work force; society gains a better-equipped school with students who are more likely succeed in life, and who may one day find employment in the GITO industry. This fits with the Porter and Kramer definition of shared value from strategic CSR.

ILLUSTRATION BY JON HAN

Steria staff told us attrition rates on the CFS project team in Steria India were substantially lower than attrition rates on similar project teams. This results in reduced costs for recruitment, training, and knowledge transfer. A CFS executive stated: "India is booming again this year. And attrition is a massive issue for the industry... attrition on the CFS account is really nothing compared to others, I mean it stands out."..."it is certainly linked to the strength of the relationship between the two organizations which is really visible to the guys in India."

Collaborative CSR helps inspire commitment to stay, work longer and harder for a responsible and caring organization. The willingness to collaborate demonstrates the importance of CSR values evidenced through personal contributions. The payback is in the form of improved productivity and reduced retraining costs. Retained staff tends to be more productive than those newly employed. For the buyer and provider this commitment offers the ability to work through difficult and unexpected problems in the outsourcing relationship. A CFS executive told us: "When you are under pressure of getting the tests done against a tight deadline, do you know what the guys say? Well, time to go home now or do they stay for midnight? These guys stay until midnight and beyond and all

night if necessary....the school [CSR] thing is just a little part of that—it just builds that."

Similarly, this was mirrored by the provider: "People stay longer, sometimes they're investing a lot of their time and it's not paid, I won't necessarily see it on my bottom line, but you will see it on the productivity of that project, of hitting targets, etc."

For the outsource provider and buyer, the enhanced trust improves organizational and individual interpersonal communications. CFS managers told us that joint CSR agendas can be another tool in building effective communications and business relationships. As one CFS executive explained: "I would just say it massively helps with our relationship and how we work together, and what it does when you're working with people painting a classroom or clearing a play area, you also bring in teamwork and there's so many other things that come into it, other skills, communication. You really get to know the people who you're working with, and when you see them out of a techie environment, it makes a huge difference."

Social networks established outside of the formal work environment on CSR projects tended to lead to esteem between individuals and friendly relationships. Subsequently, staff from both buyer and provider organizations felt able to cut through the formal organizational communications hierarchy to solve problems rather than resorting to formal contractual resolution. A Steria executive echoed a similar sentiment: "So when you've been to these places and shared the experience with people, it does help form a very close relationship...Let's face it, in outsourcing things don't go perfectly well over time, they don't and that's the reality of it. You're in a world where you're delivering projects and services. It's a fast-moving world and not all projects go perfectly well. Good relationships get you through those situations...you keep the clients that you have, and that's about strong relationships."

Opportunities for buyers to work cooperatively with providers on CSR initiatives will grow.

Directions on CSR for GITO Buyers and Providers

What lessons does our research provide for buyers and providers of outsourcing services? We have three suggestions.

- More providers should explore the collaborative CSR option, and seek to match CSR projects with buyers in order to build trust and commitment, reduce attrition, improve productivity, and increase organizational and interpersonal communications. In addition to shared views on CSR, buyers should expect CSR leadership from their outsourcing providers.
- Buyers will increasingly demand evidence of compliance with global CSR standards such as the GRI and the UN Global Compact. In a review of outsource provider public profiles we found that the large global providers demonstrated mature CSR capabilities in terms of meeting global standards, while the mid-tier or smaller outsourcer providers are still building their CSR capability.[2] We also found that buyers infrequently validate the provider CSR claims. So a caution to outsource buyers: beware of unsubstantiated CSR claims, particularly from small and mid-tier providers. Several independent consultancies are able to assist buyers with CSR audits of potential outsourcing providers.
- Although our case example focused on social responsibility, environmental responsibility is also a component of CSR concern for global IT outsourcing buyers. Providers with data centers and related technologies must be able to demonstrate energy efficiency that exceeds the levels set by buyers and required by governments. For example, a provider should be able to demonstrate reduced carbon emissions through power management that is more efficient than the buyer could achieve. In accordance with environmental responsibility, outsource providers must increasingly comply with government and industry standards.

Conclusion

In light of the benefits reported in this research, opportunities for buyers to work cooperatively with providers on CSR initiatives will grow. There will be continued uptake of CSR practices by providers, and buyers will need to determine their individual appetite and focus for working collaboratively on such projects.

Finally, we should note that this research is indicative of the reported benefits of collaborative CSR but there are many factors at play. The research has highlighted potentially significant business benefits. However, isolating and quantifying the value of collaborative CSR in substantiated financial terms has not been fully proven. Our research is continuing to examine collaborative CSR at CFS and at other organizations. We welcome comments and contributions from other organizations with similar CSR experiences.

References

1. Babin, R. and Nicholson, B. Corporate social and environmental responsibility in global IT outsourcing. *MIS Quarterly Executive 8*, 4 (Dec. 2009), 123–132.
2. Babin, R. and Nicholson, B. How green is my outsourcer: Measuring sustainability in global IT outsourcing. *Strategic Outsourcing, International Journal 4*, 1 (Jan. 2011), 47–66.
3. Babin, R. and Nicholson B. Sustainability Practices in Global IT Outsourcing. Manchester Business School Research Paper 602, University of Manchester U.K. (June 2010); http://papers.ssrn.com/abstract=1683288
4. Elkington, J. Towards the sustainable corporation: Win-win-win business strategies for sustainable development. *California Management Review 36*, 2 (Feb. 1994), 90–100.
5. Emerson, J. The blended value proposition: Integrating social and financial returns. *California Management Review 45*, 4 (Apr. 2003), 35–51.
6. Porter, M. and Kramer M. Creating shared value. *Harvard Business Review 89*, 1/2 (Jan.–Feb. 2011), 62–77.
7. Porter, M. and Kramer M. Strategy and society: The link between competitive advantage and corporate social responsibility. *Harvard Business Review 84*, 12 (Dec. 2006), 78–92.
8. Willcocks, L.P. and Lacity M. *The Practice of Outsourcing: From ITO to BPO and Offshoring.* Palgrave, London, 2009.

Ron Babin (rbabin@ryerson.ca) is an assistant professor and associate director at the Ted Rogers School of IT Management at Ryerson University in Toronto and a doctoral candidate at the Manchester Business School, U.K.

Steve Briggs (steve.briggs@cfs.coop) is the head of Strategic Partnerships at Co-operative Financial Services (CFS) in Manchester, U.K., where he has managed several major outsourcing relationships, and is also a director of the U.K. National Outsourcing Association.

Brian Nicholson (brian.nicholson@manchester.ac.uk) is a senior lecturer at Manchester Business School and has been involved in teaching, research, and consultancy projects in the broad area of global outsourcing of software and other business processes since 1995.

DOI:10.1145/1995376.1995388

Peter J. Denning and Ritu Raj

The Profession of IT
Managing Time, Part 2

Masterful time management means not just tracking of messages in your personal environment, but managing your coordination network with others.

In a previous installment of this column (March 2011) we took a new look at time management from the perspective of personal productivity.[2] We focused on practices you can adopt in your personal environment to manage your time well and productively. The practices are tracking, selecting, executing, and capacity planning.

As useful as it is, a framework for personal management of commitments is not sufficient for maximum productivity. The reason is that you depend heavily on others fulfilling their commitments to you before you can complete yours. Failures or delays in the other commitments can block your productivity, cause you to take defense measures such as nagging, and sometimes force you to find other people to supply what you need. In a personal commitment management framework, you have no control over these external factors.

Interactions with others are visible in your personal framework as points where you receive requests or issue promises. Seeing those points is not the same as managing the coordination they represent. Managing interactions is crucial for productivity of the entire group, not just you. In this column we examine how the large number of messages relating to external coordination can produce an information fog that can only be dispelled by teaching yourself to observe the coordination loops you engage in with others.

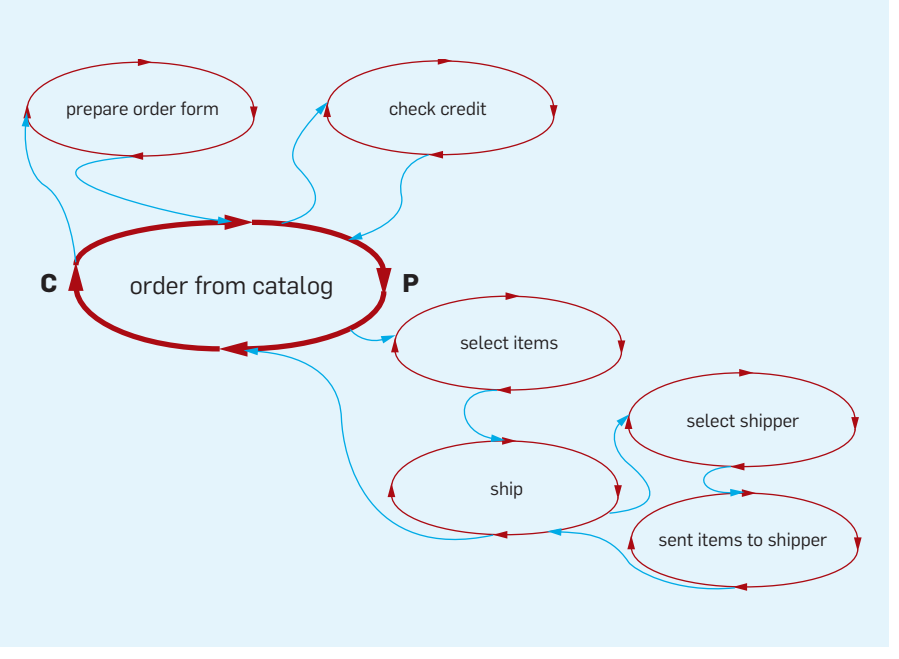
Information Glut

Information glut is an archenemy of productivity. When the total amount of information coming into your personal environment passes a saturation point, your productivity starts to suffer because you can no longer make sense of the information and find solid grounding for your decisions. How can you be productive when you must sort through a lot of irrelevant, marginally useful, or contradictory information?

On the broadest scale, the information fog includes all the information you might come across in the Internet. Some of that information is discretionary—you asked for it by searching and then "pulling" search results into your environment. Pulled information does not seem to be as serious a threat to productivity as "pushed" information—sent into your environment at the action of others. Some common forms of pushed information are:

1. Spam, ads, and phishing—those who send it have no real expectation you will respond.
2. Notices, newsletters, updates, and carbon copies—others keeping you informed: (a) because you asked

Figure 1. Customer C orders from a catalog of provider P. To implement the main conversation seen by the customer, the provider manages a coordination network of loops staffed by its employees and suppliers.



them to when delegating tasks; (b) because you agreed to a subscription or to the automatic "side benefits" of online purchases; (c) because they had other reasons to inform you even though you did not ask.

3. Specific acts of coordination.

The normal way of minimizing type 1 information is to practice rapid deletion (ignoring) and use spam filters. Most people have this under control. The amount of spam or phishing expeditions reaching their inboxes is not a major source of productivity loss.

The normal way of managing type 2 information is to make requests to be excluded from distributions you do not want to be part of. If people to whom you have delegated tasks are overdoing it, you can ask them to reduce the traffic.

That leaves type 3 information as the main source of pushes that can hurt your productivity. At first glance, it looks like this information is in the form of email, phone, chat, messaging, or even wikis, and can therefore be managed with the filing and calendaring tools embedded into office productivity software. Unfortunately, this view confuses communication of messages with coordination of actions. With a good model of coordination, you can make a significant improvement in your coordination productivity in spite of the message traffic that coordination actions generate.

From Communication to Coordination

Communication is concerned with transfers of messages from senders to recipients. Coordination is concerned with people aligning their actions to achieve common goals.

It is important to make the distinction because most of the work we do is not just our own personal tasks, it is the tasks we do together with others. We refer to the orchestration of these shared tasks as "coordination." Your productivity to a large extent depends on your skill at coordination.

Coordination depends on the parties making requests and keeping promises. The human agreements involved can be recorded, but not automated. A single coordination generates many messages among the parties involved. A good communication system can support coordination, but is not sufficient to achieve coordination.

The fundamental building block of coordination is the action loop. We just summarize it here because it has been well documented elsewhere.[1,3,4] A loop connects two parties, C (customer) and P (performer) whose actions combine to fulfill a shared condition of satisfaction. It consists of the four phases:

C: prepares and delivers a request;

P: negotiates changes and promises to deliver;

P: completes the task and delivers the result; and

C: reviews and accepts the delivery.

Many messages can be exchanged between P and C during each phase. Tracking software can record the desired outcome and monitor progress toward completion.

Either primary party (C or P) may turn to secondary parties to fulfill subtasks for them. Thus the primary loop generates a coordination network of linked subtasks, involving other players. Figure 1 shows an example.

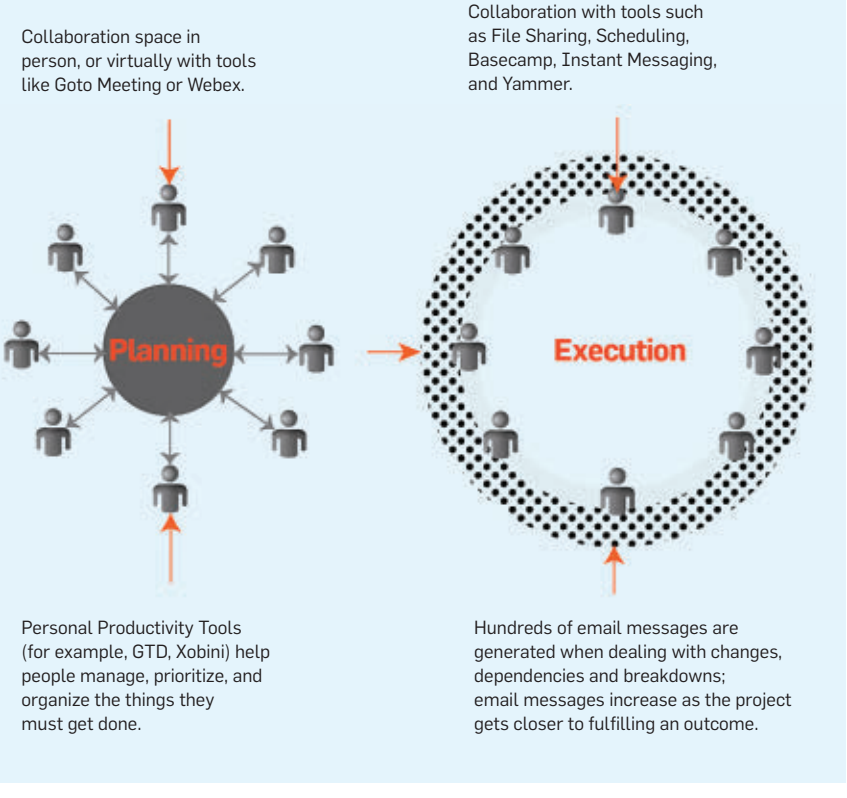
If you do not see that you are interacting with a coordination network, your mailbox will look like a miasmic mishmash of many messages mandating mindful ministration. You will not see the loops and will not complete them satisfactorily, causing you lost time and ill will to fix the mistakes. Your reputation may suffer in the process.

On the other hand, if you do see that you are interacting in a network of loops, you will want tools to help you organize your mailbox so that the loops, rather than the individual messages, are the primary units visible.

Coordination Fog

Larger outcomes need a team of people working together to produce them. In fact, almost all organizations now work in cross-functional teams, often spread over several countries. The usual protocol for making these teams work is repeat the following cycle until the job is done: hold a coordination setup meeting and then split up to do individual tasks. The meetings can be held in person or online with a meet-

Figure 2. The left figure shows what your workspace looks like during the planning stage of a project, when it looks like your part of the project is a pile of personal tasks to be managed. The right figure shows that the coordination tasks between you and others can generate hundreds of email messages, which look like "fog" if you cannot see the coordination network behind them.



ing support system. The team leader directs the conversation to create a common goal, agree on outcomes, divide the work into tasks and milestones, and assign subtasks and milestones to team members. The members then go to their own locations and time zones to carry out their parts of the plan using their personal time management systems.

Unfortunately, as suggested in Figure 2, the "personal" tasks are interdependent. Soon team members discover cases or encounter unexpected circumstances that were not discussed in the plan. Unpredictability is inevitable in our constantly evolving and changing environments. Team members turn to their email, phones, and other media for follow up, get further clarifications, develop action plans for the new circumstances, respond to unforeseen opportunities and threats, and the like. Email is by far the most common medium because, with teammates on the move in different time zones and sometimes in different cultures, it is not easy to resolve these issues on the phone. The mixture gets even more complicated when participants fall into misunderstandings and then miss deadlines or otherwise miscoordinate. They generate additional email messages to overcome misunderstandings and resolve mis-coordinated actions. These coordination issues can easily produce hundreds of email messages. Even simple things like finding a time for a phone conference to resolve issues can take dozens of email messages. This is how unseen coordination generates an information fog that interferes with productivity.

By seeing coordination as a form of conversation management and teaching ourselves to see the loops that are moving toward completion, we can maintain a clear picture of the coordination network and dispel the fog.

The conclusion is that, for most of us, most of our time management is really not "personal." Our commitments always involve others in our networks of coordination. To master your time, therefore, you need to master your ability to make requests and offers (which start loops), your ability to negotiate and agree on the promised results, and your ability to deliver your results by the time you promised. The tools that support you must at the very least track all the loops you are involved in and tell you how far toward completion each one is.

Coordination Software

What software exists to help us see and track the coordination loops we create in our coordination networks?

The first such tool was The Coordinator, produced by Action Technologies in the mid-1980s.[4] It was a mail client that resided on laptop PCs and exchanged messages through a dial-in server. The Coordinator made the individual loops, which it called "conversations for action," visible to the persons engaged in them. The interface was different from ordinary email systems. For example, you would initiate a loop by selecting "request" from a menu, filling in a description of the desired outcome and due date, and sending it to the person you wanted as the performer. The recipient would see your request in a portion of the inbox labeled "incoming requests." With a menu, the recipient would select one of the four allowable responses (accept, decline, counteroffer, or defer). Other menus and mailbox segments covered the remaining parts of unfinished loops. Local databases on both ends tracked all open loops and their states. It was easy to generate to-do lists (promises you committed to), tickler lists (undelivered promises made to you), email chains of loops, and calendar entries from the database. When you dialed in to The Coordinator server, the databases automatically synchronized.

The people who used The Coordinator reported significant productivity gains: they could manage two to 10 times more tasks and projects than before. The email messages themselves also became shorter because they were all linked to their parent loops; with a single click, for example, you could see what request an email message that said "I accept" was accepting.

A small group of critics thought The Coordinator was a form of "surveillance software" that could be abused by unscrupulous managers who might watch the fine details of people's interactions and penalize them for small infractions. The lesson was that people in organizations where employees do not trust management might not welcome a good coordination tool.

Other tools superseded The Coordinator. Action Technologies produced Metro, which mapped and tracked entire coordination networks. Lotus Notes provided a freeform system in which separate databases would track conversations within a project team. Some of the ideas such as linking promise due dates to calendars have been incorporated into modern systems such as Apple Mail and Microsoft Outlook. Recently, OrchestratorMail has been designed as an XML overlay on to any existing mail system to make visible the coordination network generating the email messages.

Conclusion

Many of us get overwhelmed by an information fog of email messages, which interferes with our ability to get productive work done and puts us into unproductive moods such as overwhelm and anger over mis-coordinated actions. One coordination task can require dozens of email messages. If all we can see is the email messages, it quickly becomes a fog. If we could see the coordination task itself, we have much less to track and we can let the computer systems manage the email messages automatically.

When this is done, we become more productive and enjoy reputations of greater trust. What a great augmentation it can be to your personal productivity system to learn the language of coordination, become an observer of coordination acts and state, and have the tools to automatically manage the underlying communications. C

References

1. Denning, P. Accomplishment. *Commun. ACM 46*, 7 (July 2003), 19–23; DOI: 10.1145/792704.792722.
2. Denning, P. Managing time. *Commun. ACM 54*, 3 (Mar. 2011); DOI: 10.1145/1897852.1897865.
3. Denning, P. and Dunham, R. *The Innovator's Way.* MIT Press, Cambridge, MA, 2010.
4. Winograd, T. and Flores, F. *Understanding Computers and Cognition.* Addison-Wesley, Reading, MA, 1987.

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA, and is a past president of ACM.

Ritu Raj (ritu@orchmail.com) is the founder and president of OrchestratorMail, was a Partner at Accenture, and started two successful companies Wag Hotels, and Avasta, which was acquired by Navisite.

DOI:10.1145/1995376.1995389

Ben Shneiderman, Jennifer Preece, and Peter Pirolli

Viewpoint

Realizing the Value of Social Media Requires Innovative Computing Research

How social media are expanding traditional research and development topics for computer and information scientists.

Social media technologies such as Facebook, Twitter, blogs, wikis, Flickr, and YouTube have garnered more than a billion users. These platforms enable more than friendly chatter and individual expression; they facilitate remarkably diverse and broad participation while accelerating the formation of effective collaborations.

Promising social media projects suggest that dramatic transformations are possible in health care, energy sustainability, environmental conservation, disaster response, and community safety.[14] Some commentators even see social media as a means for economic revitalization through business innovation, educational transformation, and civic revival.[15] However, there are deep challenges in understanding the benefits of social media and ameliorating their dangers. Computer, information, and social scientists, network analysts, system developers, community managers, and many other professionals will have important roles to play as they extend their disciplines with innovative research and development agendas.

The potential for social media impact is illustrated by international upheavals such as the Iranian elections,[9] Wikileaks information releases, and Egyptian democratic movement. In addition, a variety of U.S. and other open government efforts have been launched recently to promote transparency, participation, and collaboration. For example, *data.gov* promotes access to detailed U.S. government agency performance data and *recovery.gov* provides contracting information on the county-by-county use of stimulus money, leading to broader discussion, plus invitations to report fraud, abuse, and waste. Increased participation and collaboration that changes the relationship between government agencies and the general public is beginning with *challenge.gov*, which invites solutions to problems, *serve.gov* to expand volunteering, and wiki-based deliberative Web sites to request commentary on agency directions or regulatory plans.

There are deep challenges in understanding the benefits of social media and ameliorating their dangers.

Social media present dangers too. These include the potential for more polarized discussions as users selectively view only materials aligned with their world view and scientists retreat to narrow research topics ("balkanization") that limit the healthy interchange with related disciplines.[16] Another risk is reduced credibility of online resources as rumors and misinformation spread, unfiltered by traditional journalistic verification. Social media can distract from deep reflection as individuals respond to frequent interruptions and collaborative production methods with free distribution can undermine established reward systems, as journalists have painfully discovered.[6] Breaches of privacy and security are frequently mentioned topics and so is identity theft, online bullying, and disclosure of potentially damaging or embarrassing personal information.

Goals and Challenges for Computing Research

Realizing the full value of social media requires research agendas that include understanding the mechanisms for unleashing chain reactions of human contributions and collaborations while preventing harmful outcomes such as privacy violations, malicious attacks, and misuse by terrorists, oppressive regimes, and criminals. Evo-

lutionary patterns of activity within homogeneous or heterogeneous small, medium, and large organizations could be studied with network analysis tools to identify highly productive individuals or groups.[5,8] Understanding the dynamics of collective action, governance, and leadership in networked organizations can present grand scientific challenges that are worthy of Nobel Prize recognition, such as bestowed on Elinor Ostrom.[11] However, early successes such as Wikipedia and health discussion groups generate the impression that success in using social media is inevitable, but the reality is that failure is the norm and even successful projects have problems. For Wikipedia, only one out of every 1,000 readers registers to make contributions—and even fewer participate in durable collaborations. Higher rates of participation are needed for smaller projects to succeed.

One model of how participation evolves is the Reader-to-Leader Framework (see Figure 1), which also offers usability and sociability design guidelines.[13] This framework describes how some of the large numbers of readers mature into contributors who offer user-generated content such as videos, photos, reviews, and ratings. A smaller segment becomes intensely involved in collaborative groups who discuss substantive changes and expansions of content. Finally, a small group of leaders emerge to set policies, deal with attacks, resolve disputes, and mentor newcomers. A major research effort could validate and refine such frameworks, providing deeper insights into the nature of human motivation in different contexts.

Not every computing scientist will be interested in studying social media, but computing science social media research can have a profound impact on every discipline.

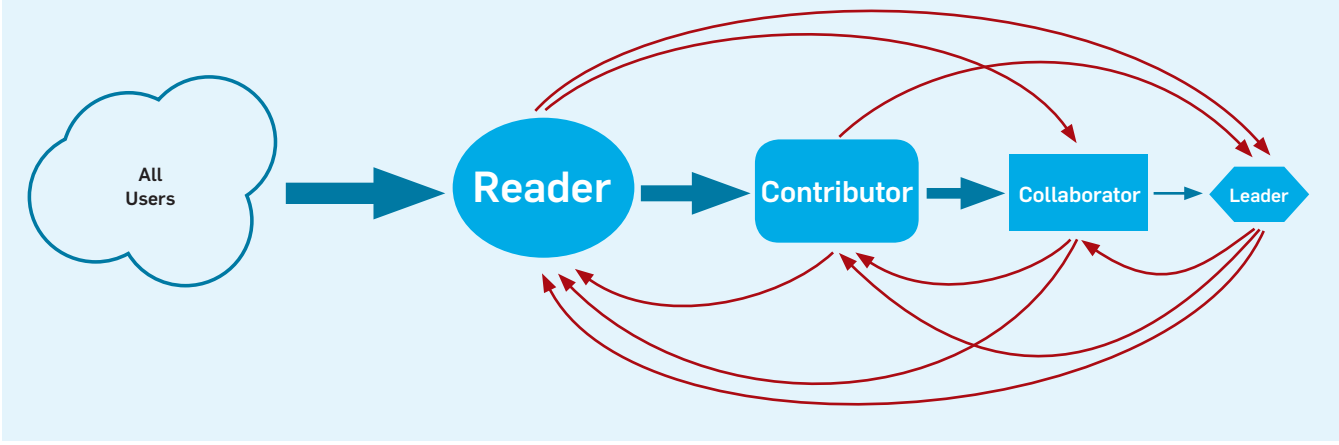
The emerging science of online motivation draws on sociological studies and political science theories, as well as on statistical methods, agent-based simulations, linguistic sentiment analysis, and network analysis/visualization.[4] For example, studying trust, in its many forms, would lead to improved designs that facilitate collaboration so that participants can rapidly resolve their differences and act effectively when needed, as some environmental groups did following the Gulf oil spill.[3]

Another research topic is the growing availability of *big social data*,[8] which presents significant challenges to algorithm designers and mathematicians possibly requiring innovative chip designs to accelerate the necessary computations. Just as graphical processing units (GPUs) have enabled rapid 3D exploration, *social processing units* (*SPUs*) may be needed to enable scalable social network analysis for computations such as eigenvector centrality, community clustering, and comprehensible layouts. While Moore's Law has signaled the steady progress of hardware technologies in petaflops and gigahertz, new laws could describe the growth of massive projects by measuring peta-contribs and giga-collabs.

New scientific measures are also needed for trust, empathy, responsibility, and privacy, and new mathematical operators could characterize the relationship among relevant usability and sociability measures. The rich contextual and volatile temporal dependencies among these measures mean traditional reductionist models need to be enriched with inter-variable sensitivity analysis and informed by qualitative studies. The motivations for early Wikipedia users may be very different from the community safety organizers who must develop trust and ensure privacy over many years. Similarly, those engaged in collective intelligence projects may respond to very different motivations from those who conduct collective action initiatives. Weak ties are sufficient for early stages and for spreading ideas, but strong ties also become vital for the deep commitments necessary to produce substantial change.

Multidisciplinary network science is rapidly emerging with models of network growth/decay, strategies for comparing thousands of apparently similar networks, and algorithms for detecting unusual bursts of activity.[1,2,4] These methods, strategies, and algorithms

Figure 1. The Reader-to-Leader Framework suggests the evolutionary path for participants in social media communities. Some users may move smoothly through the four phases, while others may take different paths as indicated by the arrows in the figure.



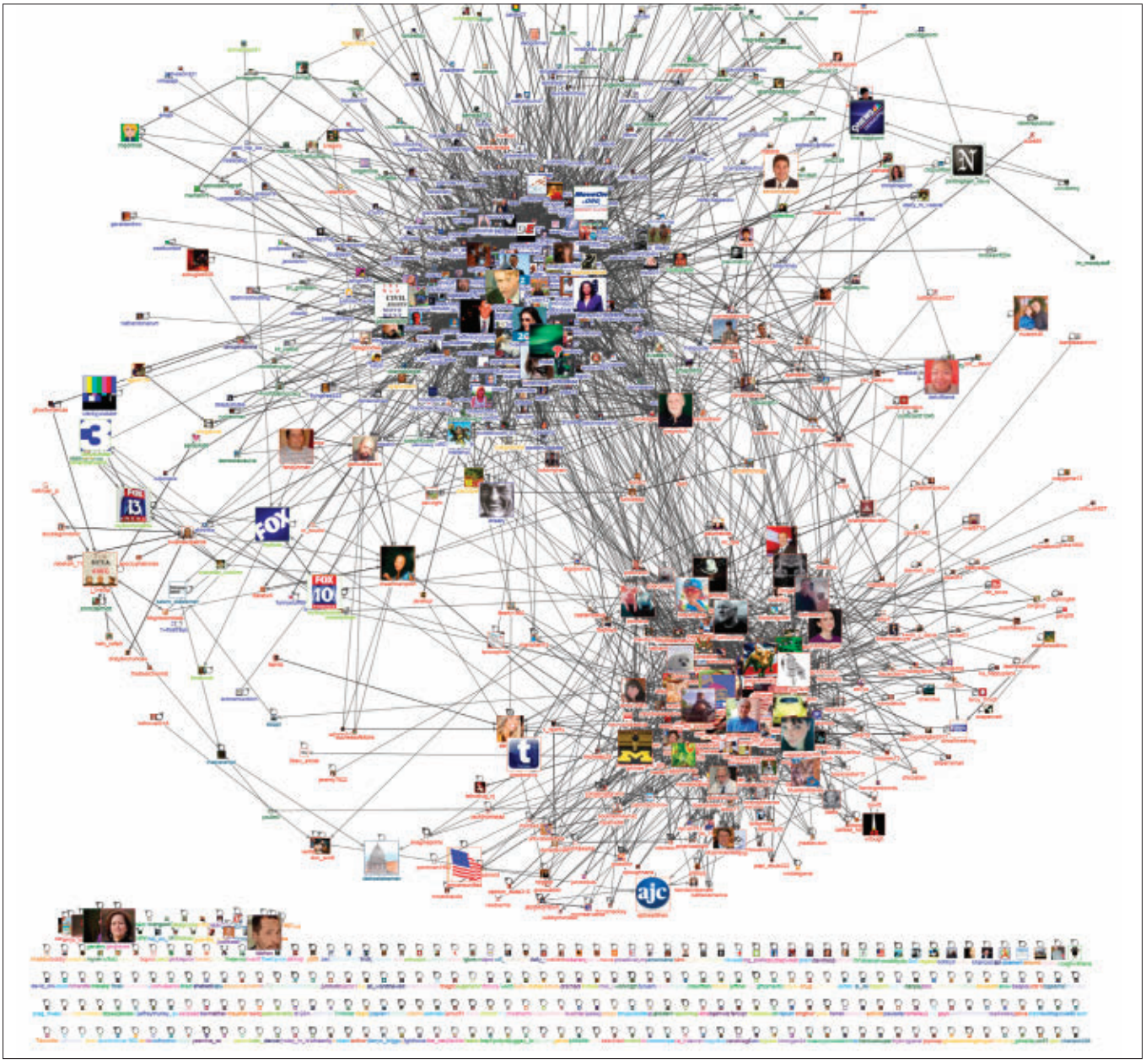


Figure 2. Connections among Twitter users who recently mentioned GOP when queried on July 25, 2011, with vertices scaled by numbers of followers. The clusters are created by the patterns of connections (follows, replies, and mentions) among the authors in the graph. The clusters were based on Clauset-Newman-Moore algorithmic analysis in which the red cluster is composed of largely GOP supporters, while the blue cluster contains largely critics and opponents of the GOP as indicated by the content of the tweets from each cluster. Other colored or shaped nodes are not strongly affiliated with either major cluster. Users on the bottom are not connected with any of the other Twitter users.

will benefit from coupling with natural language processing and discourse analysis to identify nexuses of positive collaborations as well as threatening activity from hate groups, terrorists, and criminals (see Figure 2).

Still more ambitious research goals are to identify key influencers, successful discussion generators, and reliable answer providers in discussion groups with millions of participants while curbing the damage caused by scammers, spammers, and troublemakers of many kinds who seek to undermine the efficacy of social media platforms (see http://www.wikitrust.net).

While many of these topics will be new to computer and information scientists, the social media will dramatically expand their traditional research and development topics such as large-scale heterogeneous distributed systems design, exploratory search tasks across enormous multimedia databases, and visual analytic tools with statistical components that produce valuable insights even from voluminous and noisy data. Other traditional challenges that will become even more central include context-aware systems that work on mobile, laptop, Web, and cloud-based platforms, and policy-aware systems that allow successful operation in different cultures, languages, and political systems.

Broad Scholarly Payoffs

Not every computing scientist will be interested in studying social media, but computing science social media research can have a profound impact on every discipline. Social media

VISUALIZATION BY MARC A. SMITH USING NODEXL; HTTP://WWW.CODEPLEX.COM/NODEXL

are already restructuring the ways in which scholars form collaborations and communicate their results.[10] What used to be called the *invisible college* of personal scholarly communications is now a vast and highly visible, searchable, and influential infrastructure. These new scholarly social networks, the *visible commons*, ignite hot topics, accelerate data sharing, and enable rapid refinements to theories in ways that were never before possible. For example, in August 2010, when a researcher claimed to have proven one of the most profound, challenging, and elusive problems in all of mathematics and computer science (P=NP?), blogs (such as http://rjlipton.wordpress.com), wikis, and other forms of online communication conveyed active discussion about the proof—and ultimately enabled a form of real-time "peer review" that called into question the researcher's approach.

Scientists also have begun to use social media to conduct new forms of scientific research. NASA's use of clickworkers to measure Martian craters (http://beamartian.jpl.nasa.gov) or the Encyclopedia of Life's (http://eol.org) integration of professional scientists with trained citizen scientists and nature enthusiasts are examples of even more potent methods. Scientists can now engage with thousands of peers as in the GeneWiki (http://genwiki.eva.mpg.de), with serious amateurs as in star surveys (http://galaxyzoo.org), or with numerous paid workers through services such as Mechanical Turk (http://mturk.com). Such large-scale collaborations could produce conflict over credit for breakthroughs unless new strategies for supporting trust are created.[10,15] Other ethical dilemmas come from the appropriateness of existing Institutional Review Board oversight processes or fairness of using low-paid Web-based labor in place of traditional research assistants or experimental participants.

Call to Action

These topics provoked lively discussions at two National Science Foundation (NSF)-funded workshops held in the past year. The final report[12] covers descriptive, explanatory, prescriptive, and predictive theories; opportunities in health care/wellness and e-government; ethical issues for researchers; design strategies for practitioners; motivational challenges for community managers; research infrastructure proposals; and innovative educational reforms (http://www.tmsp.umd.edu). Some steps in expanding research have already begun with the NSF's Social Computational Systems program (http://www.nsf.gov/pubs/2010/nsf10600/nsf10600.htm) and the National Institutes of Health's two programs on Social Network Analysis and Health (http://obssr.od.nih.gov/funding_opportunities/foas/faqs.aspx).

The next step will be paradigm-shifting methods for conducting scholarly research in the computing sciences and in every discipline.

Researchers from many disciplines can build on the ideas generated at these workshops and summarized here by working with funding agencies to restructure existing programs so that social media research becomes more widely supported. Evaluations of civic social media projects could make them more reliably successful by developing validated design guidelines, effective community management strategies, advanced visual analytic and statistical tools, and broader theories. Academics can spread this new knowledge by introducing segments on social media into existing courses, adding new courses, and planning degree programs for professionals and researchers.

Adventurous researchers are already using social media to improve or speed their research, but the next step will be paradigm-shifting methods for conducting scholarly research in the computing sciences and in every discipline. Faster paths to curing cancer, tracking climate change, mapping species distribution, and much more seem within reach. However, there is also a risk that social media researchers will soon confront ethical challenges as serious as those that the nuclear physicists faced in the 1950s. This time the concerns will be about inequities in Internet access, violations of privacy, vulnerability to attacks, as well as technical failures and social chaos during crises. We believe the computing sciences community can rise to these challenges and find effective solutions. C

References

1. Barabasi, A.-L., *Bursts: The Hidden Pattern Behind Everything We Do*. Dutton, NY, 2010.
2. Easley, D. and Kleinberg, J. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*, Cambridge University Press, NY, 2010.
3. Golbeck, J. Weaving a web of trust. *Science, 321*, 5896 (2008), 1640–1641.
4. Hansen, D., Shneiderman, B., and Smith, M.A. *Analyzing Social Media Networks with NodeXL: Insights from a Connected World*, Morgan Kaufmann Publishers, San Francisco, CA, 2011.
5. Hendler, J. et al. Web Science: An interdisciplinary approach to understanding the Web. *Commun. ACM 51*, 7 (July 2008), 62–69.
6. Lanier, J. *You Are Not a Gadget: A Manifesto*, Knopf Publishers, NY, 2010.
7. Latour, B. and Woolgar, S. *Laboratory Life: The Construction of Scientific Facts*. Princeton University Press, Princeton, NJ, 1986.
8. Lazer, D., et al. Computational social science. *Science 323* (Feb. 6, 2009), 721–723.
9. Lichtenstein, J. Digital diplomacy. *New York Times Magazine* (July 18, 2010), 24–29.
10. Olson, G.M., Zimmerman, A., and Bos, N., Eds., *Scientific Collaboration on the Internet*, MIT Press, Cambridge, MA, 2008.
11. Ostrom, E. *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press, NY, 1990.
12. Pirolli, P., Preece, J., and Shneiderman, B., Eds., Technology-mediated social participation (cover feature with seven articles). *IEEE Computer 43*, 11 (Nov. 2010), 20–67.
13. Preece, J. and Shneiderman, B. The Reader-to-Leader Framework: Motivating technology-mediated social participation. *AIS Transactions on Human-Computer Interaction 1*, 1 (Mar. 2009), 13–32; http://aisel.aisnet.org/thci/vol1/iss1/5/
14. Shirky, C. *Cognitive Surplus: Creativity and Generosity in a Connected Age*. Penguin Press, NY, 2010.
15. Tapscott, D. and Williams, A.D. *MacroWikinomics: Rebooting Business and the World*. Portfolio, NY, 2010.
16. Van Alstyne, M. and Brynjolfsson, E. Global village or cyber-balkans? Modeling and measuring the integration of electronic communities. *Management Science 51*, 6 (June 2005), 851–868.

Ben Shneiderman (ben@cs.umd.edu) is a professor in the Department of Computer Science, the founding director of the Human-Computer Interaction Laboratory, and a member of the Institute for Advanced Computer Studies at the University of Maryland at College Park.

Jennifer Preece (preece@umd.edu) is a professor and dean of the Information School at the University of Maryland at College Park.

Peter Pirolli (pirolli@parc.com) is a research fellow in the Augmented Social Cognition Area at the Palo Alto Research Center (PARC).

We appreciate National Science Foundation support (IIS-0956571) to conduct the two workshops and all the participants in those workshops. We appreciate the comments we received from the reviewers and James Hendler.

DOI:10.1145/1995376.1995392

Article development led by acmqueue
queue.acm.org

Technology business plans that assume no competition—ever.

BY PAUL VIXIE

Arrogance in Business Planning

IN THE INTERNET addressing and naming market there is a great deal of competition, margins are thin, and the premiums on good planning and good execution are nowhere higher. To survive, investors and entrepreneurs must be bold. Some entrepreneurs, however, go beyond "bold" and enter the territory of "arrogant" by making the wild assumption that they will have no competitors if they create a new and profitable niche. So it is with those who would unilaterally supplant or redraw the existing Internet resource governance or allocation systems. Because alternative Domain Name System (DNS) roots provide such a well-proven and understood example of this kind of arrogance, this article begins with a short slog through that swamp before discussing the more current and topical matter of alternative numbering Whois.

The DNS *root* is the dictionary of top-level domain names such as .COM or .US. It is managed cooperatively and transparently by a community that includes the Internet Activities Board (IAB), which designates and recognizes the Internet Assigned Number Authority (IANA); the Department of Commerce (U.S. DoC), which contracts for IANA services; and Internet Corporation for Assigned Names and Numbers (ICANN), which operates the IANA functions under that contract. The IANA functions contract includes among other things the job of editing the DNS *root zone* to add new top-level domain names such as .XXX. Each of these entities (IAB, U.S. DoC, ICANN) is itself a multistakeholder body that engages with the community to gather input to the decisions it makes about DNS. This governance model is imperfect, but it has worked for a long time and continues to evolve.

Technically speaking, every Internet device using DNS to look things up assumes there is a universal name space with a root zone to describe the top-level domain names, and there are some well-known root name servers to publish this root zone. To be universal in this context means that every name has a specific identity and will always mean the same thing no matter where you are on the Internet when you look that name up. The Internet Engineering Task Force (IETF) periodically revises the DNS protocol to add new capabilities, but this is always done in a backward-compatible way because of the installed base of hundreds of millions of connected devices. So while we could discuss a possible future in which new devices are connected to the Internet having a broader or somehow multiplicitous view of the DNS name space, as of today the only reliable way to treat this name space is as universal.

Given the high visibility and economic value of a new top-level domain name, DNS has been under considerable pressure to add more such names

ILLUSTRATION BY ALICIA KUBISTA


.NET
.JP
.CN
.IT
.ES
.NZ
.GE
.PT
.NL
.CZ
.INFO
.PL
.SE
.EU
.TH
.FR
.EE
.BIZ
.AR
.TW
.CH
.CA
.CL
.TV
.AZ
.GOV
.UA
.NO
.ZA
.BE
.VN
.LT
.TK
.JS
.EDU
.MOBI
.KZ
.ID
.RO
.DE
.IN
.AU
.SK
.LV
.BR
.ORG
.UK
.AT
.ME
.MX
.DK
.HR
.CC
.HU
.KR
.RU
.US
.COM

ever since the Internet climbed down from its academic ivory tower and became a world-changing dominating commercial and social apparatus. Prior work in this area includes adding a handful of new top-level names (.INFO, .MUSEUM, .BIZ, .XXX, and so on), and current work involves throwing the doors open to hundreds or thousands of new top-level domains (.APPLE or .MICROSOFT could soon exist). In addition to that, several bold (or dare I say, "arrogant") entrepreneurs have tried to enter the market unilaterally.

Here is how this kind of unilateralism goes: first you create your own root zone, usually by copying the IANA root zone at some point in time; and then you try to get ISPs to use your root name servers instead of the IANA root name servers. If you succeed at this, then you try to sell name registrations in your alternative name space, where your new names will be visible only to the ISPs you have convinced to subscribe to your system. No such alternative root zone has really taken off, since this value proposition is pretty shaky—there is no way to manage the risk of conflict between an alternative name and some future real name in the IANA system. There is also no good way to align the interests of the people publishing the alternative names with the interests of some population who might want to look up such names.

What's arrogant here isn't the willingness to charge ahead in spite of the shaky value proposition; it's the assumption that there will be only one alternative DNS name space, even if it is a financial success. Does anyone really think that other investors and entrepreneurs would not follow almost immediately, that other teams looking for their next opportunity would say, "Well, one is enough," or even, "Being a late entrant into that market will be too difficult"? I cannot think of a single supporting example; success breeds copycats, in all times and all places.

It's a marvel why the investors in today's alternative DNS systems didn't ask about copycatting. This is a pretty standard investment question. A bunch of copycats who pull various ISPs into competing alternative DNS systems could all sell the same names to different DNS operators, and there would be no way for customers to tell the difference. Being first would count for nothing.

It's a marvel why the investors in today's alternative DNS systems didn't ask about copycatting. This is a pretty standard investment question.

This spotlights a good test for whether some technology is a candidate for Internet governance infrastructure: Does it have to be done cooperatively, or do the physics allow for competition?

Alternative Numbering Whois

So far I've discussed the governance and economics of domain names, but there is another kind of Internet resource that has some superficial similarities to DNS: Internet numbering resources. Every network and every connected Internet device needs a number. This article focuses on Internet Protocol version 4 (IPv4) addresses, which are usually written as four numbers separated by three dots (e.g., 192.5.5.241 or 192.168.1.1). Some of these numbers are private and can be used only for local communication—for example, the address 192.168.1.1 is used by almost every cable or DSL router in every home in the world. Hosts connected to private networks rely on their routers to translate their private addresses into public addresses, a process known as NAT (network address translation). For the purpose of this article, the discussion is limited to public IPv4 addresses that are globally unique and used without NAT.

Before the commercialization and privatization of the Internet in the 1990s, the U.S. government assigned blocks of IP addresses without fee or contract. This befits the original purpose of the Internet, which was to be an interconnection mechanism for the government and its contractors. When commercialization and privatization began, the IP address-allocation function was moved out of government hands and into an regional Internet registry (RIR) system, which now consists of five registries serving the regions of North America and the Caribbean, Africa, Europe, Asia/Pacific, and Latin America. Each RIR is a nonprofit association serving a community of network operators including both service providers and end users. Allocation policy is set in each region by a public policy development process, and resource allocations are governed by agreements that clearly describe

the allocation as being based on "demonstrated need" for network growth. These agreements also declare that number resources are not property.

Legacy numbering allocations made in the decades before the RIR system was put in place were very large because of the technical limitations of the time. The effect of this today is about half of all allocated numbers are of the legacy type even though most allocations are of the RIR type. Now that the Internet is running short of new IPv4 numbers for network growth, many network operators are looking for ways to acquire the rights to as many IPv4 numbers as possible so they can continue to grow their networks while the Internet converts from IPv4 to IPv6. This makes the older and larger legacy numbers very attractive, since the allocations were larger and are often held by older companies and universities whose needs may be modest by current standards. The holders of legacy numbers have no contractually explicit rights concerning those numbers unless they have sought safe harbor by entering into an RIR contract, but as a practical matter anyone who is using legacy addresses received in the pre-RIR era can safely continue to do so.

The RIR system permits designated transfers between address holders. The goal of the RIR transfer regime is to bring more IPv4 addresses into active use to facilitate network growth during the IPv6 transition. Any network operator who can demonstrate near-term operational need for number resources and who can negotiate a transfer with the current holder of those resources can simply sign an RIR contract and receive rights to the resources. Because this transfer regime was developed through a public policy development process, which is therefore bottom up rather than top down in nature, these rules are literally what the community of network operators asked for—such rules cannot be imposed by any government. Some interested parties, however, may not be able to demonstrate an immediate operational need and thus will not qualify as number-resource recipients. One class of such parties is the network operator who desires a long-term forward reserve. Another class is speculators who will never have need for the numbering resources in their own names but who would like to hold the resources for later monetization (for example, rental or trading in futures).

It's necessary to digest all of this background information to understand that not all interested parties are qualified recipients by the current transfer policies and not all transferable resources are under an explicit contract. The oft-stated concern is that these resources will be traded outside the system and that the RIR records (called Whois) will become useless. Since network operators use the RIR records every day to manage and diagnose their networks, these records should be complete and accurate. One proposal often heard in this context is that RIRs should not regulate transfers in any way and should simply record any transfer brought to them by a cooperating seller and buyer. A supporting argument for this proposal is that Whois can be run by anybody and if the RIRs won't run an accurate Whois system (which is to say, a permissive system accepting the results of any and all transfers without limitation), then somebody else will do so. This argument breeds arrogance.

A strong advantage of the RIR Whois system in the eyes of network operators is that it is universal. There is only one entry for any given netblock and, therefore, effectively only one Whois system even though each RIR independently runs its part of that system. Let's assume for the purposes of argument, however, that an alternative Whois system is created and enough network operators trust it that this alternative system becomes operationally relevant and that a non-RIR resource transfer regime becomes practical. Does anybody really believe that there would be only one alternative Whois system—no copycatting? Or as in the case of alternative DNS described earlier, would not the number of potential alternative Whois systems be limited only by available capital?

It would be technically possible to maintain a list of all alternative Whois systems and to query them all in parallel whenever network operations require knowing the details about a block of IP addresses. Inevitably, however, the same network would appear to be registered to different operators in different Whois systems since freedom from transfer limitations is the stated reason for the very existence of the alternative systems. While anybody can start a new Whois system at any time, the operational usefulness and therefore the relevance of a Whois system depends on coherence and cooperation—two properties that an alternative Whois system and the alternative transfer market it supports would not have.

Conclusion

Any proposal for a competing Whois registry model is as doomed by design and destiny as every alternative DNS system. Even if it succeeds at first, it would fail after copycatting occurred. Participants in RIR public policy development would do well to remember this when evaluating dire warnings of RIR Whois irrelevancy because of an RIR transfer regime having a requirement of near-term demonstrated operational need. Speculators who want to monetize future need and network operators who want a forward reserve might still find ways to act outside the system, but resources will have to come into the system when their ultimate recipients qualify to receive the resources due to then-immediate operational need. The RIR system has no power to govern such private actions, but it need not and should not cede authority over the transfer policy and Whois registry—because that's in the physics. C

Related articles on queue.acm.org

DNS Complexity
Paul Vixie
http://queue.acm.org/detail.cfm?id=1242499

What DNS Is Not
Paul Vixie
http://queue.acm.org/detail.cfm?id=1647302

Successful Strategies for IPv6 Rollouts. Really.
Thomas A. Limoncelli, Vinton G. Cerf
http://queue.acm.org/detail.cfm?id=1959015

Paul Vixie is president of Internet Systems Consortium (ISC), a nonprofit company that operates the DNS F root name server and publishes the BIND software used by 80% of the Internet for DNS publication. He is also chairman of American Registry for Internet Numbers (ARIN), a nonprofit company that allocates Internet number resources in the North America and Caribbean region.

DOI:10.1145/1995376.1995391

Article development led by acmqueue
queue.acm.org

Did Ken, Dennis, and Brian choose wrong with NUL-terminated text strings?

BY POUL-HENNING KAMP

The Most Expensive One-Byte Mistake

INFORMATION TECHNOLOGY (IT) BOTH drives and implements the modern Western-style economy. Thus, we regularly see headlines about staggeringly large amounts of money connected with IT mistakes. Which IT or CS decision has resulted in the most expensive mistake?

Not long ago, a fair number of pundits were doing a lot of hand waving about the financial implications of Sony's troubles with its PlayStation Network, but an event like that does not count here. In my school days, I talked with an inspector from *The Guinness Book of World Records* who explained that for something to be "a true record," it could not be a mere accident; there had to be direct causation starting with human intent (such as, we stuffed 26 high-school students into our music teacher's Volkswagon Beetle and closed the doors).

Sony (probably) did not *intend* to see how big a mess it could make with the least attention to security, so this and other such examples of false economy will not qualify. Another candidate could be IBM's choice of Bill Gates over Gary Kildall to supply the operating system for its personal computer. The damage from this decision is still accumulating at breakneck speed, with StuxNet and the OOXML perversion of the ISO standardization process being exemplary bookends for how far and wide the damage spreads. But that was not really an IT or CS decision. It was a business decision that, as far as history has been able to uncover, centered on Kildall's decision not to accept IBM's nondisclosure demands.

A better example would be the decision for MS-DOS to invent its own directory/filename separator, using the backslash (\) rather than the forward slash (/) that Unix used or the period that DEC used in its operating systems. Apart from the actual damage being relatively modest, however, this does not qualify as a good example either because it was not a real decision selecting a true preference. IBM had decided to use the slash for command flags, eliminating Unix as a precedent, and the period was used between filename and filename extension, making it impossible to follow DEC's example.

Space exploration history offers a pool of well-publicized and expensive mistakes, but interestingly, I did not find any valid candidates there. Fortran syntax errors and space shuttle computer synchronization mistakes do not qualify for lack of intent. Running one part of a project in imperial units and the other in metric is a "random act of management" that has nothing to do with CS or IT.

The best candidate I have been able to come up with is the C/Unix/Posix use of NUL-terminated text strings. The choice was really simple: Should the C language represent strings as an address + length tuple or just as the

address with a magic character (NUL) marking the end? This is a decision that the dynamic trio of Ken Thompson, Dennis Ritchie, and Brian Kernighan must have made one day in the early 1970s, and they had full freedom to choose either way. I have not found any record of the decision, which I admit is a weak point in its candidacy: I do not have proof that it was a conscious decision.

As far as I can determine from my research, however, the `address + length` format was preferred by the majority of programming languages at the time, whereas the `address + magic _ marker` format was used mostly in assembly programs. As the C language was a development from assembly to a portable high-level language, I have a difficult time believing Ken, Dennis, and Brian gave it no thought.

Using an `address + length` format would cost one more byte of overhead than an `address + magic _ marker` format, and their PDP computer had limited core memory. In other words, this could have been a perfectly typical and rational IT or CS decision, like the many similar decisions we all make every day; but this one had quite atypical economic consequences.

Hardware development costs. Initially, Unix had little impact on hardware and instruction set design. The CPUs that offered string manipulation instructions—for example, Z-80 and DEC VAX—did so in terms of the far more widespread `adr+len` model. Once Unix and C gained traction, however, the terminated string appeared on the radar as an optimization target, and CPU designers started to add instructions to deal with them. One example is the Logical String Assist instructions IBM added to the ES/9000 520-based processors in 1992.[1]

Adding instructions to a CPU is not cheap, and it happens only when there are tangible and quantifiable monetary reasons to do so.

Performance costs. IBM added instructions to operate on NUL-terminated strings because its customers spent expensive CPU cycles handling such strings. That bit of information, however, does not tell us if fewer CPU cycles would have been required if a `ptr+len` format had been used.

Thinking a bit about virtual memory (VM) systems settles that question for us. Optimizing the movement of a known-length string of bytes can take advantage of the full width of memory buses and cache lines, without ever

ILLUSTRATION BY GARY NEILL

touching a memory location that is not part of the source or destination string.

One example is FreeBSD's `libc`, where the `bcopy(3)/memcpy(3)` implementation will move as much data as possible in chunks of "unsigned long," typically 32 bits or 64 bits, and then "mop up any trailing bytes" as the comment describes it, with byte-wide operations.[2]

If the source string is NUL terminated, however, attempting to access it in units larger than bytes risks attempting to read characters after the NUL. If the NUL character is the last byte of a VM page and the next VM page is not defined, this would cause the process to die from an unwarranted "page not present" fault.

Of course, it is possible to write code to detect that corner case before engaging the optimized code path, but this adds a relatively high fixed cost to all string moves just to catch this unlikely corner case—not a profitable trade-off by any means.

If we have out-of-band knowledge of the strings, things are different.

Compiler development cost. One thing a compiler often knows about a string is its length, particularly if it is a constant string. This allows the compiler to emit a call to the faster `memcpy(3)` even though the programmer used `strcpy(3)` in the source code.

Deeper code inspection by the compiler allows more advanced optimizations, some of them very clever, but only if somebody has written the code for the compiler to do it. The development of compiler optimizations has historically been neither easy nor cheap, but obviously Apple is hoping this will change with Low-level Virtual Machine (LLVM), where optimizers seem to come *en gros*.

The downside of heavy-duty compiler optimization—in particular, optimizations that take holistic views of the source code and rearrange it in large-scale operations—is that the programmer must be really careful that the source code specifies his or her complete intention precisely. A programmer who worked with the compilers on the Convex C3800 series supercomputers related his experience as "having to program as if the compiler was my ex-wife's lawyer."

Security costs. Even if your compiler does not have hostile intent, source code should be written to hold up to attack, and the NUL-terminated string has a dismal record in this respect. Utter security disasters such as `gets(3)`, which "assume the buffer will be large enough," are a problem "we have relatively under control."[3]

Getting it under control, however, takes additions to compilers that would complain if the `gets(3)` function were called. Despite 15 years of attention, over- and underrunning string buffers is still a preferred attack vector for criminals, and far too often it pays off.

Mitigation of these risks has been added at all levels. Long-missed no-execute bits have been added to CPUs' memory management hardware; operating systems and compilers have added address-space randomization, often at high costs of performance; and static and dynamic analyses of programs have soaked up countless hours, trying to find out if the byzantine diagnostics were real bugs or clever programming.

Yet, absolutely nobody would be surprised if Sony's troubles were revealed to start with a buffer overflow or false NUL-termination assumption.

Slashdot Sensation Prevention Section

We learn from our mistakes, so let me say for the record, before somebody comes up with a catchy but totally misleading Internet headline for this article, that there is absolutely no way Ken, Dennis, and Brian could have foreseen the full consequences of their choice some 30 years ago, and they disclaimed all warranties back then. For all I know, it took at least 15 years before anybody realized why this subtle decision was a bad idea, and few, if any, of my own IT decisions have stood up that long.

In other words, Ken, Dennis, and Brian did the right thing.

But That Doesn't Solve the Problem

To a lot of people, C is a dead language, and ${lang} is the language of the future, for ever-changing transient values of ${lang}. The reality of the situation is that all other languages today directly or indirectly sit on top of Posix API and the NUL-terminated string of C.

When your Java, Python, Ruby, or Haskell program opens a file, its runtime environment passes the filename as a NUL-terminated string to `open(3)`, and when it resolves cacm.acm.org to an IP number, it passes the host name as a NUL-terminated string to `getaddrinfo(3)`. As long as you keep doing that, you retain all the advantages when running your programs on a PDP/11, and all of the disadvantages if you run them on anything else.

I could write a straw-man API proposal here, suggest representations, operations, and error-handling strategies, and I am quite certain it would be a perfectly good waste of a nice afternoon. Experience shows that such proposals go nowhere because the backward compatibility with the PDP/11 and the finite number of programs written are much more important than the ability to write the potentially infinite number of programs in the future in an efficient and secure way.

Thus, the costs of the Ken, Dennis, and Brian decision will keep accumulating, like the dust that over the centuries has almost buried the monuments of ancient Rome. C

Related articles on queue.acm.org

Massively Multiplayer Middleware
Michi Henning
http://queue.acm.org/detail.cfm?id=971591

The Seven Deadly Sins of Linux Security
Bob Toxen
http://queue.acm.org/detail.cfm?id=1255423

B.Y.O.C. (1,342 Times and Counting)
Poul-Henning Kamp
http://queue.acm.org/detail.cfm?id=1944489

References

1. Computer Business Review. Partitioning and Escon enhancements for top-end ES/9000s (1992); http://www.cbronline.com/news/ibm_announcements_71.
2. ViewVC. Contents of /head/lib/libc/string/bcopy.c (2007); http://svnweb.freebsd.org/base/head/lib/libc/string/bcopy.c?view=markup.
3. Wikipedia. Lifeboat sketch (2011); http://en.wikipedia.org/wiki/Lifeboat_sketch.

Poul-Henning Kamp (phk@FreeBSD.org) has programmed computers for 26 years and is the inspiration behind bikeshed.org. His software has been widely adopted as "under the hood" building blocks in both open source and commercial products. His most recent project is the Varnish HTTP accelerator, which is used to speed up large Web sites such as Facebook.

DOI:10.1145/1995376.1995390

Article development led by acmqueue
queue.acm.org

Finding solutions as growth and fragmentation complicate mobile device support.

BY MACHE CREEGER

ACM CTO Roundtable on Mobile Devices in the Enterprise

BLACKBERRY? IPHONE? ANDROID? Thin or fat client? Carrier network or Wi-Fi? Developers of mobile applications have many variables to consider if they are going to be successful in a rapidly changing and increasingly fragmentary environment.

With rapid worldwide growth and increasingly diverse devices and networks, supporting mobile devices in the enterprise is becoming increasingly more challenging and complex.

Application service architectures, security, connectivity, testing, a constantly changing mix of devices and platforms, and an uncertain future are among the concerns mobile application developers must face in deploying mobile device services. Change in this market is the only certainty, and developers must continually look ahead to refine development and deployment strategies to keep up.

In this ACM CTO Roundtable, five leaders in the mobile applications field discuss the current challenges in supporting multiple devices on multiple networks for highly variable business requirements.

—Mache Creeger

Participants

Andrew Toy is past VP, mobile applications at a major Wall Street investment bank; past VP, mobile and syndication technology at MTV Networks; cofounder and CEO of Enterproid.

André Charland is the developer of PhoneGap; and cofounder and CEO of Nitobi.


Moderator Mache Creeger

George Neville-Neil is a past member of the Paranoids group at Yahoo!; and principal of Neville-Neil Consulting.

Carol Realini is past CEO of Chordiant; founder and CEO of Obopay.

Steve Bourne is CTO, El Dorado Ventures; past president of ACM, chair of ACM *Queue* Editorial Board, and chair of ACM Practitioner Board.

Mache Creeger (Moderator) is Principal, Emergent Technology Associates.

CREEGER: Andrew, when you were responsible for the use of mobile devices at a major financial institution years ago, what were the biggest concerns?

TOY: We focused on the BlackBerry. The two major problems we had were our inability to customize services and maintaining control of service reliability. The BlackBerry presented itself as a closed system; the NOC (Network Operations Center), the services, and the server software were all controlled by RIM (Research in Motion). There were very few APIs to work with and because of its proprietary nature, we had a limited understanding of its underlying architecture. As a result, when something broke it was hard to fix. Theoretically it was secure and RIM could talk about why that was true, but the same reasons that made it hard to penetrate made it difficult and expensive to maintain as a mission-critical platform. We always worried about losing email, with our only recourse being to call RIM and demand it be fixed.

While there are lot more device platforms choices today, if you look at operating systems with enterprise capabilities, the only real viable candidates are Apple iOS and, to a lesser extent, Google's Android.

Given these options, enterprise customers now believe they need to support more than just the BlackBerry. However, they are unsure how to go from the RIM world to this new and very different place. In the RIM environment everything is done for you. When you take things into your own hands, you recognize there are a lot of issues that the BlackBerry solution used to address that are now your problem.

NEVILLE-NEIL: What about compliance issues? Suddenly, you've put a huge amount of data that's probably controlled by compliance rules in the hands of people who are wandering around with their devices.

TOY: We found that lawyers can advise on industry-specific compliance requirements, but in finance everything does not necessarily have hard and fast mandated technical standards. Our experience was more along the lines of "you have to protect your customers."

We focused on such things as avoiding client data loss that triggered financial industry-specific mandated actions. Data loss required notifying each client of the breach and potential access by anyone, including a competitor. The loss of a mobile device meant the regulatory notification requirement would be triggered if data security was not provable to some level of technical certainty. Being able to make that guarantee drove us to ensure that proper screen locks and encryption were placed on mobile devices.

It is important to create a culture that does not view the security guy as the enemy. Security should enable things otherwise not possible. If a company wants to enable financial transfers, then you need security, because without it the business will collapse under fraud and real-world attacks. Security is not a goal but a means to deliver business value and manage risk in sustainable ways.

REALINI: My company is about delivering consumer-facing functionality over mobile devices, and we have payment and banking services at the back end. We deliver that functionality in the U.S., as well as India and Africa. Those environments are diverse—a lot of dumb phones, a lot of smart-

PHOTOGRAPHS BY TOM UPTON

STEVE BOURNE

I don't see how existing wireless carriers in the U.S. will be able to make the required capital investment to handle the increased demand for services. Certainly that is the case in the next year or two.

phones—with the mix depending on where you are in the world. The transports are also diverse. Some countries have a lot of data access. Other places, data services just aren't available.

I have worked with mainframes, client server, and Internet-based computing. Mobile is the hardest kind of computing I've experienced because it is a fragmented and rapidly changing device market. You have at least 18 platforms or operating systems, and they're in constant flux.

IT organizations may want to build that mobile expertise in-house, but that's not an effective strategy as the mobile device market is moving way too fast. Growing mobile expertise organically is hard and chances are your company would make too many rookie mistakes. Either hire or outsource to contractors with the expertise and get it right the first time.

As an IT manager, you should ask, "How fast do I need to move?" "How many platforms do I need to support?" and "In how many geographies do I need to operate?" It is important to understand that this is not just another operating system. It is a rapidly moving environment, and it's just going to change faster and get more fragmented over time.

NEVILLE-NEIL: In embedded computing, every product is different and every customer is different. If someone powerful in the company buys an iPhone or Android, he or she then drives the IT department to support it. It's a very customer-driven model.

CREEGER: How do the IT folks avoid being buffeted by everybody coming at them at once?

REALINI: You just get used to it. If you think the world is all about iPhone and Android, just blink and it will be something else. It's going to be a fragmented environment, and it will depend on your application. If you are a large financial services firm, then you may be able to dictate that everybody use BlackBerries. You don't have that luxury if you're doing consumer-facing applications. Every current and future device is part of my world, and we must have strategies to leverage those devices even as the mix of those devices changes constantly.

An interesting question to ask is, "What chance does Android have of becoming the universal operating platform for mobile—the mobile equivalent of Microsoft on the desktop?"

CHARLAND: I don't think you can ever make that assumption about any platform. A year ago, I would have said iPhone would be the universal operating platform for mobile. Today it seems like Android, but things are

ANDRÉ CHARLAND

I want to stress the minimum viable product approach: What value can we provide to our user base and can we do this in the mobile browser?

moving too fast to say that something will not change in the future. There might be a dominant player like Android, but you're never going to be able discount iPhone or BlackBerry.

CREEGER: How do folks decide on appropriate application architectures and the smartphone platforms they will support?

CHARLAND: I would focus on two things: the minimum viable product you can deliver to your users and what platforms are needed to support them. A lot of people look at their Web stats and assume that because people visit the Web site with an iPhone, iPhone should be the first supported platform. To prioritize a list of supported platforms you must perform basic research on your user base: poll your users, look at market trends, and do your best to forecast what phones your users will be buying.

REALINI: Three years ago India had 150 million phones, now it's 700 million. Moreover, the features of these phones are changing quickly. You could do research and then extrapolate, but you have to work quickly and constantly adjust to what's really happening in the market. It is almost like trying to track fashion or pop music.

How do people plan in such a fast-changing environment? They have to ask themselves two questions, and do so often because the answers change as the market changes: Do I want a thick or thin client? Which devices am I going to support? Once you answer those questions and have a strategy in place, you better ask those questions again every 6 to 12 months.

TOY: The key underlying issue is this: Are you buying people their phones or are you trying to support the phones people purchase and bring into your environment? If you mandate what is used, then you can have better control.

BOURNE: Can you really mandate in a modern enterprise, even in a large regulated financial services company?

TOY: Because of the tight regulations in financial services, most certainly. For a multimedia business, I'd say no.

NEVILLE-NEIL: Small businesses are in the most trouble because they're the least likely to be providing their employees with smartphones.

CREEGER: Is virtualization going to be a solution?

TOY: Multiple use-case profiles are the way to solve the multiple-mission problem. Is virtualization the best approach? It is difficult to do power management with a hypervisor on a device with more than one operating system. This doesn't mean that it's impossible or not viable, just that it's extremely challenging.

Right now a mobile operating sys-

tem manages power and does a lot of things under the covers to maximize battery life. Take away the operating system's direct link to the hardware, and you lose the ability to effectively manage battery life. That is a huge blow to the value of the phone.

While you could migrate power management (or the management of any limited resource) of a mobile phone operating system to a hypervisor, you would then be stretching the definition of the traditional hypervisor to something more like an operating system of operating systems—effectively, a very fat hypervisor.

NEVILLE-NEIL: It will probably not happen near term, but it might happen on Android because it has gigahertz phones. Apple will never let a hypervisor execute on an iPhone.

TOY: A sufficient solution might be more like the Unix method of multiple users. You would have one box with multiple users logged in. Each user has his or her own experience; all users run concurrently; and there is one kernel and one operating system.

REALINI: One of the biggest trends in emerging markets is that users have multiple SIM (subscriber identity module) chips in their pockets to optimize the costs of their calls. Carriers have different pricing to different destinations, and users pick the chip that minimizes cost for a specific call. Effectively, they are creating multiple profiles similar to what has been discussed, but instead of maximizing security, it's minimizing charges. In India if a phone does not have dual SIM chip modes to allow the user to change personalities, it will not sell.

Mobile phones are personal—somebody calls me and I know it's for me. We all have multiple personas: businessperson, mother...A personal device must evolve to support these multiple roles.

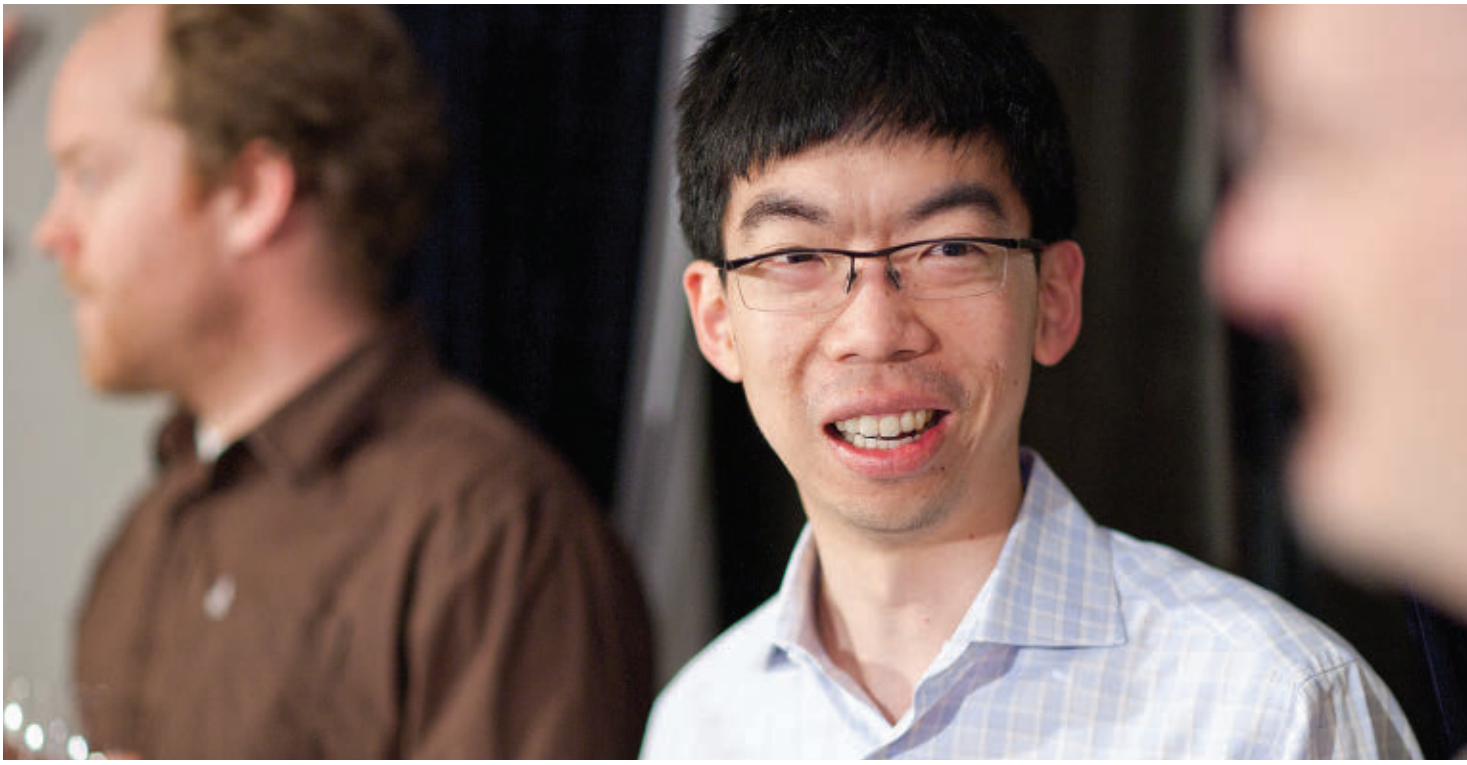
TOY: In the business world, some of those personas can be very tightly managed. For a company employee, that personality can be made to function under a formal corporate security policy.

NEVILLE-NEIL: You are going to see more of the iPhone architecture in most smartphones—a combination of Jails (http://www.freebsd.org/doc/handbook/jails.html#JAILS-SYNOPSIS) and Mac frameworks. These control structures are in Mac OSX and FreeBSD. While I don't believe Android has these functions as yet, and RIM certainly doesn't, smartphones will migrate to this type of approach because virtualization is too heavy and loses control of the lowest layer.

These Apple technologies isolate applications from each other, and

GEORGE NEVILLE-NEIL

The Apple architecture, which is a nonsharing design, is the right place to start in developing a next-generation mobile operating system.



ANDREW TOY

Security is not a goal but a means to deliver business value and manage risk in sustainable ways.

all their APIs have the ability to control where information flows. This introduces some problems when one would like to share data but cannot. Those issues notwithstanding, the Apple architecture, which is a nonsharing design, is the right place to start in developing a next-generation mobile operating system.

TOY: When I was building applications for enterprise-owned BlackBerries, we often asked RIM how to access a particular piece of data. RIM would say it was not possible because it was insecure. That mentality seemed paternalistic in that RIM was going to protect us from ourselves by not allowing certain functions to be implemented regardless of the business need.

CREEGER: Carol's experience shows that network connectivity can be very uneven worldwide. Just take a look at Africa. How do you work with issues such as intermittent connections or long latency when they are the rule rather than the exception?

REALINI: It's more widespread than just Africa. We have one U.S.-based carrier that gave us 36 duplicate SMS messages in the past 30 days. If you're deploying applications that use mobile phones, you cannot depend on a rock-solid 24/7 proprietary network even in developed countries.

CHARLAND: We're focusing on extreme applications with extreme security or other situations where you have to deal with really poor phones and poor networks. It is important not to lose sight of the big swath in the middle, especially in North America and Europe, with a reasonable average for phones and networks and relatively low security requirements. IT managers are going to be faced with this type of environment far more frequently, and their challenge is to build out to different device platforms.

We see many clients deploying HTML5 browser-delivered applications for some things and then native installed applications with a wrapper such as PhoneGap. It really depends on the devices they're targeting and their use cases.

NEVILLE-NEIL: Carol, how do you deal with the software management problem? How do you manage versioning?

REALINI: Our approach is either to purchase or build tools to help us be efficient. We figure out how to work with the most devices efficiently and how to create reference ports. You have to target what devices you believe people are going to use and be efficient about doing a reference port because it's not just one device, it's multiple devices.

We invest to get a superior user experience on 80% of the installed base of phones. This is done by an applica-

CAROL REALINI
Mobile is the hardest kind of computing I've experienced because it is a fragmented and rapidly changing device market. You have at least 18 platforms or operating systems, and they're in constant flux.

tion on the phone or on its STK (SIM application toolkit) where the carrier distributes the application as part of its SIM chip.

CREEGER: You have several different ways to develop applications. How do you make those kinds of decisions?

REALINI: You have to look at things early and often because this is a moving target. We use the 80/20 rule, with 80% of the devices providing a good to great user experience and the other 20% providing adequate user experience.

TOY: The key is to have a tiered strategy and not go for the silver bullet. Don't say which device is the right one. While all devices could probably be supported, you have to ask, "What is the right functionality to have on each platform, and what is the minimum functionality required for any device?"

REALINI: The CTO of my company puts things in three buckets:

- I know it works because I've certified it.
- I think it works because the device manufacturer said it's totally compatible with earlier implementations.
- I have no idea if it works because multiple changes have happened.

This is important because your consumers and/or employees have to be able to make a decision from the thousand or so devices that could be purchased. You have to help them identify the 200 or so devices that will probably work well and the 50 or so devices that have been certified.

TOY: With the world changing so fast, you have to make the effort to keep those buckets up to date and revisit your categorizations frequently.

REALINI: There's a cost to making sure things are in the right buckets. We had a specific credit-card application in place when the iPad was launched. At that point, everyone was told that all iPhone applications would work on the iPad. That falls into my second bucket: I think it should work because Apple told me iPhone applications work on iPads. Well, guess what? It didn't work.

That experience taught us that we have to say to our partners, such as the credit card company, that we *think* it works, but if you want to be sure we better go through a three-week certification process.

NEVILLE-NEIL: In targeting one or more platforms for an application set, you should use minimal surface area to maximal effect. Android or iOS has the system-call complexity of a modern workstation operating system—thousands and thousands of possible APIs. When trying to design portable software, use the:

- Fewest APIs possible—this limits

the complexity of porting the software to new devices.

▸ Oldest APIs—they have been around long enough to be supported by many different device variants.

▸ Best tested APIs—they will be the most reliable.

CHARLAND: Ideally, in cross-platform software development projects, we first target BlackBerry, as it is the most minimal platform. We negotiate a minimum operating system release level with the customer, typically pushing for at least 4.6. Currently, BlackBerry is at version 6.0, and if that is acceptable, it makes for a much richer application platform.

We focus on 4.6 because there are still a lot of enterprise users on it. We target what we can, using the browser as an application and build up from there to Android and iPhone. It's important to stick to this philosophy and not start with an iPhone application and try to work back to BlackBerry. That approach often leads to emulating iPhone features on a BlackBerry, at best an extremely painful effort.

NEVILLE-NEIL: Apple does try to make it easy to move things from the Mac desktop environment to iOS, but it's not the same environment and you get a poor user experience. The same thing will happen taking desktop/server Linux developers and putting them on Android.

CREEGER: Smartphones are not the only devices that we're talking about here. Not all mobile-specific devices are necessarily phones, such as iPads. How can you broaden this advice for those kinds of devices?

NEVILLE-NEIL: We have already been through this with the Palm Pilot, and in a lot of ways those lessons have been forgotten. When the Palm Pilot came out IT departments went nuts. A personal handheld device that contained a large proprietary address book and was subject to loss or inadvertent disclosure on an Internet site was not what they wanted to hear about. One should be careful about placing persistent proprietary data on a mobile device.

CHARLAND: I want to stress the minimum viable product approach: What value can we provide to our user base and can we do this in the mobile browser? The browser paradigm is a familiar concept to IT departments.

TOY: If possible, I will go with a browser-based application, but it is not always a viable choice. The only platform that allows you to keep your application behind your firewall is BlackBerry. Yes, you can run VPN (virtual private network) on the iPhone, but iOS locks up all your other applications. Plus, iPhone will not support two-factor authentication, which is becoming an industry requirement. While I agree that one should look at doing a browser-based application first, aka thin client, it's a challenging approach and will not always work. A lot of the time the juicy stuff you're trying to access with your thin client is on your intranet and behind your firewall. Today only BlackBerry gives you an easy path to get there.

REALINI: A thin client has inherent advantages if the network is powerful enough to support it. Right now in the U.S. we have huge network capacity issues. While you can talk about how thin clients can get by the firewall, there is the issue of whether the network is going to be fast enough to make that model viable.

I think the network problem in the U.S. will be fixed and will eventually be fast enough to handle the demand. So in the future, when everyone has a smartphone and the network's going to be fast enough, why wouldn't we all want thin clients?

NEVILLE-NEIL: You're going to have to battle for control. I want my data on my device and not on someone else's server. It makes perfect sense for sensitive corporate data not to be under my control, but it makes no sense for me not to have control over my own data.

REALINI: So, thin client implies that my data is in the cloud?

NEVILLE-NEIL: Yes.

CREEGER: And that means you're giving your data to Google or other data aggregators.

REALINI: Everyone should care, but I am not sure they will care as much as the technical community.

NEVILLE-NEIL: There are people who care, and more will care as more data compromises happen.

CREEGER: Is anyone pushing a fat-client approach today that focuses on the use of mobile-phone platform cycles?

TOY, NEVILLE-NEIL: iPhone.

REALINI: Do we agree that a thin client has inherent advantages in the right environment?

CREEGER: Today, a thin client is desirable because the cloud is in ascendancy and people are not sensitive to data security.

TOY: For the enterprise, personal data privacy is not a problem because it is not your data; it belongs to the company. Enterprise IT guys are going to favor thin client. They want to keep the company's data inside the data center to control access better, including revocation.

BOURNE: As mobile devices become more ubiquitous, I don't see how existing wireless carriers in the U.S. will be able to make the required capital investment to handle the increased demand for services. Certainly that is the case in the next year or two. Cellphone data transport is limited, and in the U.S. at least, carriers are not making great money on those services. How does Wi-Fi as an alternative transport layer fit in?

NEVILLE-NEIL: The urban U.S. usually has good Wi-Fi coverage. Practically all mobile devices have Wi-Fi, and people building applications would be crazy not to take advantage of that.

For the purposes of authentication, cellular phones are attractive as each one has a hard-to-duplicate ID. Plus, there are many things a carrier can do to secure data across a cellphone network that cannot be done with random Wi-Fi access points. Lastly, when you touch a Wi-Fi access point, unless your data is encrypted, everybody else is touching your data as well.

REALINI: If wireless networks don't get better, will we get to the point where smartphones are really just connected Wi-Fi devices?

My iPad is useless as a connected application, and I have stopped using it because it is too slow for some applications. If we have a situation where users have powerful devices but the network is unreliable, they will learn to roam on Wi-Fi in the same way Africans learned to carry two SIM chips.

If that becomes standard practice and carriers don't solve the problem, the cellular network will diminish in importance. People will defect from their networks and start connecting

to Wi-Fi. We'll see a shift from cellular devices to Wi-Fi devices.

CHARLAND: Regardless of whether you are doing browser-based or native mobile applications, you have to design them for a spotty connection. You can't always assume there is network connection, nor should you think that there's never a network connection.

CREEGER: What are the most important issues you would stress to our readers?

REALINI: I would stress: (1) If you don't already have seasoned, in-house, mobile expertise, rent or buy it but don't try to grow it organically. (2) Be prepared to deal with a highly fragmented environment. (3) Do your best to define what you will and will not do. In mobile, one has the opportunity to achieve huge scale if the right things are done on the right devices in the right way. Making a mistake means fragmentation and getting bogged down. (4) Expect dramatic change all the time. Along with fragmentation, mobile is moving at a much faster rate than one sees in IT. (5) As you plan to develop new software, continually ask what the market is going to look like in 6 to 12 months so you know what you are getting into.

CHARLAND: (1) Define the minimum viable product for internal and external customers. (2) Consciously choose which devices you have to support. Don't just say all; do the market research, look at the market trends, talk to customers. (3) Determine the cross-platform user experience; then pick the solution that allows the design to get close to a single application. No application is ever totally cross platform. You will have differences and they should be documented. (4) Determine if the application can function in a Web browser (including HTML5) for the devices being supported—if not today, then in the future (check W3C and other standards bodies). Also, research whether a hybrid approach such as PhoneGap is feasible. (5) Determine your plan to test all the different devices on the different carriers. It is barely good enough to buy all the devices and have one individual who can test on every device. You'll need either to have a more comprehensive testing plan or to hire a third-party testing service.

TOY: (1) Trying to make everybody happy is an unsolvable problem. One needs to define tiers of service and decide how many tiers will be supported. (2) Define the key issues that are important to your business—functionality, security, and ubiquity are three good concerns to start with. (3) For each of your tiers, define the level of resource devoted toward the support of each issue and the devices you will be supporting at that level. Trying to make every device tier support every device at the maximum level is a recipe for failure. It's fine to say that the CEO must use only a BlackBerry and cannot use an iPad to access important documents. It is also fine to say that someone further down the security stack can just synchronize an iPhone. Applications must fit into a dimension of that tiering, and perhaps people lower on the security stack just don't get certain applications, or any applications.

CREEGER: How do you suggest IT manage and track information service consumption and the threat environment?

TOY: One way is to go thin and keep information assets behind a firewall in much the same way folks have done with thin desktops. Alternatively, you can emulate what folks have done with laptops and install end-point security products to impose control directly on the device.

NEVILLE-NEIL: You have to think about what data is important to your business and its continuity. You need a disaster recovery plan that is sensitive to different types of disasters. You have to decide which data goes where, on which device, and to which people. Most computer security is trying to decide those questions, and the same will be true in mobile as well. Also, use the oldest, most established, and minimal API set possible. It will just make your life much easier in the end.

CREEGER: Does anybody have any idea what the world is going to look like in two to three years?

TOY: Tablets are the biggest workplace changer in the next two to three years inside a company and on the Internet.

CHARLAND: While I don't think native applications will ever go away completely, from a developer's perspective, the majority of applications will be Web-based. Tablets will play a bigger role, but they will blend more with laptops, and I feel the phone will always play a bigger role than the tablet.

NEVILLE-NEIL: We're going to see more splitting of the network space. More people will have personal area networks, accessing a MiFi, their phones on the cellular network, whatever. You're going to see devices talking to each other a lot more.

Applications will move from the phone to the tablet. The tablet will become the primary consumption device for media and the sweet spot for consumers. I think kids will lead the way.

In the corporate space, you won't see consolidation around Android or iOS, and both will maintain a varying percentage of the marketplace unless or until somebody produces a new game-changing killer device.

Lastly, we'll have a lot more thin client in the enterprise space. It's just an easier way to control access to data.

REALINI: Today, companies interact with customers primarily in person or on the Web. In the future, mobile is going to be the most important way those interactions take place. Smartphones will become richer and more powerful, because we're going to expect and demand it. It's only natural that a lot of customer-facing applications will be mobile. I think mobile is going to fundamentally change the types of services that can be delivered; how efficiently those services can be provided; and what types of customers can be engaged. I think mobile will create vast new markets to broaden the reach of commerce way beyond its traditional scope. C

Related articles on queue.acm.org

Mobile Media: Making It a Reality
Fred Kitson
http://queue.acm.org/detail.cfm?id=1066066

Four Billion Little Brothers?: Privacy, mobile phones, and ubiquitous data collection
Katie Shilton
http://queue.acm.org/detail.cfm?id=1597790

Mobile Application Development: Web vs. Native
Andre Charland, Brian LeRoux
http://queue.acm.org/detail.cfm?id=1968203

DOI:10.1145/1995376.1995393

Establish a global cyber "neighborhood watch" enabling users to take defensive action to protect their operations.

BY STEPHEN J. LUKASIK

Protecting Users of the Cyber Commons

CYBER PROTECTION HAS long been a concern; recall the Morris worm in 1988, widespread use of the commons with the introduction of commercial email and Web browsers in the early 1990s, and the U.S. Presidential Commission on Critical Infrastructure Protection (PCCIP) in 1996.[11] A Google search yields more than 43 million articles dealing with computers and networks. This much attention, without dependable security for users, leads one to wonder why the problem persists. Are computer vulnerabilities growing faster than measures to reduce them? Perhaps the problem is not purely a technical matter, but more to do with users. Carelessness in protecting oneself, tolerance of bug-filled software, vendors selling inadequately tested products, or the unappreciated complexity of network connectivity have led today's abuse of the commons.

However, among potential remedies, current U.S. government-led approaches appear to be going at them piecemeal, fixing those that demand immediate attention. Since this approach is not keeping up, it may be useful to rethink it, seeing if there are strategic directions more likely to deliver benefits.

Protecting users of the cyber commons, nationally or globally, has both top-down and bottom-up aspects. Calls for government action to "protect cyberspace" relate to top-down processes that, while identifying drivers of policy, wash out lower-level detail. That is the way governments think and what people have come to expect from them. Protecting a national commons would appear little different from other aspects of national security, which is clearly a government responsibility. In the U.S., under the recently organized Defense Department Cyber Command, the National Security Agency has been designated as the U.S. cyber force,[4] including both the 24th "Air Force" and the 10th "Fleet," in quotes because neither is a conventional flying nor floating combat unit, consisting instead of people at computers, the newest element of net-centric warfare.

Bottom-up processes are equally important; they are what "really happens," the way processes work, rich in detail, but leave some major drivers of events invisible. The difference between the two perspectives—top-down and bot-

» key insights

- **Top-down processes (such as regulation, national strategies, federal funding, and international agreements) protecting users of the cyber commons operate far more slowly than offensive and defensive technologies.**
- **Bottom-up processes (such as the affinity groups that characterize social nets) take advantage of the character of public networks, offering additional defensive options to protect them from abuse.**
- **These processes mimic how the ARPANET was created, contribute to network evolution, and share the concept behind the IETF and other volunteer network mechanisms.**

tom-up—is the same as between legislation and how complex implementation rules perform in practice; complete descriptions include elements of both.

Threat Reduction

First, what threats against whom should be reduced? Starting with the universe of all users of the cyber commons worldwide, illustrative groups can be identified that share common security requirements. As sovereign states, governments have considerable latitude and resources. Infrastructure operators and communication carriers are together a particularly powerful group when they feel they have liability, responsibility, and authority. State, county, and local governments have responsibility but often lack resources, financial or human. Large private organizations have significant financial and human resources if they define the defense of the cyber commons as a sufficiently high priority (see Figure 1).

While government programs are easily justified when targeting specific sets of users for particular purposes, they leave the rest of us to fend for ourselves.

A recent National Research Council committee report examined a number of research areas relating to cybersecurity,[5] offering a cybersecurity "bill of rights" that defines these user expectations:

- Availability of system and network resources to legitimate users;
- Convenient recovery from successful attacks;
- Control over and knowledge of one's computing environment;
- Confidentiality of stored information and information exchange;
- Authentication and provenance of information;
- The technological ability to exercise fine-grain control over the flow of information in and through systems;
- Security using computing directly or indirectly in important applications, including financial, health care, and electrical transactions, as well as in real-time remote control of devices that interact with physical processes;
- The ability to access any source of information safely;
- Awareness of the security being delivered by a system or component; and
- Redress for security problems caused by another party.

While one might complain, the typical user is far from enjoying these "rights" in the cyber domain, and how to achieve them in a global commons is by no means obvious. They are perhaps more like stars to navigate by than places one can expect to reach.

Top-Down Perspective

Possible defensive actions cover at least four dimensions: mandatory protection of cyber domains essential to the economic health and quality of life; national strategies, plans, and programs, helping coordinate protection of the commons; international legal regimes and their supporting international structures, encouraging and assisting defense of the commons; and technology to warn, prevent, and thwart misuse of the commons.

There is no silver bullet. The amount and types of protection varies with individual jurisdiction and time, as adversaries and technology change and attackers refine their attacks and redefine their goals and targets.

Mandatory protection. In the U.S., regulation of private domestic activities is a function of each of the 50 states, intending to enhance public safety, increase reliability, maintain law and order, and protect citizens from exploitation. Government-owned infrastructure should be subject to the same regulation, but the governments regulate themselves and thus have some flexibility compared to private operators. Those aspects of the infrastructure on which the public depends require mandates through the agencies responsible for their oversight.

Some see regulation as a restriction on the efficient operation of markets and as foreclosing potentially beneficial options. These concerns notwithstanding, there is agreement that critical infrastructure services merit some degree of regulation. A central issue is how to define "critical" and how much

regulation is enough.

Regulation implies restrictions on the operation of markets, possibly foreclosing potentially beneficial options. There is general recognition that infrastructure services merit some degree of regulation to protect against inequitable access to service and the abuse of what can be natural monopolies. Deciding what to protect defines what not to protect. By default, the latter are left to market forces. The decision of what to regulate should hinge on the allocation of resources to provide the greatest protection to the greatest number of people. This requires analyses of users, their relevance to national goals, and the interdependencies among their needs. What we currently have in the U.S. is mandated protection of central infrastructures and national security assets, with the rest dependent on market forces to balance security, cost, and convenience.

In 1997, the PCCIP identified eight critical infrastructures, and, in preparing for the expected disruption of computers at the beginning of 2000, the U.K. identified 11 critical infrastructures as central to the operation of society[2]; the European Commission also identified 11, though they differed from other lists.[1] If one looks for the infrastructures common to such lists, along with factoring in estimates of their interdependence, three emerge: telecommunications, electric power, and transfer of funds.

Infrastructures depend on the reliable transmission of information for their operation. If one is to protect any part of the cyber commons, the command-and-control mechanism of critical infrastructures is part of what should be done.

An example of how to protect critical infrastructure is provided by the Federal Energy Regulatory Commission (FERC), the regulator of the U.S. electric-power system, consulting with and coordinating its regulatory actions with industry groups, including the North American Electric Reliability Council (NERC). The FERC Final Rule, issued in 2008 after a rule-making proceeding, is a useful starting point.[3] While heretofore reliability was treated as desirable, and outages were reported to FERC and analyzed by NERC, the requirements on the industry were flexible. The Final Rule detailed actionable security processes for infrastructure protection that recognize both the realities of computer technology and the tendency of regulated entities to cut corners.

Regulators attempt to force a desired level of performance, while regulated entities deploy armies of lawyers to thwart them by bringing suit against the regulator. Regulatory actions, whether originating in independent regulatory agencies chartered by the U.S. Congress or by agencies established within the executive branch, under the separation of powers in the U.S. government, are subject to review by the federal judiciary. The judicial system and its due-process requirements are thus the final arbiter of regulations. The traditional paths to circumvent regulation are to claim the need to exercise reasonable business judgment, maintain that a higher level of risk than provided for in the regulation is adequate, and challenge the technical feasibility of the regulation.

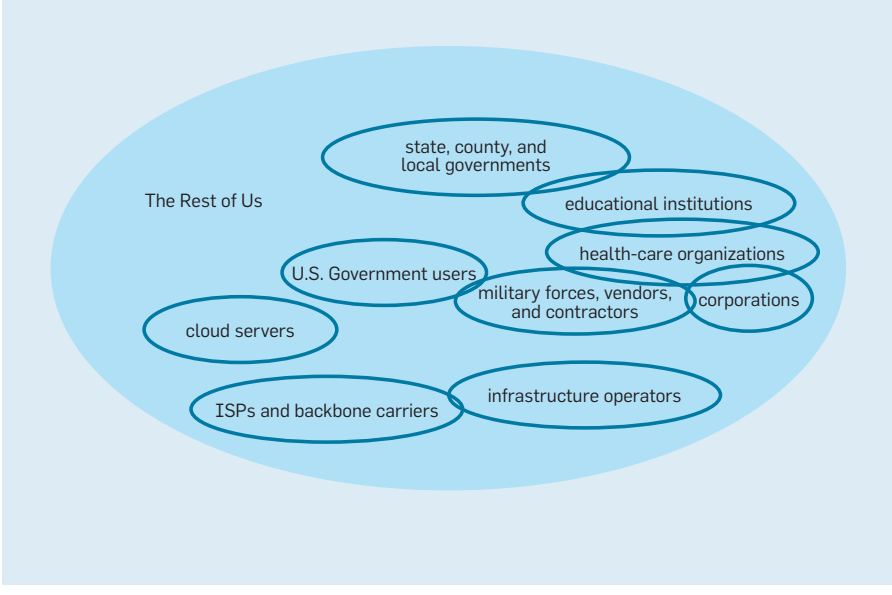
The FERC order is firm in blocking such arguments. With regard to business judgment, the Report said the Commission noted in the Critical Infrastructure Protection Notice Of Proposed Rule-making (CIP NOPR) that "Cybersecurity standards are essential to protecting the Bulk-Power System against attacks by terrorists and others seeking to damage the grid. Because of the interconnected nature of the grid, an attack on one system can affect the entire grid. It is therefore unreasonable to allow each user, owner or operator to determine compliance with the CIP Reliability Standards based on its own 'business interests.' Business convenience cannot excuse compliance with mandatory Reliability Standards."

Regarding the second tactic of evasion—operator willingness to accept risk—"The Commission continues to view the term 'acceptance of risk' as representing an uncontrolled exception from compliance that creates unnecessary uncertainty about the existence of potential vulnerabilities. Responsible entities should not be able to opt out of compliance with mandatory Reliability Standards. The Commission, therefore, directs the ERO [Electric Reliability Organization] to remove acceptance of risk language from the CIP Reliability Standards."

Finally, regarding technical feasibility, the Final Rule said: "The Commission adopts the CIP NOPR proposal and directs the ERO to develop a set of conditions or criteria that a responsible entity must follow when relying on the technical feasibility exception contained in specific Requirements of the CIP Reliability Standards... We note that the Commission did not propose to eliminate references to technical feasibility from the CIP Reliability Standards, only that the term be interpreted narrowly and without reference to considerations of business judgment."

The Congress attempted to extend the proceeding as far beyond the elec-

Figure 1. Users of the cyber commons.



tric-power system as possible, but the Commission drew the line at its defined authority, saying: "The Commission is sensitive to the concerns raised by the Congressional Representatives regarding the severe impact that a cyberattack on assets not critical to the Bulk-Power System could still have on the public. The Commission, however, believes that its authority under section 215 of the FPA [Federal Power Act] does not extend to other infrastructure. Section 215 of the FPA authorizes the Commission to approve Reliability Standards that 'provide for the reliable operation of the bulk-power system,' defined by the statute as the facilities and control systems necessary for operation of an interconnected electric energy transmission network and the electric energy needed to maintain transmission system reliability. In addition, section 215(a)(1) specifically excludes from the definition of Bulk-Power System 'facilities used in the local distribution of electric energy.'"

The most significant change in behavior attempted by FERC involved the matter of trust, saying: "The Commission proposed in the CIP NOPR to direct the ERO to modify Reliability Standard CIP-003-1 to provide direction on the issues and concerns that a mutual distrust posture must address to protect a control system from the 'outside world.' The Commission noted that interconnected control-system networks are susceptible to infiltration by a cyber intruder and that responsible entities should protect themselves from whatever is outside their control systems... The Commission noted that a mutual distrust posture requires each responsible entity that has identified critical cyber assets to protect itself and not trust any communication crossing an electronic security perimeter, regardless of where that communication originates... Mutual distrust does not imply refusal to communicate; it means the exercise of appropriate skepticism when communicating. The Commission believes additional guidance on what this means specifically in current practice would help responsible entities to avoid these misunderstandings... The Commission therefore directs the ERO to provide guidance, regarding the issues and concerns that a mutual distrust posture must address in order to protect a responsible entity's control system from the outside world."

Such injunctions amount to saying that from here on you must take seriously cyber and other attack threats to reliability, and not ignore them when inconvenient. While it is still too soon to know how effective this new approach to infrastructure cybersecurity will be, one conclusion is that even in a strongly deregulatory environment, regulatory bodies can provide legal handles on cybersecurity in regulated entities otherwise lacking in most other parts of the cyber commons.

A last-resort approach by a regulated entity seeking to minimize the effect of regulation is to minimize its domain of applicability by excluding from the FERC order as much of the generation, transmission, and distribution assets as they can get away with by declaring them non-critical. This is, of necessity, a continuing area of contention, as new technology is adopted and new energy needs are identified.

A recent study by the Center for Strategic and International Studies also considered whether effective cyber defense can be provided by current methods or whether fundamentally different approaches must be explored.[12] Sponsored by the House Homeland Security Subcommittee on Emerging Threats, Cyber Security and Science and Technology, it made two proposals—regulation and identity management—that have long been sidestepped or rejected by most groups dealing with the problem. It said: "We believe cyberspace cannot be secured without regulation." Of its 25 recommendations, six related to actions that should be required of infrastructures overseen by regulatory agencies or the authentication practices required of critical infrastructures, including: allowing consumers to use government-issued identity credentials; requiring all businesses to adopt a risk-based approach to credentialing; and encouraging risk-based processes over specific prescriptions.

The proposal concerning regulation of digital identities would eliminate anonymity from users in order to facilitate accountability for actions in the cyber commons. This is no different from identifying taxpayers or displaying a license plate on a vehicle. However, the downside could be elimination of the use of the net for political protest, an otherwise important benefit. This could be addressed by providing for unlicensed users, not unlike how unlicensed electromagnetic spectrum is allocated, with the understanding that no liability would be incurred by and no accountability would be expected of its users. Acceptance of communications from unlicensed users would be at the receivers' risk.

Regulation is necessary for protecting important parts of the cyber commons and a necessary tool for protectors. But one must recognize that the entities so regulated will accept it only after avoiding it through every possible legal and political channel available to them.

National strategies. Another necessary government role in protecting the commons goes beyond protection of their own internal users and computers. This is a national leadership role enabling and coordinating pri-

vate actions. Governments also play an implementation role in proposing legislation, enforcing mandates, and protecting users of the commons too small or weak to function effectively on their own behalf.[10]

While the U.S. government relies on public-private partnerships to achieve many of its goals, the degree to which network security is worsening suggests the need for new mechanisms. Since commercial organizations see computer security as a cost and do not value the corresponding benefit, private efforts have to date been insufficient. Both sides of the partnership are failing to stem the tide of abuse of the commons.[7]

Efforts by President Barack Obama and his Administration suggest this posture may be changing. In 2009 remarks, Melissa Hathaway, then acting senior director for cyberspace at the National Security Council, representing the National Security and Homeland Security Councils, said, "The Federal government cannot entirely delegate or abrogate its role in securing the nation from a cyber incident or accident. The Federal government has the responsibility to protect and defend the country, and all levels of government have the responsibility to ensure the safety and well-being of citizens."[6]

Though government leadership is necessary for protecting the nation from cyber abuse, it is indirect, with much distance between government-strategy documents and demonstrable security.

International mechanisms. Cyber abusers and their victims can be in different sovereign jurisdictions. Actions against violators are supported by common standards of unacceptable behavior. Rationalizing laws globally makes sense but is time consuming and eventually limited by the speed each country adapts to new technical, economic, and political circumstances.

For international agreement to be effective, implementing mechanisms are needed for accommodating changes suggested by evolving needs: monitoring compliance by the signatories to maintain their trust and confidence; enforcing the agreement should signatories depart from agreed-upon norms; resolving disputes among the signatories; addressing technical issues of definitions, standards, and forensic collection; and rendering assistance to signatories to respond to technical challenges expeditiously. However, this process is also slow, as diverse signatories must be convinced they need to take action.

While many protective steps can be taken without formal agreement, if global changes in security are to be achieved, a larger international framework will be necessary for facilitating cooperation among signatories; drawing from common international contexts, Sofaer and Goodman[13] discussed elements of such a framework.

As with the previous three dimensions of a framework for cybersecurity, international organizations have a role to play but, like regulation and government strategy, find it difficult to respond to the needs posed by a dynamic technology environment and aggressive and quick learners among those who would abuse the commons.

Technology to limit abuse. The view of many is that today's lack of security of the commons and its information is no more than a bump on the road of technical progress, fixable by layering on more and better technology. Using technology to fix technology is questionable as a response to a problem with roots deep in the growing complexity of the worldwide network.

Were technology to change more slowly, such an approach might have a chance of success. Problems arise when unexpected coupling between parts of large computer-based networks of logical processes exhibit behavior that, while following precisely from their programmed logic, cannot be completely anticipated. Large networked systems have so many internal states they can never all be exhaustively tested, and proving their security appears unlikely.

Technology creates new power through enhanced performance in terms of size, speed, bandwidth, capacity, connectivity, and functionality, but, even as it "fixes" old problems and improves functionality, the technology creates new problems, embedding them deeply within unverifiable systems. The matter is one of relative rates of change. If problems are fixed more quickly than new problems are created, one can imagine achieving a stable balance. But when new technology introduces new problems more quickly than it fixes old ones, the resulting divergent situation defies control.

Malevolence threatening the cyber commons introduces a new rate-of-change parameter. Attackers quickly reverse-engineer security alerts and patches to exploit related flaws before defenders can eliminate them. The defender fix-install rate must be faster than the attacker reverse-engineering rate.

Cloud computing is a current example of technological exuberance. Users are encouraged to move their information and applications from machines under their direct inspection and potential control and which could

conceivably become adequately secure into a "cloud" of networked computers of unknown ownership, location, management, and security. Should users enquire of the cloud's gatekeepers about such matters, they are told to "trust us," though one can hardly refrain from asking, "But why should I?"

Technology is an enabler for the first three necessary components of protection of the commons but like the others is insufficient. It is both part of the problem and part of the solution. Most important, behavioral adjustments by users of the commons are also needed to break the cycle of self-destructive technology:

Connections. Users should revisit the premise that any two devices are better connected than unconnected;

Conceptual errors. Managers should recognize that entrusting the fixing of flaws to the people who created them has natural limits, and that, perhaps, the security problem is not a matter of minor execution errors but of major conceptual errors;

Any computer. Decision makers should recognize that any computer can be penetrated, just as any building can be entered and any object can be stolen; and

Distrust as default. All users are well advised to replace trust with distrust as a default condition in all computer-mediated interactions.

These should not necessarily deter technical innovation but call for adjustment in the expectations of managers and users of the technologies they adopt.

One must recognize that entities so regulated will accept it only after they have avoided it through every possible legal and political channel available to them.

Bottom-Up Perspective

Voluntary legal user-controlled, self-defense efforts are also necessary but inherently on a smaller scale than their governmental counterparts. They are most easily accomplished when user organizations are large enough and smart enough to identify and implement cost-effective protection. They help establish a market for protection technologies and educate a new generation of security professionals who understand options and risks that often remain classified or proprietary and are difficult to share widely.

Voluntary self defense asks: Who does the volunteering and the defending? The answer depends on the technical knowledge available to users and the resources they can devote to something that is not their professional focus. The newly emerging popularity of informal social networks points to an alternative to top-down processes.

Voluntary user-oriented mechanisms (such as the Internet Engineering Task Force, or IETF) have served the Internet well, developing protocols to provide greater security and fostering next-generation networks.[9] Computer emergency response teams (CERTs), industry-information-sharing-and-analysis centers (ISACs), informal regional system-administrator groups, software vendors, and the Forum of Incident Response and Security Teams (FIRST) all help but have difficulty staying ahead of aggressive attackers.

How can voluntary defense establish a trust mechanism? The seeds of today's Internet security problems were planted when the ARPANET began to grow beyond its first small circle of researchers more than 40 years ago.[8] Early generations of network users were homogeneous, scientifically oriented, cooperative, dedicated to developing network technology and its applications, and had no reason to distrust or harm one another. With net growth has come many more users with no knowledge of one another and with divergent agendas. Distrust should replace trust, but the means of practicing distrust are poorly served by network technology created to support trusted users.

The National Strategy to Secure Cyberspace published in 2003 relied on the 1997 PCCIP principles: voluntary action, public-private partnerships, public awareness, international cooperation, and the central importance of critical infrastructure.[14] It viewed cyberattacks as crimes for which, through due process, perpetrators would be identified, prosecuted, and punished. Vulnerabilities were to be reduced through an unending search for flaws and their elimination through decisions by vendors, service companies, and computer owners and operators. It presumed software flaws could be reduced over time to acceptable levels. The defensive concept was to distribute response capabilities to user organizations acting on their own behalf and in their own best interests.

The security problems experienced

today are significantly greater than when PCCIP issued its recommendations. The fixes are not working.[7] There is heavy reliance on government and foot-dragging over what organizations will be forced to do. Another factor is the deep-seated view that security goals cannot be achieved without significant federal R&D funding. While time has been devoted to negotiating treaties related to cybercrime, nations use the delay to strengthen their cybersystem penetration capabilities for intelligence collection and to develop the means for conducting cyberwar, aka "information operations."

Law-enforcement paradigms do not address rapidly evolving threats well and fail under emergency circumstances. The prospect of zero-day attacks, enabled by current trends in viruses that evolve quickly and an aggressive malware industry, are relevant. Changes in the nature of zero-day threats, the uncountable vulnerabilities of systems, and the motivations of cyberattackers require warning systems to detect attacks with enough time to initiate protection responses. Protection must be managed in near-real time so at least some attackers are thwarted. However, real-time warning and response must be on a global rather than a local basis.

One possible way of doing this exploits the nature of self-organizing social networks, starting with the proposition that users have a role in leading efforts for their own protection, not simply accepting what others choose to do, or not do, on their behalf.

Social networks have two characteristics that mimic development of early networks: respond directly as participants perceive value, growing in directions and at rates determined by that value; and overhead costs, typically riding on the Internet, where users pay for access and where participating Web sites may be supported through advertising income. Some central management is needed to maintain the integrity of the social network. Illustrative of the informal yet resilient nature of such networks are Facebook rules to protect privacy, open source software, user-created wikis, and apps purchased from developers through commercial sites.

Commons Protection Union

Proposed here is what might be called a Commons Protection Union (CPU) or, perhaps, cyber "neighborhood watch," to recognize attacks in real time and provide information to users or their service-provider proxies, enabling them to disconnect from parts of the commons to contain a "disturbance" until it can be analyzed for its origin and characteristics and systems restored to full connectivity. Since cybersecurity problems derive from connectivity, managing connectivity is likely part of the solution.

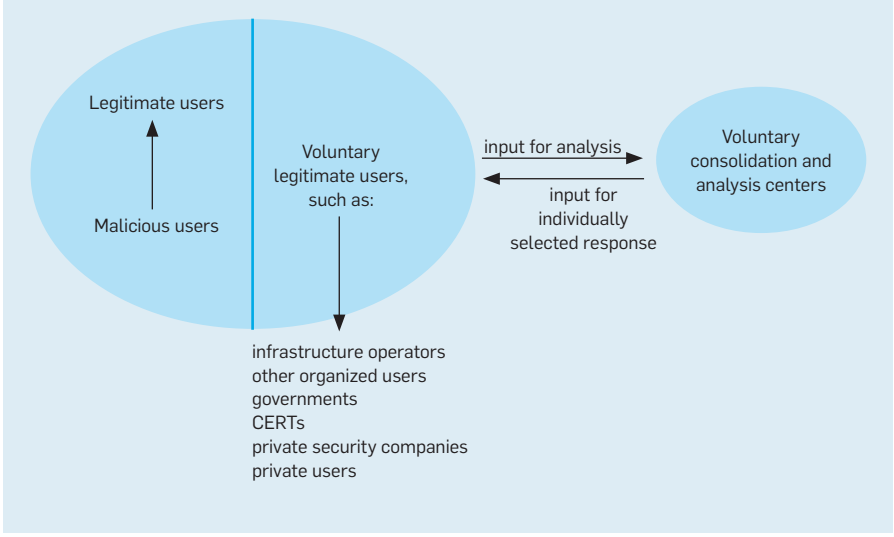
Operating such a function can be done more responsively than is possible when response to attacks is paced by the rate of adopting intergovernmental agreements and the implementation speed of national response agencies. A flexible, voluntary approach is required, free of contested mandates. Being open and voluntary, governments could participate in increasing their effectiveness to whatever degree they choose. Real-time event information from users, private security companies choosing to participate, and such public information as governments choose to contribute could enable distributed examination of malware and attacks and provide information to participants for quick analysis.

The arrangement would make attack and ongoing probe information available for the common good, the essence of a commons. On the basis of such real-time information, participating users could take such defensive actions as they choose; for example, they could reduce load, route around congestion, disconnect from parts of the net, collect and preserve forensic information, and increase their hardness level, depending on their assessment of the real-time threat level and the criticality of their operations.

Carriers and Internet service providers do some of this. The new elements would be voluntary sharing, global real-time data provided to users or their proxies, and trusted third parties as consolidators. The high-level nature of the traffic monitoring can be designed to yield statistical measures for automated diagnostics and decision making while respecting the privacy constraints placed on the information by its contributors. Global traffic monitoring would include parameters to assess flow pathologies and detect anomalous patterns. What is proposed is not unlike a missile-launch-detection-and-tracking system but in which the defensive components are distributed and under user control.

How might such an addition to the computer- and network-security environment be brought about? The same way many activities on the Internet begin; someone creates something of value, and it spreads without prodding. Such an approach can potentially spread at the Internet speed of social networks rather than at government speed. As outlined in Figure 2, the upper-left oval represents the Internet, with legitimate users dealing with other legitimate users, but, now, malicious users inject themselves

Figure 2. Commons Protection Union: a social network.



into it, only masquerading as legitimate users. The CPU is authorized to receive the externals of such traffic, as the voluntary users have authorized; these data streams are analyzed through the voluntary actions of those participating in the CPU social network for anomalies that can indicate a cyberattack or preparation for a cyberattack. The members of the CPU network send statistical information or alerts of varying degrees of urgency to its contributors who are then able to initiate defensive responses, depending on the nature of their information to be protected and the criticality of their operations.

The process is characterized by various operational and business models, several supported by distributed agents. Consolidation and analysis centers CAC(s) would receive traffic externals from user sources, including infrastructure operators and other organized entities. They would also receive hierarchically processed flows (such as EROs) for parts of the power infrastructure, nodes in upper levels of communication systems, feeds from CERTs, network-security companies, and, most important, private and small-business users. Governments are likely to have their own systems for their needs but could participate with filtered flows should they choose. The CAC(s) could provide near-real-time alerts and network status reports to users, with lengthier analyses following as more data is analyzed.

CAC(s) might be organized as a not-for-profit corporation supported by user consortia consisting of network-affinity groups, possibly as a subscription service with various levels of timeliness and depth of analysis. Amateurs perform similar services, including ham-radio operators in emergencies, astronomers searching for asteroids, and gamers exploring approaches to protein folding. It could be a research operation studying network dynamics while also providing a real-time product, an objective that would also provide useful guidance for research. Output data could be used as a basis for for-profit value-added services. There is even a civil-defense aspect governments might support.

The basic governance principle, as with the IETF, would be openness, rough consensus, and running code to be improved collectively over time.

Following any of the paths outlined here, a social-network-based CPU will develop in directions its users feel provide value. Existing social networks (such as Facebook, Twitter, blogs, and wikis) could provide marketing and distribution channels.

Further issues will also have to be addressed, as with any user-controlled network. Participants have to choose between privacy and the degree to which the network demonstrably improves their protection. The CPU's own protection is necessary to prevent it being manipulated by the abusers whose activities it seeks to mitigate. A CPU could also give network abusers feedback on the effectiveness of their attacks, but attackers already know the responses being taken by software providers and security vendors.

The voluntary technical contributions needed for its operation will have to be forthcoming from the participant community. The degree to which a CPU competes against the security products of its commercial participants will have to be balanced against the benefits they would receive.

It may be that the most capable and dedicated security innovators are found in the same research community that formed the basis for the ARPANET. Such an experiment would be worth trying.

Acknowledgments

I benefitted greatly from my discussions on improving cybersecurity with Seymour E. Goodman and Anthony M. Rutkowski. This study is based on a grant from Science Applications International Corporation to The Center for International Security, Technology, and Policy at the Georgia Institute of Technology.

References

1. Commission of the European Communities. Brussels, Nov. 17, 2005.
2. Ernst & Young. Y2K study, Aug. 1998.
3. Federal Energy Regulatory Commission. *Order No. 705: Mandatory Reliability Standards for Critical Infrastructure Protection, Docket No. RM06-22-000*, Jan. 18, 2008; http://www.ferc.gov/whats-new/comm-meet/2008/011708/E-2.pdf and http://www.ferc.gov/industries/electric/indus-act/reliability/cip.asp#skipnavsub
4. Gates, R.M. *Secretary of Defense Memorandum: Establishment of a Subordinate Unified U.S. Cyber Command under Strategic Command for Military Cyberspace Operations*, June 23, 2009.
5. Goodman, S.E. and Lin, H.S., Eds. *Toward a Safer and More Secure Cyberspace*. National Academies Press, Washington, D.C., 2007.
6. Hathaway, M. Keynote at RSA Conference: *The Obama Administration's Cyberspace Policy Review* (San Francisco, CA, Apr. 22, 2009).
7. Internet Crime Complaint Center, Federal Bureau of Investigation. *2007 Internet Crime Report*. National White Collar Crime Center, Bureau of Justice, Department of Justice, Washington, D.C., 2007; http://www.ic3.gov/media/annualreport/2007_ic3report.pdf
8. Lukasik, S.J. Why the ARPANET was built. *IEEE Annals of the History of Computing* (Sept. 2011).
9. Lukasik, S.J. *Protecting the global information commons telecommunications policy, Next-Generation Internet Conference* (London, Feb. 21–23, 2000); http://www.cistp.gatech.edu/publications/
10. Lukasik, S.J., Goodman, S., and Longhurst, D. *Protecting Critical Infrastructures Against Cyber-Attack, Adelphi Paper 359*. International Institute for Strategic Studies, London, 2003.
11. President's Commission on Critical Infrastructure Protection. *Critical Foundations: Protecting America's Infrastructures*, report. The White House, Washington, D.C., Oct. 1997.
12. *Securing Cyberspace for the 44th Presidency*. Georgia Tech, Atlanta, GA, Dec. 2008; http://www.csis.org/tech/
13. Sofaer, A.D. and Goodman, S.E., Eds. *The Transnational Dimension of Cyber Crime and Terrorism*. Hoover Institution Press, Stanford University, 2001; see Lukasik, S.J., Chapter 4: Current and future technical capabilities.
14. The White House; http://www.whitehouse.gov/pcipb/cyberspace_strategy.pdf

Stephen J. Lukasik (steve@gnsl.org) is Distinguished Senior Research Fellow at the Center for International Strategy, Technology, and Policy of The Sam Nunn School of International Affairs at the Georgia Institute of Technology, Atlanta, GA.

DOI:10.1145/1995376.1995395

Technologies are available to unlock radio spectrum as consumers need it.

BY CRAIG PARTRIDGE

Realizing the Future of Wireless Data Communications

WIRELESS WILL PLAY an even greater role in future data communications than it does today. For ubiquity of service and ease of connection, wireless is unmatched as an access protocol and seems poised to be the primary means by which people and machines access the Internet and its successors.

Wireless technology is in the midst of an important stage in its technical evolution—commercial transition from radios with behavior fixed in hardware to radios with behavior determined by software. This transition could enable far more flexible radios able to more fully exploit the radio spectrum to deliver data both faster and more reliably.

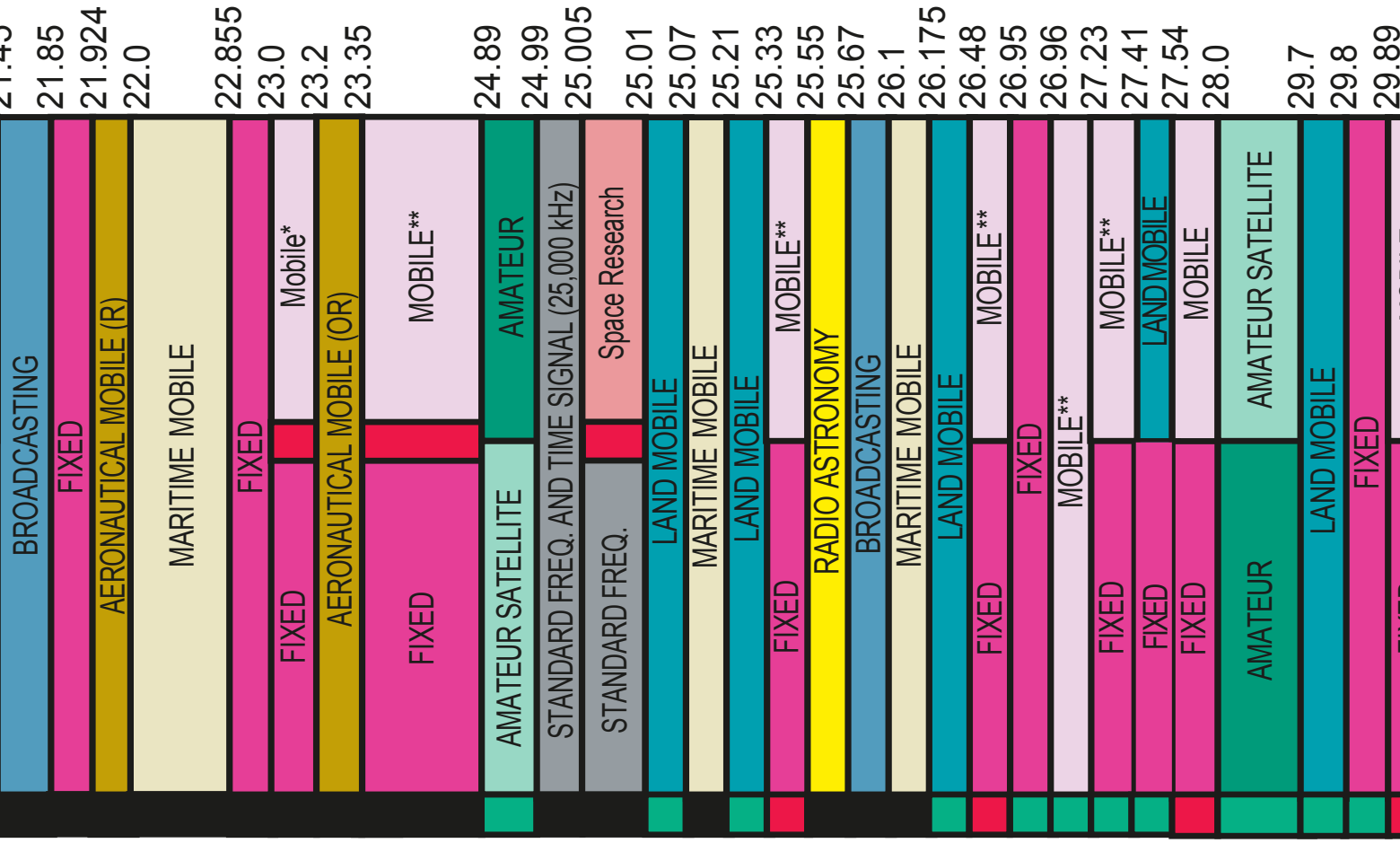
The research community has envisioned this moment since the early 1990s.[10] It is now here.

Unfortunately, computer science, radio engineering, and public-policy advocates are all imperfectly prepared to take advantage. Research on vital questions has been extremely variable, with wonderful work (such as finding the right mix of programmable hardware to support high-performance signal processing in radios) undermined by almost complete neglect (such as how to describe radio behavior independent of platform and how best to share spectrum). The research, funding, and public-policy communities have hard work to do if they are to realize the promise of wireless data communications.

Here, I sketch the technology path wireless data communications is on, as well as the opportunities the future wireless environment will bring, then

» key insights

- **Protocols (such as WiFi and Bluetooth) will be radio applets by about 2020.**
- **If an application needs more bandwidth, it can ask its radio to find capacity in unused spectrum.**
- **An important problem is how to ensure radios do not use spectrum inappropriately (such as to interfere with public-safety channels).**



Detail of U.S. frequency allocations of the radio spectrum.

focus on the critical research questions we need to examine to realize (or rule out) the opportunities and make the policy decisions that will drive our common wireless future.

From Hardware to Software

By about 2020 software radios[a] will have become the standard technology for commercial, as well as military, radios, employed in a range of devices, from battery-powered sensors and handheld devices to plugged-in devices (such as base stations). In software radios, all or virtually all functions, from the physical layer of frequencies and coding (such as *PSK and *-QAM) to the media-access layer (such as time-division multiplexing and carrier sense multiple access) are determined and can be changed in real time by software running in the radio.

Software radios are not new, having been around and seen as the future of military radios since the mid-1990s; they are slowly transitioning into the U.S. military today.[b] What is changing is their cost and packaging are reaching the point where they will also move into non-military markets.[6] In the mid-1990s a software radio was the size of a small refrigerator and could easily cost more than $100,000. A software radio today is the size of computer battery and costs as little as $500. Examples include the Wireless Network after Next (WNAN),[c] Universal Software Radio Peripheral (USRP),[d] and the somewhat more expensive Microsoft Research Software (Sora) radios.[e] Radio chipsets with some programmable features cost even less and are incorporated into consumer WiFi products (such as programmable base-station products from Picochip). Following today's trends, it is reasonable to expect that by 2020 fully programmable radio chipsets will be available at prices consistent with consumer products.

The importance of software radios is that they bring unrivaled flexibility; they are chameleons, running a telephony protocol (such as CDMA) one moment

a The definition of software radio is somewhat fluid and can be used to mean any of a variety of approaches to programmable radios, including cognitive radios, radios that limit their programmability to certain functions, and radios that use DSPs programmed in C vs. radios that use FPGAs programmed in VHDL. Insofar as is possible, this article uses the term generically to include all approaches.

b In particular, through the Joint Tactical Radio System (http://www.public.navy.mil/jpeojtrs/Pages/Welcome.aspx), a family of radios that conforms to a common hardware and software architecture called the Software Communications Architecture, or SCA.

c http://www.darpa.mil/sto/solicitations/WNaN

d http://www.ettus.com

e The Wireless Open-Access Research Platform, or WARP, (http://warp.rice.edu) is another notable platform widely used for research around the world despite being substantially more expensive than the other radios.

and a data communication protocol (such as WiFi) the next. This flexibility comes from the fact that the radio's behavior is determined by software. Furthermore, software control changes the pace of innovation. Making a change in the radio's behavior in the traditional hardware world means waiting six months or more for new hardware. In the world of software, change comes as quickly as a programmer can compile and debug, or overnight.

Living in a World of Software Radios

What is different about a world where software radios are the typical radio? The most obvious is that a consumer no longer buys a wireless protocol when buying a device. The notion that a PDA manufacturer would advertise support for Bluetooth or WiFi makes no sense in a world of software radios, as "Bluetooth" and "WiFi" would be applets any PDA could run. The focus will be on the PDA's radio processing power, expressed in digital signal processor (DSP) or field programmable gate array (FPGA) capabilities.

Recasting this observation as an illustrative scenario, suppose when people arrive in a foreign country their PDAs would automatically download and start running the right phone and data-communications protocols for that country. If the protocols change overnight, the PDA simply loads (wirelessly) the new versions in the morning. If the people go inside and want to use a local wireless network, the PDA downloads the protocols from the local base station, using, perhaps, WiFi as a legacy protocol to download the new protocols. All these steps happen without requiring any action by the PDA's user.

Another difference is available bandwidth. If an application needs more wireless bandwidth, it simply asks the radio for more. The software radio would then scan the wireless spectrum looking for unused frequencies and agree with its peer radio (such as the base station) to employ an unused frequency to provide the necessary bandwidth.

In this future world, software radios would offer consumers wireless communication not limited at time of purchase to a particular set of protocols and data-communications bandwidth not limited to a small set of overused frequencies.

They are chameleons, running a telephony protocol (such as CDMA) one moment and a data communication protocol (such as WiFi) the next.

The path to this future requires both technical and regulatory innovation and, as I aim to show here, the two paths are interlinked.

Types of Software Radios

Having sketched the future software radios have to offer, we need to consider how manufacturers will build them. At the moment it appears there will be a range of choices for constructing software radios. It is simplest to view this range from the extreme ends, where radios are near opposites in terms of their trade-offs.

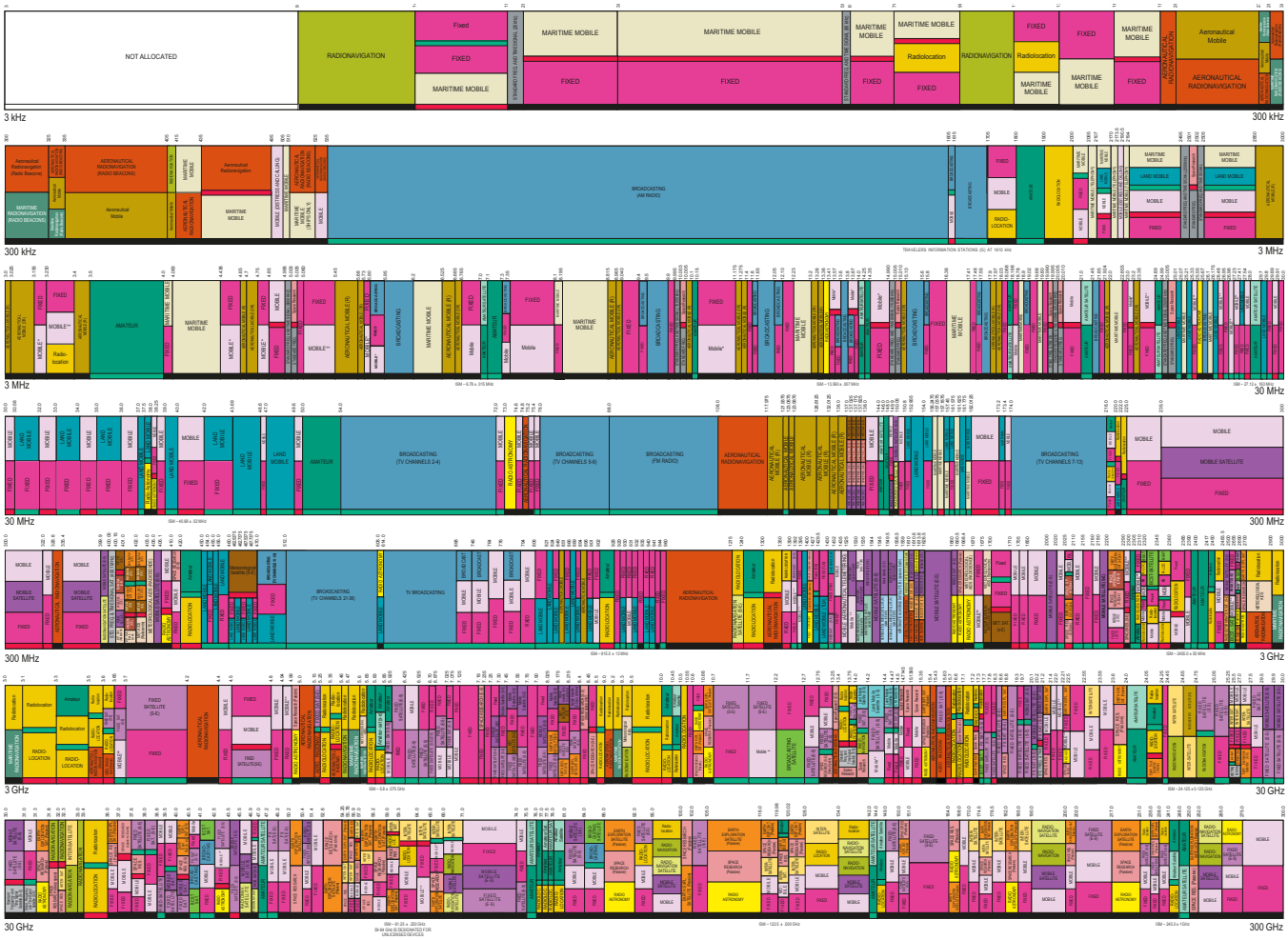
The first type of software radio is a collection of programmable components, mixing FPGAs, DSPs, and possibly an embedded processor. To program it, a software engineer writes or assembles a suite of software for the programmable components.

Observe that the mix of components varies widely. The central issue is how to provide enough processing power, often parallel processing power, to address streams of digital samples at the rates required for the frequency ranges covered by the radio's antennas.

Designers differ over how to best mix FPGAs, DSPs, and embedded processors to achieve the right processing power. There are also larger system issues; for instance, consider the filters used to select frequencies; better filters yield cleaner signals, which require less processing, but filters are more expensive than DSPs and FPGAs, so some systems choose less-good filters and more processing power. While there is still plenty of room to innovate, particularly in hardware accelerators that coexist comfortably with DSPs and FPGAs, the radio-engineering community understands this design space, as evidenced by a 2010 software radio design[4] that cited 43 references.

At the other end of this design space for software radios is a highly configurable chip or chipset. To program the radio, the software engineer would set configuration registers in the chip to determine what frequencies, coding, and media-access protocol features are used.

The conceptual difference between the two ends of the design space is stark. In the programmable radio, software



U.S. frequency allocations of the radio spectrum.

engineers' ability to implement a new data-communications protocol is limited only by their imaginations and the processing power of the components. In the chipset radio, software engineers are limited to the million or so permutations of frequencies, coding, and media-protocol features the chip supports. So, for instance, it is highly likely that engineers inventing a new form of coding will be able to implement it in the programmable radio and almost certain they cannot implement it in the chipset radio. Because programmable radios offer the widest experimental range, they are the preferred approach in most research efforts, but a chipset radio has important advantages, as discussed here.

Commercial chipset manufacturers appear headed for software radios between the two extremes. The chipsets will run software in DSPs or FPGAs for some functions and configurable (but not programmable) hardware for others. Indeed, some manufacturers in the cellphone industry have already mixed processors with chipsets.[f]

A natural question when one thinks about building software radios is whether they are "green." Because software radios use processors, one might assume they are substantially more power hungry than the traditional radios they replace. However, the situation is more nuanced, with much of the energy in a radio dissipated by analog components (such as amplifiers, filters, and antennas); in some radios, they are the primary energy cost. So, while the trade-off between a processor and customized hardware influences energy costs, it is not the only concern and in many radios is not even the primary concern.[g]

Radio designers should worry about enabling software radios to use techniques to reduce energy consumption by analog components. The essential step toward reducing energy consumption is to turn the radio off when not in use, something easier said than done. The key problem is how a radio knows to turn itself on when another radio wants to send it something. Considerable progress in recent years has taken two complementary paths: First, mechanisms make it possible for radios in a group to know when other radios are awake.[3,5,12,16] And second, using a low-power wakeup, or "doorbell," radio to signal another radio to turn on its higher-power radio to receive a transmission lowers the cost of being awake. Some working radios today use over 99% less energy than a typical WiFi chipset while transmitting the same traffic, and radio designers are beginning to migrate the

f http://www.picochip.com

g Consider also this contrarian observation: Increasingly, analog components in radios are being replaced by digital components with lower-energy profiles, as in Black Sand Technologies' CMOS power amplifier, while other components, especially filters, use MEMS technology. We could therefore expect the processor to be the top energy-consuming component in the future.

results into software radios.[13]

Another green issue concerns disposable radios. With lower energy consumption, we envision radios with such long operating lives it may be simpler to replace than to recharge them. But if such radios are to be ubiquitous, how can we keep them from adding to our trash? One research effort in the Center for Wireless Sensor Networks at Uppsala University seeks to make radios biodegradable.[h]

Processors vs. chipsets radios. While this article takes the view that there is a substantial difference between a radio built from programmable components and one built on a highly configurable chipset, I would be remiss if I did not mention an alternative perspective.

There is an argument that fully programmable and chipset radios are not very different. The core observation is that RF signaling and propagation is a mature field. Radio engineers know a lot about RF physics. Many of today's protocols, especially for the physical layer (frequencies and coding), represent sweet spots for high-quality data channels.

From this perspective, it is perfectly reasonable to assume there is a limited set of reasonable choices for radio communications and entirely plausible that a radio engineer could implement all the reasonable permutations in a chipset. If this assumption holds up, then the difference between chipset radios and radios built from programmable components is practically nil.

Unfortunately, this is a paper argument. No one has attempted to build a sufficiently rich chipset radio, so we do not know if it is possible.

Realizing the World of Software Radios

Recall that in the PDA scenario described earlier, the PDA downloads the "right" protocols whenever it needs them, but how exactly would that work? How does the PDA's radio ensure it does not load rogue software that would interfere with, say, a public-safety radio channel? This is an essential research problem relating to both how to exploit the spectrum and how to address regulatory concerns.

Describing radio behavior. Given that the central feature of software radios is their ability to change behavior, one might imagine a lot of practical and theoretical work has been done on how to tell a radio how to behave and how a radio can describe its own behavior. However, rather stunningly, little work has targeted this problem.

To appreciate the inadequate state of research, consider how a PDA might learn what software to download; all possible choices are poorly understood.

One scenario is that there's a standard radio channel (or set of channels) continuously transmitting the right software for a particular region. In a poorly designed world, this channel repeatedly broadcasts the software for each product. So, for consumers who own a Nokia device, their PDA would listen until the Nokia software is transmitted. This solution has one benefit: the local spectrum regulator is able to track what software is being broadcast and ensure only "safe" protocols are distributed. Otherwise, the system wastes valuable spectrum, repeatedly transmitting software for every possible radio, and radios may have to wait a long time before their software is transmitted and available.

A much better version of this scenario, for software engineers and consumers alike, would be if all PDAs used the same software. Imagine something like Java for radio protocols. The software channel described earlier transmits only a handful of protocol implementations running on all devices. The dual challenges are that creating programming languages to program physics is difficult[1] and finding a programming abstraction that works equally well for DSPs, FPGAs, configurable chipsets, and any given mix of them is, perhaps, even more difficult.

A variation is the local channel broadcasts specifications of radio protocols. Now imagine a common language describing the physical layer (such as frequencies used, coding, and power rules) and the media-access layer (such as time division vs. code division and packet formats). A radio receiving this specification would convert the specification into a configuration (chipset radios) or compile it into software that drives the radio. Some work has been done in this area,[15,17] with one nice concept being that beyond specifying what the radio does, the specification also describes how the radio might scan the spectrum to learn what frequencies are available.

A different approach is that a standards body registers names for each protocol in use, an approach that works best with a small set of protocols and assumes that each radio has the software (or configuration information) for all protocols pre-loaded. It is roughly what the Joint Tactical Radio System (JTRS) uses, but the JTRS team has sought to reduce the list of approved protocols, suggesting the approach is limited.[i]

Approved use of the spectrum. Software radios have the potential to dramatically change how the radio spectrum is used, unsettling some regulators and spectrum licensors. Regulators worry that a software device will be programmed (intentionally or accidentally) to interfere with existing approved uses of particular frequencies. An oft-cited example is a software radio that decides to use a frequency reserved for emergency services (often idle), interfering with authorized transmissions in an emergency.

Likewise, spectrum licensors with exclusive rights to use particular frequencies, often finding it difficult to fill those frequencies with traffic, worry that software radios will be used to "squat" on their frequencies without paying the incumbent.

On paper, at least, these fears are baseless. There appears to be multiple ways to protect the spectrum from improper or unauthorized use. Unfortunately, but for some small and unpublished experiments, no one has actually confirmed that the paper solutions work in the real world.

All proposed solutions assume some executive component or terminal reconfiguration manager within each radio ensuring the radio obeys the rules. The reconfiguration manager can take many forms. Considering a few representative examples, it is useful to assume that national spectrum authorities and spectrum licensors can digitally sign informa-

h http://www.wisenet.uu.se/

i http://www.public.navy.mil/jpeojtrs/Pages/Welcome.aspx lists nine approved waveforms, reduced from an originally planned 32 waveforms.

tion using public keys and a radio's reconfiguration manager can check these signatures.

The simplest solution is to have each radio download a table of acceptable configurations, digitally signed by the spectrum authorities. This approach works particularly well in a chipset with a limited number of configurations. It could also work in a programmable radio; one can imagine a configuration that specifies what versions of various software modules are required and the frequencies that can be used by which software. Open questions include: What specification language should be used for the table?; How big should the table be?; and Will the table have to be broken into chunks by spectrum range, with the radios selectively downloading what they need?

Another solution is to assign every geographic area a wireless network manager that informs the radios within its area of the local operating rules. This approach is being taken by the IEEE 1900.4 and 1900.5 standards efforts, seeking to define a management architecture (1900.4) and policy language through which the network manager tells the terminal reconfiguration manager the operational rules (1900.5). However, unexplained in this approach is what a radio is able to do in the absence of a manager.

Yet another approach is for the various worldwide national spectrum authorities and licensors to digitally sign the software or radio specifications described earlier. The signers would also add attributes designating the frequencies on which the software or specification can be used. In this case, the reconfiguration manager must ensure the software is signed and running on approved frequencies. There's some worry about how easy it would be for a spectrum authority to verify software, but research[2] shows that automated verification of device drivers can be effective, suggesting verification could be an automated task.

A fourth approach is to make the trusted module in the radio into a cognitive reasoner. The radio periodically scans the spectrum for available frequencies. The reasoner then examines a signed set of spectrum rules (composed of spectrum rules from the national spectrum authority and from licensors), creating a protocol able to best use the available spectrum. A slight variation is there's both a reasoner (not trusted) and a validator (trusted), with the reasoner creating a protocol and the validator confirming the protocol is legal. This approach is ambitious, but two projects[11,14] have demonstrated validators, suggesting they might be feasible commercially.

The key problem is how a radio knows to turn itself on when another radio wants to send it something.

These approaches are not exclusive. A central wireless manager could designate some portions of the spectrum available for use by radios capable of cognitive reasoning. A cognitive radio could restrict itself to deciding which of the several signed protocol specifications is appropriate in the current environment. But little research is available to inform us about what combinations of these approaches would make sense.

How much spectrum? One motivation for developing software radios is their presumed ability to use underutilized frequencies (such as the example discussed earlier of moving to an unused frequency to get enough bandwidth). That presumption raises the question of how much of the spectrum is, in fact, unused at any given time in any given place. Unfortunately, we can only partly answer.

A limited 2005 study for the National Science Foundation surveyed the spectrum from 30MHz to 3GHz at six locations (five urban and one rural), finding all the spectrum almost completely unused. In the rural test, occupancy was only 1%, and in the most used location (in New York City), occupancy was only 13%.[9]

As insightful as it was, the study represents only a starting point. It measured energy in the spectrum, but energy in the spectrum is an imperfect measure. Just because a public-safety frequency is not in use doesn't mean it's free for others to use. Similarly, in the newly freed white-space frequencies (formerly used for analog television) in the U.S., new users must take care not to interfere with wireless microphones and other historical users of the spectrum. Furthermore, there are many ways to share spectrum, including underlaying, where a radio transmits using modest power in the

same band as a strong signal, as in a TV broadcast, so regular users do not see interference, but collaborating radios distinguish between the different transmissions. Needed are richer measurement studies that test more locations and cover enough detail so software and radio engineers are able to estimate what sharing mechanisms will work well and how much bandwidth a particular radio can access and use; a first example of such a study appeared in 2010.[7] More are needed.

Observe an important, though often-ignored, point in the last paragraph. The nature of wireless research is changing. The idea of simply testing how a standardized wireless protocol works under certain conditions (such as urban vs. rural) is rarely useful research. In a world in which radios can change their protocols in seconds, we must discover which protocol should run in those conditions and how a radio might learn about its environment so it can instantiate the protocol.

But even before these more sophisticated measurements are done, it is safe to say the current perceived shortage of wireless bandwidth is, in large part, a function of our inability to exploit a hugely underused spectrum.

Conclusion

Wireless is a vital piece of our datacommunications present and will be an even more vital piece of the future, with software in commercial radios able to maximize that future.

Yet, looking over this article, I hope it is clear that we (computer science, radio engineering, manufacturing, and consumer and public-policy advocates) suffer from myopia. For most key topics, including radio behavior, approved use of the spectrum, and even how poorly the spectrum is used today, we have sometimes barely enough information to be excited about it and not enough to make an informed decision about how best to realize it. The point worth repeating is we are ill-prepared to make decisions about future use of wireless data communications.

We must move briskly, however, or risk missing the untapped promise of the wireless spectrum. Research is the way to fill the information gap, but in a world where low-cost software radios are beginning to appear, there's little time to do the research. Needed instead is an evolving research plan.

It helps to start with what is going right. Radio engineers are well on the way to having wonderful radio platforms on which to run software, with USRP, WNAN, and Sora leading the way.

Regulators are beginning to provide spectrum for experimentation with these radios. Ireland's spectrum regulator ComReg leads here, having both licensed spectrum for research and publicly declared its willingness in 2006 to make more spectrum available.[8]

The most pressing need is research into languages to describe radio behavior. Most visibly, software engineers need ways to describe a protocol to heterogeneous radios in the field such that they can immediately run the protocol. It should be possible to write a new protocol and deploy it to radios from multiple manufacturers in minutes (or at most hours, if regulatory approval is needed).

Research is also needed in ways to allow software radios to use the spectrum appropriately. Researchers have several paper solutions but only one implemented approach (incorporated into products from Shared Spectrum http://www.sharedspectrum.com/), but there is only limited experience. Such an important problem needs more attention.

Government research agencies need to fund a few efforts to build a chipset radio. As outlined here, several challenging problems look like they might be easier to solve on a chipset radio–if we can only just build one.

There is also a need for researchers to perform richer measurements of the available spectrum to better understand how much of it is used worldwide. Furthermore, we need to understand how much available bandwidth the underused spectrum represents, meaning experiments that do not simply measure energy but that also estimate what protocols would work best in a given location and the data rates they could provide.

If done in the next five years, this research would provide the information we need to make informed choices about how to unlock the wireless spectrum for data communications. We must not delay. C

References

1. Ashley-Rollman, M.P., Lee, P., Goldstein, S.C., Pillai, P., and Campbell, J.D. A Language for large ensembles of independently executing nodes. In *Proceedings of the International Conference on Logic Programming* (Pasadena, CA). Springer Verlag, Berlin, 2009, 265–280.
2. Ball, T., Bounimova, E., Cook, B., Levin, V., Lichtenberg, J., McGarvey, C., Ondrusek, B., Rajamani, S.K., and Ustuner, A. Thorough static analysis of device drivers. In *Proceedings of the First ACM Sigops/Eurosys European Conference on Computer Systems* (Leuven, Belgium, Apr. 18–21). ACM Press, New York, 2006, 73–85.
3. Dai, L. and Basu, P. Energy and delivery capacity of wireless sensor networks with random duty-cycles. In *Proceedings of the IEEE International Conference on Communications* (Istanbul, June). IEEE Press, 2006, 3503–3510.
4. Dutta, P., Kuo, Y.-S., Ledeczi, A., Schmid, T., and Volgyesi, P. Putting the software radio on a low-calorie diet. In *Proceedings of ACM HOTNETS 2010* (Monterey, CA). ACM Press, New York, 2010, 20:1–20:6.
5. IEEE Std 802.11e-2005. *IEEE Standard for Information Technology. Telecommunications and Information Exchange Between Systems. Local and Metropolitan Area Networks. Specific Requirements. Part 11: Wireless LAN Medium Access Control and Physical Layer Specifications. Amendment 8: Medium Access Control Quality of Service Enhancements.* IEEE, Nov. 11, 2005.
6. Kaul, A. Software-defined radio: The transition from defense to commercial markets. In *Proceedings of the Software Defined Radio Forum Technical Conference* (Denver, Nov. 5–9, 2007); http://data.memberclicks.com/site/sdf/sdr07-13.0-001_InvitedPaper_Kaul.pdf
7. Kone, V., Yang, L., Yang, A., Zhao, B.Y., and Zheng, H. On the feasibility of effective opportunistic spectrum access. In *Proceedings of the ACM Internet Measurement Conference* (Melbourne, Australia, Oct. 20–22). ACM Press, New York, 2010, 151–164.
8. Lillington, K. Overcrowded airwaves mean it's time to hop ahead. *The Guardian*, (Mar. 2, 2006).
9. McHenry, M.A. *NSF Spectrum Occupancy Measurements: Project Summary.* Shared Spectrum Co., Arlington, VA, Aug. 15, 2005.
10. Mitola, J. Software radios: Survey, critical evaluation, and future directions. In *Proceedings of the National Telesystems Conference* (May). IEEE Press, 1992.
11. Perich, F. Policy-based network management for next-generation spectrum access control. In *Proceedings of the Second IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks* (Dublin, Apr. 17–20). IEEE Press, 2007, 496ff.
12. Redi, J. *Energy-Conserving Protocols for Wireless Data Networks*. Ph.D. Thesis, Boston University, 1998.
13. Redi, J., Kolek, S., Manning, K., Partridge, C., Rosales-Hain, R., Ramanathan, R., and Castineyra, I. JAVeLEN: An ultra-low energy ad hoc wireless network. *Ad Hoc Networks 6*, 1 (Jan. 2008), 108–126.
14. Santivanez, C., Ramanathan, R., Partridge, C., Krishnan, R., Condell, M., and Polit, S. Opportunistic spectrum access: Challenges, architecture, protocols. In *Proceedings of the Second Annual International Wireless Internet Conference* (Boston, Aug. 2–5). ACM Press, New York, 2006.
15. Sutton, P.D., Lotze, J., Lahlou, H., Fahmy, S.A., Nolan, K.E., Ozgul, B., Rondeau, T.W., Noguera, J., and Doyle, L.E. Iris: An architecture for cognitive radio networking testbeds. *IEEE Communications Magazine 48*, 9 (Sept. 2010), 114–122.
16. Ye, W., Heidemann, J., and Estrin, D. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking 12*, 3 (June 2004), 493–506.
17. Zhong, S., Dolwin, C., Strohmenger, K., and Steinke, B. Performance evaluation of the functional description language in an SDR environment. In *Proceedings of the Software Defined Radio Forum Technical Conference* (Denver, Nov. 5–9, 2007).

Craig Partridge (craig@bbn.com) is Chief Scientist for Networking Research at Raytheon BBN Technologies, an ACM Fellow, and former chair of ACM SIGCOMM.

DOI:10.1145/1995376.1995394

Checking the satisfiability of logical formulas, SMT solvers scale orders of magnitude beyond custom ad hoc solvers.

BY LEONARDO DE MOURA AND NIKOLAJ BJØRNER

Satisfiability Modulo Theories: Introduction and Applications

Constraint-satisfaction problems arise in diverse application areas, including software and hardware verification, type inference, static program analysis, test-case generation, scheduling, planning, and graph problems, and share a common trait—a core component using logical formulas for describing states and transformations between them. The most well-known constraint satisfaction problem is propositional satisfiability, or SAT, aiming to decide whether a formula over Boolean variables, formed using logical connectives, can be made true by choosing true/false values for its variables. Some problems are more naturally described with richer languages (such as arithmetic). A supporting theory (of arithmetic) is then required to capture the meaning of the formulas. Solvers for such formulations are commonly called "satisfiability modulo theories," or SMT, solvers.

In the past decade, SMT solvers have attracted increased attention due to technological advances and industrial applications. Yet SMT solvers draw on some of the most fundamental areas of computer science, as well as a century of symbolic logic. They combine the problem of Boolean satisfiability

» key insights

- **Many tools for program analysis, testing, and verification are based on mathematical logic as the calculus of computation.**
- **SMT solvers are the core engine of many of these tools.**
- **Modern SMT solvers integrate specialized solvers with propositional satisfiability search techniques.**

Figure 1. Encoding job-shop scheduling.

$d_{i,j}$	Machine 1	Machine 2
Job 1	2	1
Job 2	3	1
Job 3	2	3

max = 8

Solution

$t_{1,1} = 5, \quad t_{1,2} = 7,$
$t_{2,1} = 2, \quad t_{2,2} = 6,$
$t_{3,1} = 0, \quad t_{3,2} = 3$

Encoding

$$(t_{1,1} \geq 0) \wedge (t_{1,2} \geq t_{1,1} + 2) \wedge (t_{1,2} + 1 \leq 8) \wedge$$
$$(t_{2,1} \geq 0) \wedge (t_{2,2} \geq t_{2,1} + 3) \wedge (t_{2,2} + 1 \leq 8) \wedge$$
$$(t_{3,1} \geq 0) \wedge (t_{3,2} \geq t_{3,1} + 2) \wedge (t_{3,2} + 3 \leq 8) \wedge$$
$$((t_{1,1} \geq t_{2,1} + 3) \vee (t_{2,1} \geq t_{1,1} + 2)) \wedge$$
$$((t_{1,1} \geq t_{3,1} + 2) \vee (t_{3,1} \geq t_{1,1} + 2)) \wedge$$
$$((t_{2,1} \geq t_{3,1} + 2) \vee (t_{3,1} \geq t_{2,1} + 3)) \wedge$$
$$((t_{1,2} \geq t_{2,2} + 1) \vee (t_{2,2} \geq t_{1,2} + 1)) \wedge$$
$$((t_{1,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{1,2} + 1)) \wedge$$
$$((t_{2,2} \geq t_{3,2} + 3) \vee (t_{3,2} \geq t_{2,2} + 1))$$

with domains (such as those studied in convex optimization and term-manipulating symbolic systems). They involve the decision problem, completeness and incompleteness of logical theories, and complexity theory. Here, we explore the field of SMT and some of its applications.

Increased attention has led to enormous progress in constraint-satisfaction problems that can be solved due to innovations in core algorithms, data structures, heuristics, and the careful use of modern microprocessors. Modern SAT[27] procedures can check formulas with hundreds of thousands of variables. Similar progress has been observed for SMT solvers for more commonly occurring theories, including such state-of-the art SMT solvers as Barcelogic,[8] CVC,[3,7] MathSAT,[10] Yices,[18] and Z3.[14]

The annual competitions for SAT (http://www.satcompetition.org) and SMT (http://www.smtcomp.org) are a key driving force.[4] An important ingredient is a common interchange format for benchmarks, called SMT-LIB,[33] and the classification of benchmarks into various categories, depending which theories are required. Conversely, a growing number of applications can generate benchmarks in the SMT-LIB format to further improve SMT solvers.

There is a relatively long tradition dating to the late-1970s of using SMT solvers in specialized contexts. One prolific case is theorem-proving systems (such as ACL2[26] and PVS[32]) that use decision procedures to discharge lemmas encountered during interactive proofs. SMT solvers have also been used for the past 15 years in the context of program verification and extended static checking[21] where verification focuses on assertion checking.

Progress in the past four years in SMT solvers has enabled their use in diverse applications, including interactive theorem provers and extended static checkers, as well as in scheduling, planning, test-case generation, model-based testing and program development, static program analysis, program synthesis, and run-time analysis.

We begin by introducing an application we use as a running example.

Scheduling. Consider the classical job-shop-scheduling decision problem, involving n jobs, each composed of m tasks of varying duration that must be performed consecutively on m machines. The start of a new task can be delayed as long as needed in order for a machine to become available, but tasks cannot be interrupted once they are started. The problem involves essentially two types of constraints:

Precedence. Between two tasks in the same job; and

Resource. Specifying that no two different tasks requiring the same machine are able to execute at the same time.

Given a total maximum time max and the duration of each task, the problem consists of deciding whether there is a schedule such that the end-time of every task is less than or equal to max time units. We use $d_{i,j}$ to denote the duration of the j-th task of job i. A schedule is specified by the start-time ($t_{i,j}$) for the j-th task of every job i. The job-shop-scheduling problem can be encoded in SMT using the theory of linear arithmetic. A precedence constraint between two consecutive tasks $t_{i,j}$ and $t_{i,j+1}$ is encoded using the inequality $t_{i,j+1} \geq t_{i,j} + d_{i,j}$; this inequality states that the start-time of task $j + 1$ must be greater than or equal to the start time of task j plus its duration. A resource constraint between two tasks from different jobs i and i' requiring the same machine j is encoded using the formula $(t_{i,j} \geq t_{i',j} + d_{i',j}) \vee (t_{i',j} \geq t_{i,j} + d_{i,j})$, stating the two tasks do not overlap. The start time of the first task of every job i must be greater than or equal to zero, so the result is $t_{i,1} \geq 0$. Finally, the end time of the last task must be less than or equal to max, hence $t_{i,m} + d_{i,m} \leq max$. Figure 1 is an instance of the job-scheduling problem, its encoding as a logical formula, and a solution. The logical formula combines logical connectives (conjunctions, disjunction, and negation) with atomic formulas in the form of linear arithmetic inequalities. We call it an SMT formula. The solution in Figure 1 is a satisfying assignment, a mapping from variables $t_{i,j}$ to values that make the formula *true*.

SMT-Solving Techniques

Modern SMT solvers use procedures for deciding the satisfiability of conjunctions of literals, where a literal is an atomic formula or the negation of an atomic formula. Throughout this article, we call these procedures "theory solvers." The scheduling application demonstrates that this kind of procedure alone is not sufficient in practice, because the encoding contains disjunctive sub-formulas, as in

$$(t_{1,1} \geq t_{2,1} + 3) \vee (t_{2,1} \geq t_{1,1} + 2)$$

SMT solvers handle sub-formulas like this by performing case analysis, which is in the core of most automated deduction tools. Most SMT solvers rely on efficient satisfiability procedures for propositional logic (SAT solvers) for performing case analysis efficiently. A standard technique for integrating SAT solvers and theory solvers[1,5,15,20,30] is described next.

SAT: A propositional core. Propositional logic is a special case of predicate logic in which formulas are built from Boolean variables, called atoms, and composed using logical connectives (such as conjunction, disjunction, and negation). The satisfiability problem for propositional logic is famously known as an NP-complete problem[12] and therefore in principle computationally

intractable. Yet recent advances in efficient propositional logic algorithms have moved the boundaries for what is intractable when it comes to practical applications.[27]

Most successful SAT solvers are based on an approach called "systematic search." The search space is a tree with each vertex representing a Boolean variable and the out edges representing the two choices (*true* and *false*) for this variable. For a formula containing n Boolean variables, the tree has 2^n leaves. Each path from the root to a leaf corresponds to a truth assignment. A model is a truth assignment that makes the formula true. We also say the model satisfies the formula, and the formula is satisfiable.

Most search-based SAT solvers are based on the DPLL/Davis-Putnam-Logemann-Loveland algorithm.[13] The DPLL algorithm tries to build a model using three main operations: `decide`, `propagate`, and `backtrack`. The algorithm benefits from a restricted representation of formulas in conjunctive normal form, or CNF. CNF formulas are restricted to be conjunctions of clauses, with each clause, in turn, a disjunction of literals. Recall that a literal is an atom or the negation of an atom; for example, the formula $\neg p \wedge (p \vee q)$ is in CNF. The operation `decide` heuristically chooses an unassigned atom, assigning it to *true* or *false*, and is also called branching or case-splitting. The operation `propagate` deduces the consequences of a partial truth assignment using deduction rules. The most widely used deduction rule is the unit-clause rule, stating that if a clause has all but one literal assigned to *false* and the remaining literal l is unassigned, then the only way for the clause to evaluate to true is to assign l to *true*.

Let C be the clause $p \vee \neg q \vee \neg r$, and M the partial truth assignment $\{p \rightarrow false, r \rightarrow true\}$, then the only way for C to evaluate to *true* is by assigning q to *false*. Given a partial truth assignment M and a clause C in the CNF formula, such that all literals of C are assigned to *false* in M, then there is no way to extend M to a complete model M' that satisfies the given formula. We say this is a conflict, and C is a conflicting clause. A conflict indicates some of the earlier decisions cannot lead to a truth assignment that satisfies the given formula, and the DPLL procedure must `backtrack` and try a different branch value. If a conflict is detected and there are no decisions to backtrack, then the formula is unsatisfiable; that is, it does not have a model. Many significant improvements to this basic procedure have been proposed over the years, with the main ones being lemma learning, non-chronological backtracking, and efficient indexing techniques for applying the unit-clause rule and preprocessing techniques.[27]

A solver for difference arithmetic. The job-shop-scheduling decision problem can be solved by combining a SAT solver with a theory solver for difference arithmetic. Difference arithmetic is a fragment of linear arithmetic, where predicates are restricted to be of the form $t - s \leq c$ and where t and s are variables and c a numeric constant (such as 1 and 3). Every atom in Figure 1 can be put into this form; for example, the atom $t_{3,1} \geq t_{2,1}+3$ is equivalent to the atom $t_{2,1}-t_{3,1} \leq -3$. For atoms of the form $s \leq c$ and $s \geq c$, a special fresh variable z is used. We say z is the zero variable, and the atoms are represented in difference arithmetic as $s - z \leq c$ and $z - s \leq -c$, respectively; for example, the atom $t_{3,2} + 3 \leq 8$ is represented in difference arithmetic as $t_{3,2} - z \leq 5$. A set of difference arithmetic atoms can be checked efficiently for satisfiability by searching for negative cycles in weighted directed graphs. In the graph representation, each variable corresponds to a node, and an inequality of the form $t - s \leq c$ corresponds to an edge from s to t with weight c. Figure 2 is a subset of atoms (in difference arithmetic form) from the example in Figure 1, along with the corresponding graph. The negative cycle, with weight –2, is shown by dashed lines. The cycle corresponds to the following schedule that cannot be completed in eight time units:

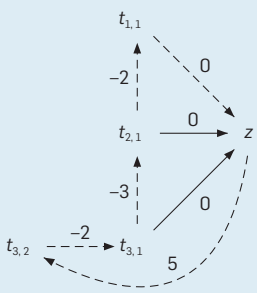
task 1/job 1 → task 1/job 2 →
task 1/job 3 → task 2/job 3

Recall that the scheduling problem in Figure 1 is satisfiable but requires assigning a different combination of atoms to *true*.

Interfacing solvers with SAT. We've outlined a theory solver for difference arithmetic and now describe how a SAT procedure interacts with this theory solver. The key idea is to create an abstraction that maps the atoms in an SMT formula into fresh Boolean variables $p_1, \ldots, p_n$; for example, the formula $\neg(a \geq 3) \wedge (a \geq 3 \vee a \geq 5)$ is translated into $\neg p_1 \wedge (p_1 \vee p_2)$, where the atoms $a \geq 3$ and $a \geq 5$ are replaced by the Boolean variables p_1 and p_2, respectively. The new abstract formula can then be processed by a regular SAT procedure. If the SAT procedure finds the abstract formula to be unsatisfiable, then so, too, is the SMT formula. On the other hand, if the abstract formula is found to be satisfiable, the theory solver is used to check the model produced by the SAT procedure. The idea is that any model produced by the SAT procedure induces a set of literals; for example, $\{p_1 \rightarrow$ false, $p_2 \rightarrow$ true$\}$ is a model for the formula $\neg p_1 \wedge (p_1 \vee p_2)$, inducing the set of literals $\{\neg(a \geq 3), a \geq 5\}$ that is unsatisfiable in the theory of arithmetic. Therefore, the formula (clause) $a \geq 3 \vee \neg(a \geq 5)$ is valid in the theory of arithmetic. The abstraction of this formula is the clause $p_1 \vee \neg p_2$. We say it is a "theory lemma," and since it is based on a valid formula from the theory of arithmetic, we can then add it to our original formula, obtaining the new formula:

Figure 2. Example of difference arithmetic.

$$
\begin{array}{lcll}
z & - & t_{1,1} & \leq 0 \\
z & - & t_{2,1} & \leq 0 \\
z & - & t_{3,1} & \leq 0 \\
t_{3,2} & - & z & \leq 5 \\
t_{3,1} & - & t_{3,2} & \leq -2 \\
t_{2,1} & - & t_{3,1} & \leq -3 \\
t_{1,1} & - & t_{2,1} & \leq -2
\end{array}
$$



$$\neg p_1 \wedge (p_1 \vee p_2) \wedge (p_1 \vee \neg p_2)$$

The SAT solver is executed again, taking the new formula as input, and finds the new formula to be unsatisfiable, proving the original formula $\neg(a \geq 3) \wedge (a \geq 3 \vee a \geq 5)$ is also unsatisfiable. In practice, many theory lemmas are created until this process converges. Note, too, this process always converges because there is a finite number of atoms, and, consequently, there is a finite number of theory lemmas that can be created using them.

Given an unsatisfiable set of theory literals S, we say a justification for S is any unsatisfiable subset J of S. Any unsatisfiable set S is, of course, also a justification for itself. We say a justification J is non-redundant if there is no strict subset J' of J that is also unsatisfiable. It is desirable to have a theory solver that produces non-redundant justifications, as they may drastically reduce the search space. This observation follows from the fact that smaller sets produce smaller theory lemmas (clauses) and consequently have fewer satisfying assignments.

Returning to the example in Figure 2, the negative cycle corresponds to a non-redundant unsatisfiable set of difference atoms. The negation of these atoms corresponds to the following valid clause in difference arithmetic:

$$\neg(t_{3,1} - t_{3,2} \leq -2) \vee \neg(t_{2,1} - t_{3,1} \leq -3) \vee \neg(t_{1,1} - t_{2,1} \leq -2) \vee \neg(z - t_{1,1} \leq 0) \vee \neg(t_{3,2} - z \leq 5)$$

This integration scheme is also known as the "lazy offline" approach and includes many refinements; one is to have a tighter integration between the two procedures, where the theory solver is used to check partial truth assignments being explored by the SAT solver (online integration). In it, additional performance gains can be obtained if the theory solver is incremental (new constraints can be added at minimal cost) and backtrackable (constraints can be removed at minimal cost). Theory deduction rules can also be used to prune the search space being explored by the DPLL solver (theory propagation). In difference arithmetic, theory propagation can be implemented by computing the shortest distance between two nodes. Returning to the example in Figure 2, assume the inequality $t_{2,1} - t_{3,1} \leq -3$ is not there. Thus, the graph on the right-hand side will not contain an edge from $t_{3,1}$ to $t_{2,1}$ and, consequently, the negative cycle. The shortest distance between the nodes $t_{2,1}$ and $t_{3,1}$ is 1 by following the path

$$t_{2,1} \rightarrow t_{1,1} \rightarrow z \rightarrow t_{3,2} \rightarrow t_{3,1}$$

This fact implies that $t_{3,1} - t_{2,1} \leq 1$, and one can verify the result by adding the inequalities associated with each edge. The inequality $t_{3,1} - t_{2,1} \leq 1$ is equivalent to $t_{2,1} - t_{3,1} \geq -1$, implying $\neg(t_{2,1} - t_{3,1} \leq -3)$. Therefore, if the SAT solver has assigned the atoms $t_{1,1} - t_{2,1} \leq -2$, $z - t_{1,1} \leq 0$, $t_{3,2} - z \leq 5$ and $t_{3,1} - t_{3,2} \leq -2$ to *true*, then, by theory propagation, the atom $t_{2,1} - t_{3,1} \leq -3$ can be assigned to *false*, thus avoiding the inconsistency (negative cycle) in Figure 2.

Program 3.1. Greatest common divisor program.

```
int GCD (int x, int y)
  while (true) {
    int m = x % y;
    if (m == 0) return y;
    x = y;
    y = m;
  }
}
```

Program 3.2. Greatest common divisor path formula.

```
int GCD (int x0, int y0) {
   int m0 = x0 % y0;          (m0 = x0 % y0)     ∧
   assert (m0 != 0);          ¬(m0 = 0)          ∧
   int x1 = y0;               (x1 = y0)          ∧
   int y1 = m0;               (y1 = m0)          ∧
   int m1 = x1 % y1;          (m1 = x1 % y1)     ∧
   assert (m1 == 0);          (m1 = 0)
}
```

SMT in Software Engineering

Software developers use logical formulas to describe program states and transformations between program states, a procedure at the core of most software-engineering tools that analyze, verify, or test programs. Here, we describe a few such applications:

Dynamic symbolic execution. SMT solvers play a central role in dynamic symbolic execution. A number of tools used in industry are based on dynamic symbolic execution, including CUTE, Klee, DART, SAGE, Pex, and Yogi,[23] designed to collect explored program paths as formulas, using solvers to identify new test inputs that can steer execution into new branches. SMT solvers are a good fit for symbolic execution because the semantics of most program statements are easily modeled using theories supported by these solvers. We later introduce the various theories that are used, but here we focus on connecting constraints with a solver. To illustrate the basic idea of dynamic symbolic execution, consider the greatest common divisor in Program 3.1, taking the inputs x and y and producing the greatest common divisor of x and y.

Program 3.2 represents the static single assignment unfolding corresponding to the case where the loop is exited in the second iteration. Assertions are used to enforce that the condition of the if statement is not satisfied in the first iteration and is in the second iteration. The sequence of instructions is equivalently represented as a formula where the assignment statements have been turned into equations.

The resulting path formula is satisfiable. One satisfying assignment that can be found using an SMT solver is of the form:

$$x_0 = 2, y_0 = 4, m_0 = 2, x_1 = 4, y_1 = 2, m_1 = 0$$

It can be used as input to the original program; in this example, the call `GCD(2,4)` causes the loop to be entered twice, as expected.

Fuzz testing is a software-testing technique that provides invalid or unexpected data to a program. The program

being fuzzed is opaque, and fuzzing is performed by perturbing input vectors using random walks. "White-box fuzzing" combines fuzz testing and dynamic symbolic execution and is actively used at Microsoft. Complementing traditional fuzz testing, it has been instrumental in uncovering several subtle security-critical bugs that traditional testing methods are unable to find.

Program model checking. Dynamic symbolic execution finds input that can guide execution into bugs. This method alone does not guarantee that programs are free of all the errors being checked for. The goal of program model checking tools is to automatically check for freedom from selected categories of errors. The idea is to explore all possible executions using a finite and sufficiently small abstraction of the program state space. The tools BLAST,[25] SDV,[2] and SMV from Cadence[a] perform program model checking. Both SDV and SMV are used as part of commercial tool offerings. The program fragment in Program 3.3 is an example of finite-state abstraction, accessing requests using `GetNextRequest`. The call is protected by a lock. A question is whether it is possible to exit the loop without having a lock. The program has a very large, potentially unbounded, number of states, since the value of the program variable count can grow arbitrarily.

However, from the point of view of locking, the actual values of count and `old _ count` are not interesting. On the other hand, the relationship between these program variables contains useful information. Program 3.4 is a finite-state abstraction of the same locking program. The Boolean variable `b` encodes the relation `count == old _ count`. In it, we use the symbol $*$ to represent a Boolean expression that nondeterministically evaluates to *true* or *false*. The abstract program contains only Boolean variables, thus a finite number of states. We can now explore the finite number of branches of the abstract program to verify the lock is always held when exiting the loop.

SMT solvers are used for constructing finite-state abstractions, like the one in Program 3.4. Abstractions can be created through several approaches; in one, each statement in the program

Most SMT solvers rely on efficient satisfiability procedures for propositional logic (SAT solvers) for performing case analysis efficiently.

is individually abstracted; for example, consider the statement `count = count + 1`. The abstraction of it is essentially a relation between the current and the new values of the Boolean variable `b`. SMT solvers are used to compute the relation by proving theorems, as in

```
count == old _ count →
count+1 != old _ count
```

which is equivalent to checking unsatisfiability of the negation

```
count == old _ count ∧
count+1 == old _ count
```

The theorem says if the current value of `b` is *true*, then after executing the statement `count = count + 1`, the value of `b` will be *false*. Note that if `b` is *false*, then neither of the following conjectures is valid:

```
count != old _ count →
count+1 == old _ count
count != old _ count →
count+1  != old _ count
```

In each, an SMT solver will produce a model for the negation of the conjec-

Program 3.3. Processing requests using locks.

```
do {
    lock ();
    old_count = count;
    request = GetNextRequest();
    if (request != NULL) {
        unlock();
        ProcessRequest(request);
        count = count + 1;
    }
}
while (old_count != count);
unlock();
```

Program 3.4. Processing requests using locks, abstracted.

```
do {
    lock ();
    b = true;
    request = GetNextRequest();
    if (request != NULL) {
        unlock();
        ProcessRequest(request);
        if (b) b = false; else b = *;
    }
}
while (!b);
unlock();
```

a http://www.kenmcmil.com

ture. Therefore, the model is a counterexample of the conjecture, and when the current value of `b` is false, nothing can be said about its value after the execution of the statement. The result of these three proof attempts is then used to replace the statement `count = count + 1; by if (b) b = false; else b = *;`. A finite state model checker can now be used on the Boolean program and will establish that `b` is always *true* when control reaches this statement, verifying that calls to `lock()` are balanced with calls to `unlock()` in the original program.

Static program analysis. Static program analysis tools work like dynamic-symbolic-execution tools, checking feasibility of program paths. On the other hand, they never require executing programs and can analyze software libraries and utilities independently of how they are used. One advantage of using modern SMT solvers in static program analysis is they accurately capture the semantics of most basic operations used by mainstream programming languages. The program fragment in Program 3.5 illustrates the need for static program analysis to use bit-precise reasoning, searching for an index in a sorted array `arr` containing a key.

The `assert` statement is a precondition for the procedure, restricting the input to fall within the bounds of the array `arr`. The program performs several operations involving arithmetic, so a theory and corresponding solver that understands arithmetic is arguably a good match. However, it is important for software-analysis tools to take into account that languages (such as Java, C#, and C/C++) all use fixed-width bit-vectors as representation for values of type `int`, meaning the accurate theory for `int` is two-complements modular arithmetic. Assuming a bit-width of 32b, the maximal positive 32b integer is $2^{31}-1$, and the smallest negative 32b integer is -2^{31}. If both `low` and `high` are 2^{30}, `low + high` evaluates to 2^{31}, which is treated as the negative number -2^{31}. The presumed assertion $0 \le mid < high$ does therefore not hold. Fortunately, several modern SMT solvers support the theory of "bit-vectors," accurately capturing the semantics of modular arithmetic. The bug does not escape an analysis based on the theory of bit-vectors. Such analysis would check that the array read `arr[mid]` is within bounds during the first iteration by checking the formula

$$(low > high \vee 0 \le low < high < arr.length) \wedge (low \le high \rightarrow 0 \le (low + high)/2 < arr.length)$$

As in the case of code fragment 3.5, the formula is not valid. The values `low = high` $= 2^{30}$, `arr.length` $= 2^{30}+1$ provide a counterexample. The use of SMT solvers for bit-precise static-analysis tools is an active area of research and development in Microsoft Research. Integration with the solver Z3[14] and the static analysis tool PREfix led to the automatic discovery of several overflow-related bugs in Microsoft's codebase.

Program verification. The ideal of verified software is a long-running quest since Robert Floyd and C.A.R. Hoare introduced (in the late 1960s) program verification by assigning logical assertions to programs. Extended static checking uses the methods developed for program verification but in the more limited context of checking absence of runtime errors. The SMT solver Simplify[16] was developed in the context of the extended static-checking systems ESC/Modula 3 and ESC/Java.[21] This work was and continues to be the inspiration for several subsequent verification tools, including Why[19] and Boogie.[3] These systems are actively used as bridges from several different front ends to SMT-solver back ends; for example, Boogie is used as a back end for systems that verify code from languages (such as an extended version of C# called Spec#), as well as low-level systems code written in C. Current practice indicates that a lone software developer can drive these tools to verify properties of large codebases with several hundred thousand lines of code. A more ambitious project is the Verifying C-Compiler system,[11] targeting functional correctness properties of Microsoft's Viridian Hyper-Visor. The Hyper-Visor is a relatively small (100,000 lines) operating-system layer, yet formulating and establishing correctness properties is a challenge. The entire verification effort for this layer is estimated by Microsoft to take around 60 programmer years.

Program-verification applications often use theories not already supported by existing specialized solvers but that are supported indirectly using axiomatizations with quantifiers. As an example of such a theory, in object-oriented-type systems used for Java and C#, it is the case that objects are related using a single inheritance scheme; that is, every object inherits from at most one unique immediate parent. To illustrate the theory, let *array-of*(*x*) be the array type constructor for arrays of values of type *x*. In some programming languages, if *x* is a subtype of *y*, then *array-of*(*x*) is a subtype of *array-*

Figure 3. Axioms for *sub*.

$(\forall x: sub(x, x))$

$(\forall x,y,z: sub(x, y) \wedge sub(y, z) \rightarrow sub(x, z))$

$(\forall x,y: sub(x, y) \wedge sub(y, x) \rightarrow x = y)$

$(\forall x,y,z: sub(x, y) \wedge sub(x, z) \rightarrow sub(y, z) \vee sub(z, y))$

$(\forall x,y: sub(x, y) \rightarrow sub(array\text{-}of(x), array\text{-}of(y)))$

Program 3.5. Binary search.

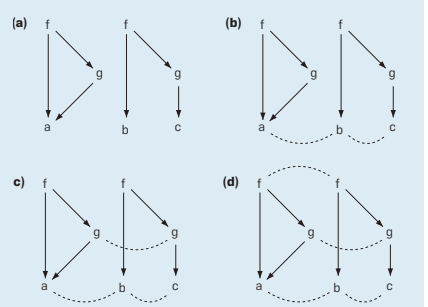
```
int binary_search(
    int[] arr, int low, int high, int key) {
        assert (low > high || 0 <= low < high);
        while (low <= high) {
            //Find middle value
            int mid = (low + high)/2;
            assert (0 <= mid < high);
            int val = arr[mid];
            //Refine range
            if (key == val) return mid;
            if (val > key) low = mid+1;
            else high = mid-1;
        }
        return -1;
}
```

of(*y*). In this case, we say arrays behave in a monotone way with respect to inheritance. Using first-order axioms, we specify in Figure 3 that the inheritance relation *sub*(*x, y*) is a partial order satisfying the single inheritance property and that the array type constructor *array-of*(*x*) is monotone with respect to inheritance.

The theory of object inheritance illustrates why SMT solvers targeted at expressive program analysis benefit from general support for quantifiers.

All the applications we have treated so far also rely on a fundamental theory we have not described: the theory of equality and free functions. The axioms used for object inheritance used the binary predicate *sub* and the function *array-of*. All we know about *array-of* is that it is monotone over sub, and, for this reason, we say the function is free. Decision procedures for free functions are particularly important because it is often possible to reduce decision problems to queries over free functions. Given a conjunction of equalities between terms using free functions, a congruence closure algorithm can be used to represent the smallest set of implied equalities. This representation can help check if a mixture of equalities and disequalities are satisfiable, checking that the terms on both sides of each disequality are in different equivalence classes. Efficient algorithms for computing congruence closure are the subject of long-running research[17] in which terms are represented as directed acyclic graphs, or DAGS. Figure 4 outlines the operation of a congruence closure algorithm on the following limited example
$a = b, b = c, f(a, g(a)) \neq f(b, g(c))$

Figure 4. Example of congruence closure.



SMT solvers are a good fit for symbolic execution because the semantics of most program statements are easily modeled using theories supported by these solvers.

In Figure 4(a), we spelled out a DAG for all terms in the example; in Figure 4(b), the equivalences $a = b$ and $b = c$ are represented by dashed lines; in Figure 4(c), nodes $g(a)$ and $g(c)$ are congruent because $a = c$ is implied by the first two equalities; and finally, in Figure 4(d), nodes $f(a, g(a))$ and $f(b, g(c))$ are also congruent, hence the example is unsatisfiable due to the required disequality $f(a, g(a)) \neq f(b, g(c))$.

Modeling. SMT solvers represent an interesting opportunity for high-level software-modeling tools. In some contexts these tools use domains from mathematics (such as algebraic datatypes, arrays, sets, and maps) and have also been the subject of long-running research in the context of SMT solvers. Here, we introduce the array domain that is frequently used in software modeling.

The theory of arrays was introduced by John McCarthy in a 1962 paper[28] as part of forming a broader agenda for a calculus of computation. It included two functions: *read* and *write*. The term *read*(*a*, *i*) produces the value of array *a* at index *i*, and the term *write*(*a*, *i*, *v*) produces an array equal to *a*, except for possibly index *i*, which maps to *v*. To make the terminology closer to how arrays are read in programs, we write *a*[*i*] instead of *read*(*a*, *i*). These properties are summarized through two equations:

$write(a, i, v)[i] = v$
$write(a, i, v)[j] = a[j]$ for $i \neq j$

They state that the result of reading *write*(*a*, *i*, *v*) at index *j* is *v* for *i* = *j*. Reading the array at any other index produces the same value as *a*[*j*]. Consider, for example, the program swap, swapping the entries *a*[*i*] and *a*[*j*].

```
void swap (int [] a, int i, int j)

{
int tmp = a[i];
a[i] = a[j];
a[j] = tmp;
}
```

The statement that *a*[*i*] contains the previous value of *a*[*j*]can be expressed as
$a[j] = write(write(a, i, a[j]), j, a[i])[i]$

Here, we summarize a few areas in the context of software modeling where SMT solvers are used. Model programs are behavioral specifications that can be described succinctly and at a high level of abstraction. These descriptions are state machines that use abstract domains. SMT solvers are used to perform bounded model-checking of such descriptions. The main idea of bounded model-checking is to explore a bounded symbolic execution of a program or model. Thus, given a bound (such as 17), the transitions of the state machines are unrolled into a logical formula describing all possible executions using 17 steps. Model-based designs use high-level languages for describing software systems. Implementations are derived by refinements. Modeling languages present an advantage, as they allow software developers to explore a design space without committing all design decisions up front. SMT solvers are the symbolic reasoning engines used in model-based designs; for example, they are used for type-checking designs and in the search for different consistent choices. Model-based testing uses high-level models of software systems, including network protocols, to derive test oracles. SMT solvers have been used in this context for exploring related models using symbolic execution. Model-based testing is used on a large scale by Microsoft developers in the context of disclosure and documentation of Microsoft network protocols.[24] The model-based tools use SMT solvers for generating combinations of test inputs, as well as for performing symbolic exploration of models.

One advantage of using modern SMT solvers in static program analysis is they accurately capture the semantics of most basic operations used by mainstream programming languages.

Combining Theory Solvers

How to combine multiple theory solvers is a fundamental problem for SMT solvers. As we discussed earlier, applications ranging from test-case generation to software verification require a combination of theories; for example, a combination of arithmetic and arrays is needed to reason about Program 3.5. Fundamental questions include: Is the union of two decidable theories still decidable? Is the union consistent? And how can we combine different theory solvers? In general, combining theory solvers is a very difficult problem. However, useful special cases have good answers. An established framework for combining theory solvers is known as the Nelson-Oppen combination method,[29] which assumes theories do not share symbols except for the equality relation. When the only shared symbol is the equality relation, we say the theories are disjoint; for example, the theory of linear arithmetic uses the constants, functions, and relations +, 0, 1, ≤, and the theory of arrays uses the disjoint set *read*, *write*. It should also be possible to merge the models from the two theory solvers into one without contradicting assumptions one theory might have about the size of models. A condition that guarantees solutions can be combined is known as "stable infiniteness"; a theory T is stably infinite if whenever a (quantifier-free) formula is satisfiable in T, then it is satisfiable in a model of T with an infinite universe (size).

In many practical cases, the disjointness and stable infiniteness conditions are easily satisfied when combining theory solvers. However, not all theory combinations satisfy these side conditions, and research over the past 10 years has sought to generalize the framework where signatures are nondisjoint or where theories are non-stably infinite.[22,34]

Convexity, complexity, and propositional search. Convexity is an important notion in the context of combining theories. A theory is convex if for all sets of ground literals S and all sets of equalities between variables E if S implies the disjunction of E, then it also implies at least one equation of E; for example, the theory of free functions is convex, but difference arithmetic over integers is not.

Convexity plays an important role in operations research, as well as in SMT, because efficient, polynomial time techniques exist for combining solvers for convex theories.[31] The key property is that the equalities can be deduced, without backtracking, instead of guessed, with backtracking. On the other hand, nonconvex theories incur a potential exponential time combination overhead. It therefore becomes an additional requirement on solvers in the Nelson-Oppen combination method that they also indicate which variables are implied equal based on a set of assertions.

The advent in the late-1990s of efficient methods for propositional search allowed viewing the theory combination problem from a different, more advantageous perspective. The delayed theory combination[9] method creates one atomic equality for every pair of variables shared between solvers. These additional atomic equalities are assigned to *true* or *false* by a SAT solver. In this approach, the SAT solver is used to guess the correct equalities between shared variables. If the theory solvers disagree with the (dis)equalities, then the conflict causes the SAT solver to backtrack. The approach is oblivious to whether or not theories are convex. Delayed theory combination potentially pollutes the search space with a large number of mostly useless new atomic equalities. The "Model-based theory combination" method[14] allows more efficient handling of convex and non-convex theories, asking the solvers to generate a model. The atomic equality predicates are created only if two shared variables are equal in a model.

Conclusion

Over the past 10 years, SMT has become the core engine behind a range of powerful technologies and an active, exciting area of research with many practical applications. We have presented some of the basic ideas but did not cover many details and heuristics; other recent topics in SMT research[6] include proof-checking, integration with first-order quantifiers, quantifier elimination methods, and extraction of so-called Craig interpolant formulas from proofs. We also did not cover several existing and emerging applications, including sophisticated runtime analysis of real-time embedded systems,[b] estimating asymptotic runtime bounds of programs, and program synthesis.

SMT-solving technologies have had a positive effect on a number of application areas, providing rich feedback in terms of experimental data. The progress in the past six years has relied heavily on experimental evaluations that uncovered new theoretical challenges, including better representations and algorithms, efficient methods for combining procedures, theories for quantifier reasoning, and various extensions to the basic search method. C

b http://www.eecs.berkeley.edu/~sseshia/research/embedded.html

References

1. Audemard, G., Bertoli, P., Cimatti, A., Kornilowicz, A., and Sebastiani, R. A SAT-based approach for solving formulas over Boolean and linear mathematical propositions. In *Proceedings of the Conference on Automated Deduction, Vol. 2392 of LNCS* (Copenhagen, July 27–30). Springer-Verlag, Berlin, 2002.
2. Ball, T. and Rajamani, S.K. The SLAM project: Debugging system software via static analysis. (Symposium on Principles of Programming Languages). *SIGPLAN Notices 37*, 1 (Jan. 16–18, 2002), 1–3.
3. Barnett, M., Leino, K.R.M., and Schulte, W. The Spec# programming system: An overview. In *Proceedings of the International Workshop on Construction and Analysis of Safe, Secure and Interoperable Smart Devices, LNCS 3362* (Marseille, Mar. 10–13). Springer-Verlag, Berlin, 2005, 49–69.
4. Barrett, C., de Moura, L., and Stump, A. Design and results of the first Satisfiability Modulo Theories Competition. *Journal of Automated Reasoning 35*, 4 (Nov. 2005), 372–390.
5. Barrett, C., Dill, D., and Stump, A. Checking satisfiability of first-order formulas by incremental translation to SAT. In *Proceedings of the International Conference on Computer Aided Verification* (Copenhagen, July, 27–31). Springer-Verlag, Berlin 2002, 236–249.
6. Barrett, C., Sebastiani, R., Seshia, S.A., and Tinelli, C. *Satisfiability Modulo Theories, Vol. 185 of Frontiers in Artificial Intelligence and Applications*, Chapter 26. IOS Press, Feb. 2009, 825–885.
7. Barrett, C. and Tinelli, C. CVC3. In *Proceedings of the 19th International Conference on Computer Aided Verification, Vol. 4590 of LNCS*, W. Damm and H. Hermanns, Eds. (Berlin, July 3–7). Springer-Verlag, Berlin, 2007, 298–302.
8. Bofill, M., Nieuwenhuis, R., Oliveras, A., Rodríguez Carbonell, E., and Rubio, A. The Barcelogic SMT Solver. In *Proceedings of the 20th International Conference on Computer Aided Verification, Vol. 5123 of LNCS*, A. Gupta and S. Malik, Eds. (Princeton, July 7–14). Springer-Verlag, Berlin, 2008, 294–298.
9. Bozzano, M., Bruttomesso, R., Cimatti, A., Junttila, T.A., Ranise, S., van Rossum, P., and Sebastiani, R. Efficient satisfiability modulo theories via delayed theory combination. In *Proceedings of the International Conference on Computer Aided Verification, Vol. 3576 of LNCS*, K. Etessami and S. K. Rajamani, Eds. (Edinburgh, July 6–12). Springer-Verlag, Berlin, 2005, 335–349.
10. Bruttomesso, R., Cimatti, A., Franzén, A., Griggio, A., and Sebastiani, R. The MathSAT 4 SMT Solver. In *Proceedings of the 18th International Conference on Computer Aided Verification, Vol. 5123 of LNCS*. Springer-Verlag, Berlin, 2008.
11. Cohen, E., Dahlweid, M., Hillebrand, M., Leinenbach, D., Moskal, M., Santen, T., Schulte, W., and Tobies, S. VCC: A practical system for verifying concurrent C. In *Proceedings of the International Conference on Theorem Proving in Higher Order Logics* (Munich, Aug. 17–20). Springer-Verlag. Berlin, 2009, 23–42.
12. Cook, S.A. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (May 3–5). ACM Press, New York, 1971, 151–158.
13. Davis, M., Logemann, G., and Loveland, D. A machine program for theorem proving. *Commun. ACM 5*, 2 (July 1962), 394–397.
14. de Moura, L. and Bjørner, N. Z3: An efficient SMT solver. In *Proceedings of the International Conference on tools and algorithms for the Construction and Analysis of Systems, Vol. 4963 of LNCS*, C.R. Ramakrishnan and J. Rehof, Eds. (Budapest, Mar. 29–Apr. 6). Springer-Verlag, Berlin, 2008, 337–340.
15. de Moura, L. and Rueß, H. Lemmas on demand for satisfiability solvers. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing* (Cincinnati, May 6–9, 2002).
16. Detlefs, D., Nelson, G., and Saxe, J.B. Simplify: A theorem prover for program checking. *Journal of the ACM 52*, 3 (May 2005), 365–473.
17. Downey, P.J., Sethi, R., and Tarjan, R.E. Variations on the common subexpression problem. *Journal of the ACM 27*, 4 (Oct. 1980), 758–771.
18. Dutertre, B. and de Moura, L. A fast linear-arithmetic solver for DPLL(T). In *Proceedings of the 16th International Conference on Computer Aided Verification, Vol. 4144 of LNCS* (Seattle, Aug. 17–20). Springer-Verlag, Berlin, 2006, 81–94.
19. Filliâtre, J.-C. *Why: A Multi-Language Multi-Prover Verification Tool. Technical Report 1366*, Université Paris Sud, 2003.
20. Flanagan, C., Joshi, R., Ou, X., and Saxe, J.B. Theorem proving using lazy proof explication. In *Proceedings of the 15th International Conference on Computer Aided Verification* (Boulder, CO, July 8–12). Springer-Verlag, Berlin, 2003, 355–367.
21. Flanagan, C., Leino, K.R.M., Lillibridge, M., Nelson, G., Saxe, J.B., and Stata, R. Extended static checking for Java. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation* (Berlin, June 17–19). ACM Press, New York, 2002, 234–245.
22. Ghilardi, S., Nicolini, E., and Zucchelli, D. A comprehensive framework for combined decision procedures. In *Proceedings of the Fifth International Workshop on Frontiers of Combining Systems, Vol. 3717 of LNCS*, B. Gramlich, Ed. (Vienna, Sept. 19–21). Springer-Verlag, Berlin, 2005, 1–30.
23. Godefroid, P., de Halleux, J., Nori, A.V., Rajamani, S.K., Schulte, W., Tillmann, N., and Levin, M.Y. Automating software testing using program analysis. *IEEE Software 25*, 5 (Sept./Oct. 2008), 30–37.
24. Grieskamp, W., Kicillof, N., MacDonald, D., Nandan, A., Stobie, K., and Wurden, F.L. Model-based quality assurance of Windows protocol documentation. In *Proceedings of the First International Conference on Software Testing, Verification, and Validation* (Lillehammer, Norway, Apr. 9–11). IEEE Computer Society Press, 2008, 502–506.
25. Henzinger, T.A., Jhala, R., Majumdar, R., and Sutre, G. Software verification with blast. In *Proceedings of the 10th International SPIN Workshop, Vol. 2648 of LNCS*, T. Ball and S. R. Rajamani, Eds. (Portland, May 9–10). Springer-Verlag, Berlin, 2003, 235–239.
26. Kaufmann, M., Manolios, P., and Moore, J.S. *Computer-Aided Reasoning: An Approach*. Kluwer Academic, June 2000.
27. Malik, S. and Zhang, L. Boolean satisfiability from theoretical hardness to practical success. *Commun. ACM 52*, 8 (Aug. 2009), 76–82.
28. McCarthy, J. Towards a mathematical science of computation. In Congress of the International Federation for Information Processing, 1962, 21–28.
29. Nelson, G. and Oppen, D.C. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems 1*, 2 (Oct. 1979), 245–257.
30. Nieuwenhuis, R., Oliveras, A., and Tinelli, C. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM 53*, 6 (Nov. 2006), 937–977.
31. Oppen, D.C. Complexity, convexity and combinations of theories. *Theoretical Computer Science 12*, 3 (1980), 291–302.
32. Owre, S., Rushby, J.M., and Shankar, N. PVS: A prototype verification system. In *Proceedings of the 11th International Conference on Automated Deduction* (Saratoga, NY, June 15–18). Springer-Verlag, Berlin, 1992, 748–752.
33. Ranise, S. and Tinelli, C. *The Satisfiability Modulo Theories Library (SMT-LIB)*, 2006; http://www.SMT-LIB.org
34. Tinelli, C. and Zarba, C.G. Combining nonstably infinite theories. *Journal of Automated Reasoning 34*, 3 (Apr. 2005), 209–238.

Leonardo de Moura (leonardo@microsoft.com) is a senior researcher in the Software Reliability Research group at Microsoft Research, Redmond, WA.

Nikolaj Bjørner (nbjorner@microsoft.com) is s senior researcher in the Foundations of Software Engineering group at Microsoft Research, Redmond, WA.

DOI:10.1145/1995376.1995396

Timed automata and their extensions allow for analysis of a wide range of performance and optimization problems.

BY PATRICIA BOUYER, ULI FAHRENBERG, KIM G. LARSEN, AND NICOLAS MARKEY

Quantitative Analysis of Real-Time Systems Using Priced Timed Automata

THE PROBLEMS OF time-dependent behavior in general, and dynamic resource allocation in particular, pervade many aspects of modern life. Prominent examples range from reliability and efficient use of communication resources in a telecommunication network to the allocation of tracks in a continental railway network, from scheduling the usage of computational resources on a chip for durations of nano-seconds to the weekly, monthly, or longer-range reactive planning in a factory or a supply chain.

These problems have been subject to substantial research for decades by different communities such as operational research, computer systems performance evaluation as well as planning and scheduling, witnessed by large communities such as ACM SIGMETRICS. In this article we argue the formalism of timed automata together with recent extensions provides an alternative framework with complementary, yet competitive, results in terms of modeling capabilities and efficiency of analysis.

Timing: Twenty years ago, R. Alur and D. Dill introduced the notion of *timed automata*. As a witness for the importance of the formalism one may consider the 2008 Computer-Aided Verification Award given to Alur and Dill for their seminal 1990 article *Automata for modeling real-time systems*,[5] which provided the theoretical foundation for the computer-aided verification of real-time systems.

Real-time systems and resource al-

» key insights

- **Timed automata and their priced and game extensions provide a mathematically beautiful formalism for modeling real-time systems, allowing constraints on quantitative aspects such as time, power, memory, and bandwidth to be easily expressed.**
- **The timed-automata based formalisms come equipped with powerful algorithmic techniques allowing for a wide range of analyses relevant for real-time systems to be carried out automatically and efficiently. In particular, the techniques are now available in a number of mature tools and have been applied to the modeling, analysis, and optimization of numerous applications.**
- **Embedded software engineers should be following the potential capabilities of priced timed automata and their algorithmic support, as it paves the way for the effective handling of quantitative constraints in model-driven development of real-time systems.**

ILLUSTRATION BY STUDIO TONNE

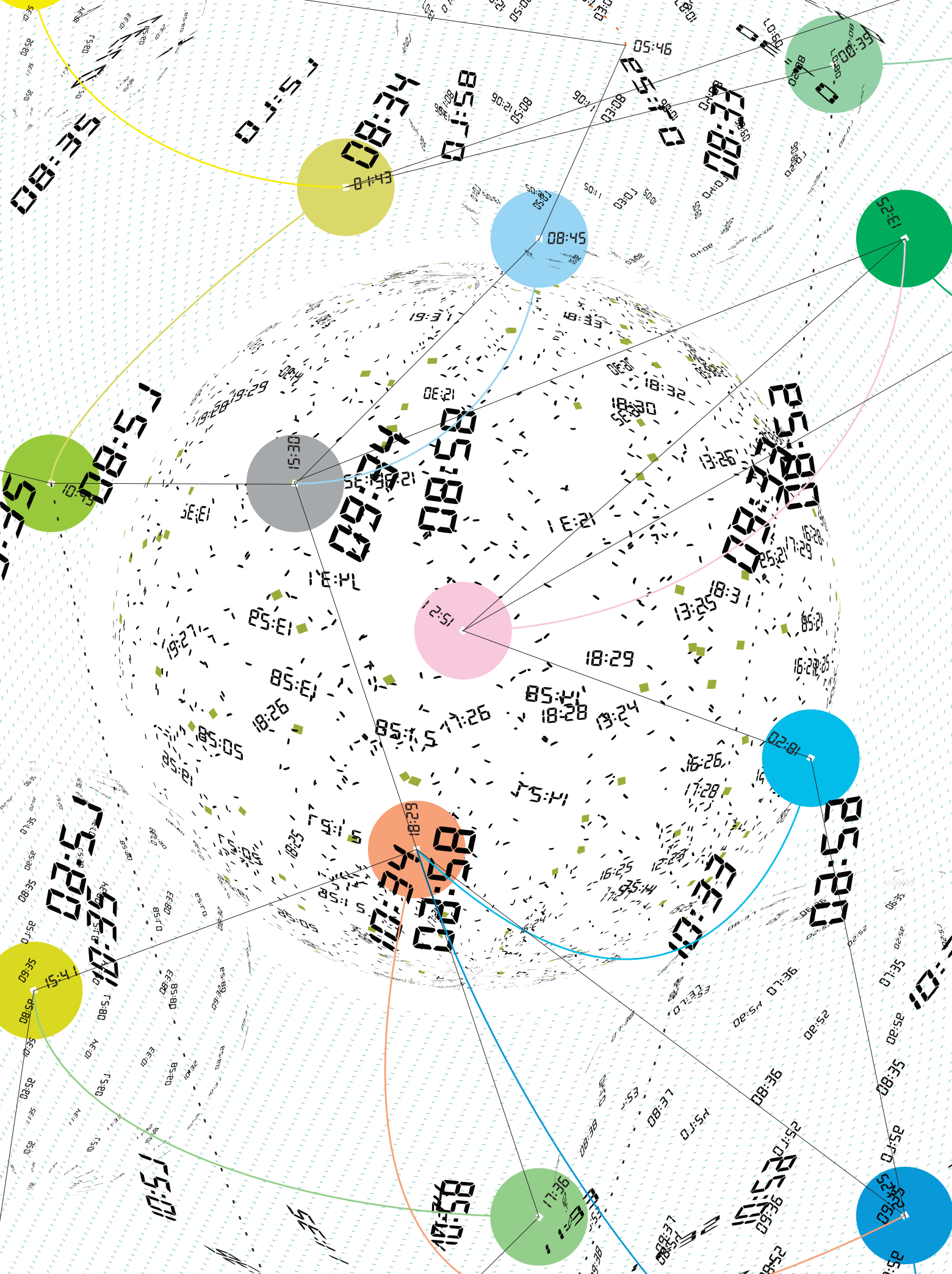
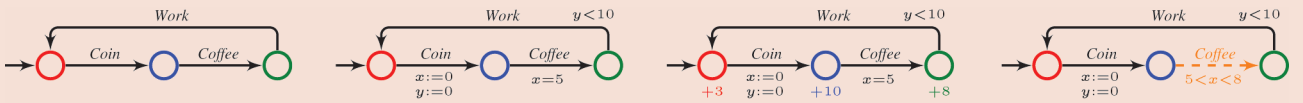


Figure 1. Several refinements of a model (a) of the working mathematician according to Erdős: after insertion of a coin into the coffee dispenser, coffee can be collected, and the scientist can go back to work. In the timed-automaton model (b), precisely five time units pass between coin insertion and coffee collection, and the time which passes between coin insertion and going back to work is less than 10 time units. In the priced timed automaton (c), cost rates (modeling, for example, energy consumption) are associated with the three states. In the timed game (d), uncertainty as to precisely when coffee is delivered is modeled as an uncontrolled edge.



(a) A finite automaton (b) A timed automaton (c) A priced timed automaton (d) A timed game

location problems have manifested themselves under different names in application domains such as manufacturing, transport, communication networks, embedded systems, and digital circuits, and have been treated using theories and methods in several disciplines. Most of these applications involve distributed, reactive systems of considerable complexity, and with a number of real-time constraints in the sense that correctness not only depends on the logical ordering of events of the systems, but also on the relative timing between these.

State-based models have been the basis of a wide range of successful computer-supported verification methodologies allowing the efficient prediction of functional properties, for example, absence of deadlock or memory overflow. However, many of the models used in this methodology are purely discrete and their treatment of time is purely qualitative, that is, behaviors are just sequences of events appearing one after the other but without any quantitative timing information about the duration of actions and the time between events. Timed automata allow such timing constraints to be expressed, while being amenable to computer-aided analysis methods such as *simulation*, *verification*, *optimization*, and *controller synthesis*.

Performance: In all of the above applications, an explicit constraint on timing is only one of a number of quantitative aspects of importance. Within embedded systems additional key quantities include *energy* and *memory* consumption, in communication networks required *bandwidth* is a key quantity, and within the factory and supply chain applications need for *storage* and overall *cost* for a given production are crucial quantities. The extended notion of *priced* or *weighted* timed automata has been put forward as a formalism allowing for such additional and time-dependent quantities to be modeled, without hampering efficient analysis and even permitting *optimization*.

Uncertainty: Classical models for scheduling in manufacturing, such as job-shop problems, are somewhat detached from industrial practice and reality. They assume that the duration of every step as well as the arrival times are fixed and known with certainty; in practice however, it is rarely the case that a schedule is executed as planned.

For solving problems related to *expected* time and performance properties, *stochastic process models* have been very successful. When aiming at *guaranteed* time and performance properties under uncertainty, so-called *timed games* may be used instead. They provide efficient offline algorithms for synthesizing reactive schedulers with performance guarantees. Such algorithms can plan for the *best* or *worst* case, but the scheduling strategies they produce are adaptive and can take advantage, for example, of the fact that a task has terminated before it was expected to.

In this article we present the formalism of timed automata and its priced and game extension as a unifying mathematical framework for the modeling, analysis, optimization, and synthesis of real-time related phenomena. Figure 1 shows some simple examples of these formalisms; later we provide more elaborate and realistic examples and case studies.

Timed Automata

A model for time. Timed automata[5] are a powerful model for representing and reasoning about systems where the notion of time is essential. They are an extension of classical finite-state automata with real-valued variables called *clocks*. These clocks all increase at the same rate, and their values can be used to restrict availability of transitions and how long one can stay in a location (or state). Also, clocks can be reset to zero when a transition is taken. To this end, each transition has associated with it a *guard* (which must be satisfied for the transition to be enabled) and a set of clocks to be reset, and each location carries an *invariant* that must be continuously satisfied when the system is in the location. Below we show an example of a timed automaton with two clocks x and y, and label set $\{a, b, c, d, e\}$. Note that no time can elapse in location ℓ_1 due to the invariant $(y = 0)$; locations with this property are called *urgent*.

Guards and invariants are given as comparisons $x \leq \alpha$ or $x < \alpha$, or the reverse relations, of a clock value with an integer constant, or as conjunctions of these. Sometimes also so-called *diagonal* constraints $x - y \leq \alpha$ (or $<$ or other) are allowed, but other extensions quickly lead to undecidability issues, see below.

A configuration of the system is made of a location and a clock valuation (in our case, values for both clocks x and y). A possible execution in our example is:

$$\begin{bmatrix}\ell_0\\0\\0\end{bmatrix} \xrightarrow[1.3]{\text{delay}} \begin{bmatrix}\ell_0\\1.3\\1.3\end{bmatrix} \xrightarrow[a]{\text{action}} \begin{bmatrix}\ell_1\\1.3\\0\end{bmatrix} \xrightarrow[c]{\text{action}} \begin{bmatrix}\ell_3\\1.3\\0\end{bmatrix} \xrightarrow[0.7]{\text{delay}} \begin{bmatrix}\ell_3\\2\\0.7\end{bmatrix} \xrightarrow[e]{\text{action}} \begin{bmatrix}\ell_3\\2\\0.7\end{bmatrix}$$

where the first component of a configuration is the location and the second and third components give the values of clocks x and y, respectively. This execution corresponds to a delay of 1.3 time units in ℓ_0, the firing of transition a

(which is enabled because the value of clock x is less than two; clock y is then set to 0), the firing of transition c (which occurs without delay as ℓ_1 is urgent), etc.

In the context of verification, several problems are of interest, like the model-checking of safety properties ("Can a distinguished set of states be avoided?"), reachability/liveness properties ("Can/will a distinguished goal state be reached?"), or more involved properties such as response properties ("Is any request eventually granted?"). As a model for real-time systems, these properties can include quantitative constraints, for instance time-bounded reachability, or time-bounded response properties ("Is any request granted within two minutes?"). It is also relevant to compute optimal time bounds for these properties, for example, optimal-time reachability ("What is the minimum time required for reaching a distinguished set of states?").

The region abstraction. A timed automaton is a syntactical representation of an infinite transition system, since clocks take (nonnegative) real values. However, there is a way to deal with this infinity of configurations by reasoning symbolically: the main theoretical ingredient for solving problems on timed automata is the notion of *regions*,[5] which provide a finite partitioning of the state space such that states within a given region are *bisimilar*, that is, behaviorally indistinguishable.

The precise definition of regions is such that inside a region, integral parts of clock values do not change, and also the ordering of clocks according to their values' fractional parts stays the same. Special consideration has to be given to the cases where one or more clock values are integers, and finiteness of the region partitioning is ensured by considering as equivalent all clock values that exceed the maximal constant appearing in guards and invariants of the timed automaton in question. In the left part of Figure 2 we show the 44 regions for two clocks x and y with maximal constant equal to two. In this two-clock case, regions can be points (both clocks have integer values), open line segments (one clock has integer value, or their fractional parts are equal), open triangles, or open unbounded rectangles.

From two equivalent configurations (same location, region equivalent valuations), by delaying or by taking a transition, similar regions will be visited and similar behaviors will be possible. Regions are thus a way to finitely abstract the behaviors of a timed automaton. There are finitely many regions, and by considering as abstract configurations pairs of locations and regions, we get a finite automaton, called the *region automaton*, which preserves many properties including reachability, liveness, and safety. Hence, verification of those properties on the original timed automaton can be transferred to the finite region automaton and then checked using standard algorithms.

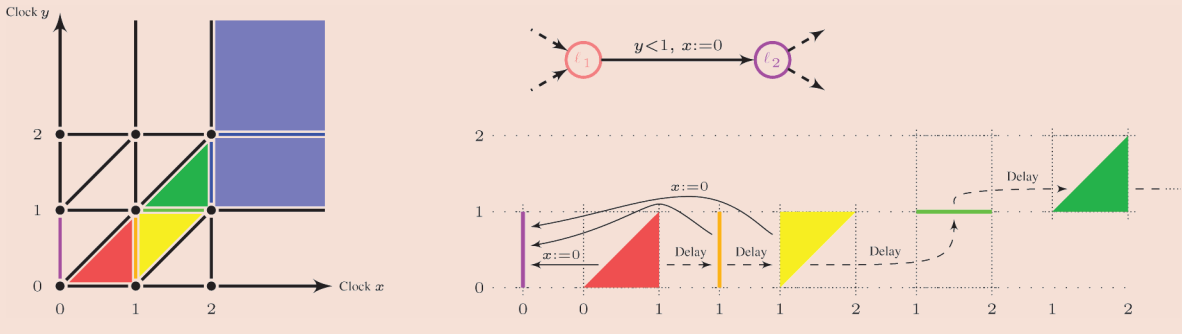
The limits of the region abstraction. Not all properties can be decided on timed automata using the region abstraction, and problems such as checking inclusion ("Are all real-time behaviors of a timed automaton also behaviors of another timed automaton?") and universality ("Can all real-time behaviors be realized in a given timed automaton?") are undecidable.

Also, the set of real-time behaviors exhibited by timed automata is not closed under complement, and not all timed automata are determinizable. As a counterexample for these properties, one can use the following timed automaton:

It accepts all behaviors with at least two a's separated by one time unit. It can be shown that no deterministic timed automaton exists with exactly the same behaviors, and also that no timed automaton can implement precisely all complementary behaviors.

Timed automata in practice. The region abstraction is a powerful tool for showing decidability of a number of interesting properties, but unfortunately, the region-based verification algorithms introduced above are infeasible in practice. The number of regions grows exponentially with the number of clocks, hence these algorithms have exponential time complexity. Using *on-the-fly* techniques, one can reduce the complexity to polynomial-space, and indeed it can be shown[4] that, for example, the reachability problem is PSPACE-complete.

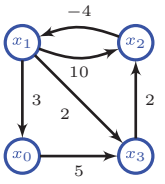
Figure 2. *The region abstraction* is a finite representation of all possible behaviors of the timed automaton. Consider the timed automaton on top of the picture, and assume we enter location ℓ_1 with clock values (x_0, y_0) for which $0 < y_0 < x_0 < 1$ (a point in the red triangle, see the picture on the left); as clock y has value strictly less than 1, we have the option to switch to location ℓ_2, which would reset clock x and end up in the purple region. We also have the option to delay in ℓ_1; in that case, we exit the red triangle and reach the orange line. Here again we have two options: switching to ℓ_2, or delaying to the yellow clock region. In case we still decide to wait in that region, we reach the green line. From that region on, the transition to ℓ_2 is not enabled anymore. This description of the possible behaviors starting from the red region, which has been represented on the picture to the right, does not depend on the precise values of the clocks: region equivalence preserves enough information to encode exactly the behaviors of the underlying timed automaton.



Algorithms that have shown to be feasible, even efficient, in practice are based on the so-called *zone graph* abstraction[30]: a *zone* is a set of clock valuations defined by a clock constraint and can hence be represented by such; the zone graph has as vertices pairs of locations and zones that satisfy the location's invariant, and its edges are derived from the transitions of the given timed automaton. The number of zones is unbounded, so unlike the region graph, the zone graph is infinite. Finiteness can be enforced using a technique known as normalization[12]; however, the number of zones is still much larger than the number of regions, and moreover the same zone can be represented using many different clock constraints.

The reason for zone-based algorithms to be efficient in practice is twofold: First, the algorithms used have no need to explore all of the zone graph (they work *on-the-fly*), and zones are commonly bigger than regions, hence the part of the zone graph to be explored is smaller. Second, operations on zones can be implemented very efficiently (in time cubic in the number of clocks): zones are usually represented using *difference-bound matrices*, or DBMs. The DBM representation of a zone on a set of k clocks has $(k+1)\times(k+1)$ entries, where an entry $c_{i,j}$ represents a clock constraint $x_i - x_j \leq c_{i,j}$ and an extra clock x_0 is added to represent absolute clock constraints $x_i \leq c_{i0}$. DBMs in turn can be represented as directed weighted graphs; see below for an example of a zone and its DBM (graph) representation. Canonical representations of zones can be obtained using shortest-path closure or shortest-path reduction of their DBM graphs, and delay and reset operations on zones can be efficiently implemented on the DBM representations.

$$Z = \begin{cases} x_1 \leq 3 \\ x_1 - x_2 \leq 10 \\ x_1 - x_2 \geq 4 \\ x_1 - x_3 \leq 2 \\ x_3 - x_2 \leq 2 \\ x_3 \geq -5 \end{cases}$$

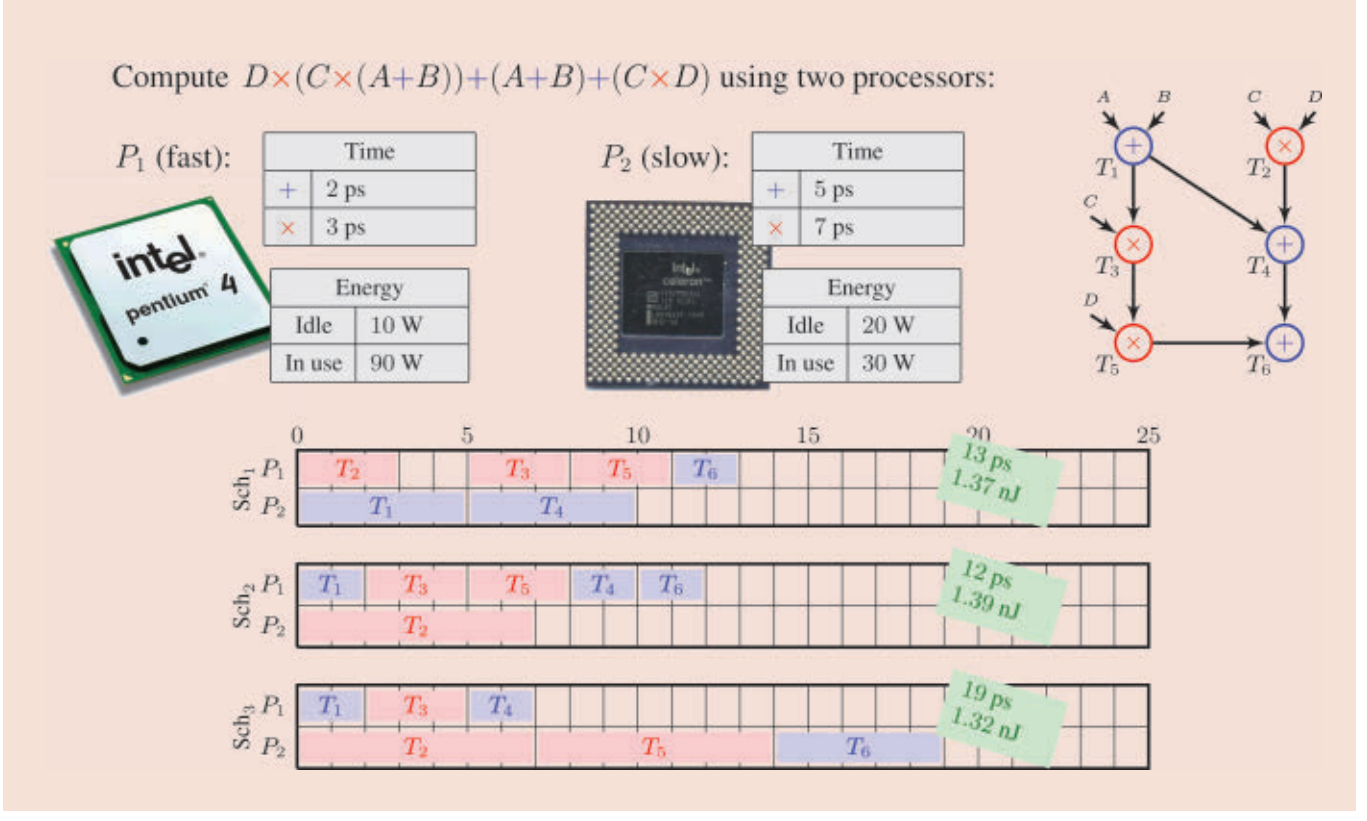


Task graph scheduling: time optimality. A task graph problem involves a number of tasks $T_1,\ldots,T_m$, a number of machines or processors $P_1,\ldots,P_n$, and a (partial) mapping d giving, for each task T_i and processor P_j, the time $d(i,j)$ for computing T_i on P_j. In addition there is a partial order on the tasks used for describing dependencies. Figure 3 is an example of a task graph problem.

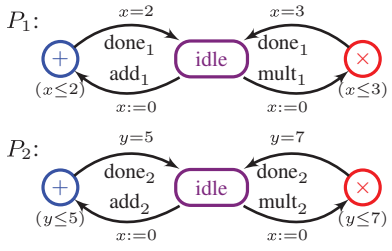
We want to determine a schedule of when to start the execution of tasks, and on which processors, that minimizes the total execution time while being feasible in respecting the following conditions: (a) a task can be executed only if all its predecessors have completed; (b) each machine can process at most one task at a time; (c) tasks cannot be preempted.

Task graph scheduling problems may be easily modeled as networks of timed automata so that every run corresponds to a feasible schedule and the fastest run gives the time-optimal schedule: for each processor we construct a small timed automaton able—when idle—to handle within the appropriate amount of time the

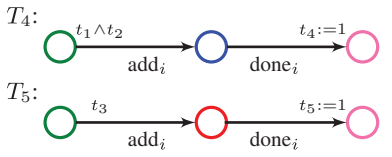
Figure 3. *Task graph problem* with six tasks, where each task corresponds to the computation of a given sub-expression of the term $(D\times(C\times(A+B)))+((A+B)+(C\times D))$. Given the execution platform with two processors, P_1 and P_2, and corresponding computation times for addition and multiplication, as well as their energy consumption, Sch_1–Sch_3 provide three feasible schedules, where Sch_2 is in fact time-optimal, and Sch_3 is energy-optimal.



requests from the tasks. For the processors of Figure 3, these are as follows:



Each task is modeled as a timed automaton waiting to be served by either of the processors, conditioned by the completion of its predecessors (indicated by Boolean variables t_1–t_5). Tasks T_4 and T_5 of our example can be represented as follows:



Extensive experiments on benchmarks have demonstrated that the above timed automata approach to task graph scheduling is competitive compared with more traditional approaches from operations research (for example, mixed-integer linear programming) as well as specialized, heuristic algorithms from planning and scheduling.[1] Furthermore, the generic approach of timed automata admits easy incorporation of more specialized features (for example, release times, deadlines) to the models and scheduling.

Extensions of timed automata. Timed automata are a rich extension of classical automata with efficient tool support and several successful industrial applications, as we will discuss later. As such, they are often cited as the model of choice for representing and reasoning about embedded and real-time systems.

This success has led to several extensions of the model, for instance with more general guards or resets being allowed (for example, additive guards[11] or non-deterministic updates of clocks[12]), or with more involved dynamics measuring other quantities than time. Unfortunately, these extensions quickly lead to undecidability; for example, for timed automata in which clocks can be stopped (so-called *stopwatch automata*), even basic properties such as safety or liveness are undecidable.[29]

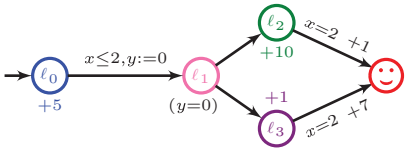
On the other hand, the model of *hybrid automata*,[29] though suffering from the same undecidability problems as mentioned for other classes above, has emerged as a popular formalism for which semi-decision and approximation procedures have been developed. The model of *priced timed automata*, which we shall discuss next, form an intermediate class between timed and hybrid automata for which some of the good decidability properties of timed automata are retained. Other intermediate classes of models have been investigated, including *linear hybrid automata*[29] and *integration graphs*,[33] providing semi-decidability in general and decidability under certain restrictions.

Priced Timed Automata

A model for resources. Time is not the only quantitative notion of interest when designing embedded systems; other quantities such as energy or memory consumption, required bandwidth, or accumulated cost can be important to measure in such systems.

These notions are intimately connected to time, because the longer the device is operating, the more resources it consumes. This makes timed automata the model of choice to reason about those quantities, and has led to the definition of *priced timed automata*,[6,10] extending timed automata with *cost* (which is the general name we will use in the sequel to refer to the various quantities that can be modeled within this formalism; in some other literature, this is referred to as *reward*).

A priced timed automaton is hence a timed automaton with extra information indicating how the cost is evolving in locations and during transitions. To avoid the undecidability problems of hybrid automata, cost information cannot be used to guard transitions; the cost is only an *observer variable*, and whether a transition is enabled only depends on timing information, not cost value. An example of a priced timed automaton, extending the timed automaton of the previous section, is depicted below (labels omitted):



A decoration +10 on a location indicates that cost increases by 10 units per time unit in the location; a decoration +7 on a transition indicates that taking the transition increases overall cost by 7 units (locations and transitions without cost indication have cost 0). The executions of such an automaton are those of the underlying timed automaton. The total cost of the example execution given earlier (delaying 1.3 time units in ℓ_0, 0.7 time units in ℓ_3, and ending in the rightmost location) can be computed as

$$1.3 \times (+5) + 0.7 \times (+1) + 7 = 14.2$$

Optimizing the resources. Natural optimization questions can be posed on that model, for example, the *optimal reachability* problem (minimum cost for reaching a given goal), the *mean-cost optimization* problem (mean cost used in the long run), or the *discounted-cost optimization* problem (where costs are discounted exponentially as time elapses).

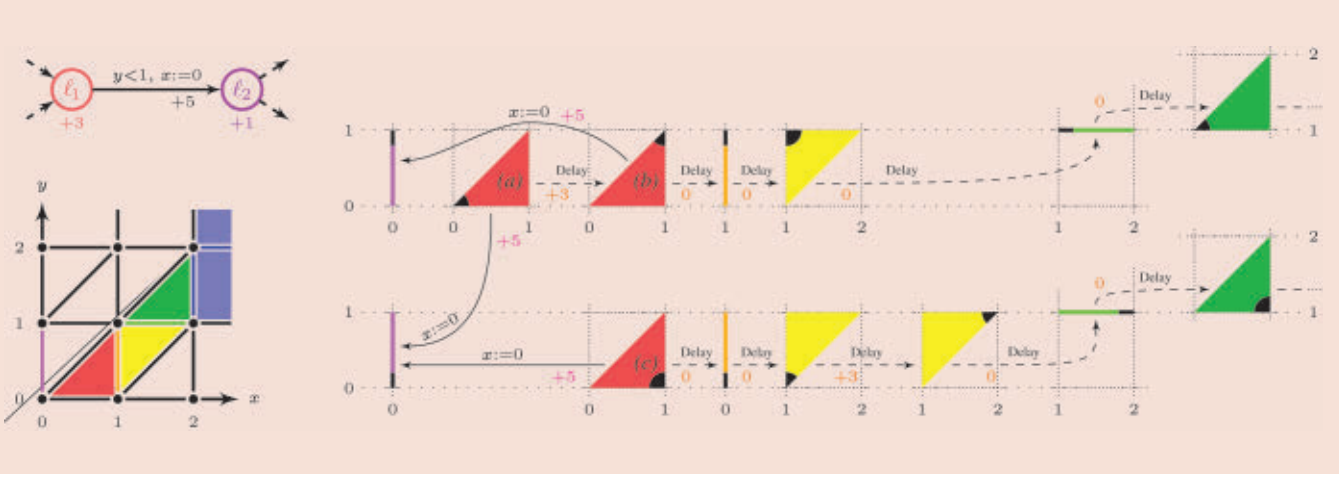
As an example, we compute the minimum cost that is required for reaching location ☺ in the previous example. There are two families of executions: those that go through ℓ_2 and those that go through ℓ_3. Furthermore, in each family, there is a single parameter t: the time elapsed in location ℓ_0; everything else is determined by the guards in the automaton. Hence the minimum cost is:

$$\inf_{0\leq t\leq 2} \min \begin{pmatrix} 5t+10(2-t)+1 \\ 5t+(2-t)+7 \end{pmatrix} = 9 \quad (1)$$

where the expressions $5t + 10(2 - t) + 1$ and $5t + (2 - t) + 7$ give the cost of executions going through ℓ_2 respectively ℓ_3 after delaying t time units in location ℓ_0.

The standard region construction is not accurate enough to properly keep track of cost information, and a refinement of the region abstraction, the *corner-point abstraction*,[13] has to be used to solve the optimization problems mentioned above. For this abstraction, regions are refined by distinguishing their corner points. As an

Figure 4. ***The corner-point abstraction*** **refines the region abstraction by also keeping track of the corner point close to which an execution runs. This is needed to measure costs: for instance, if we are in location ℓ_1 and in the red region where $0 < y < x < 1$, the price of delaying depends on the value of the clocks. From *(a)*, where both x and y are arbitrarily close to 0, we can let almost one time-unit elapse and reach *(b)*. The resulting cost is arbitrarily close to +3. On the other hand, from *(c)*, where x and y are arbitrarily close to 1 and 0, respectively, letting time elapse takes us to the subsequent region, so that the cost is arbitrarily close to 0. (Notice that for readability, some resetting transitions have been omitted.)**



example, the two-dimensional region depicted below is refined into three region-corner pairs; the meaning of a region-corner pair is that the current clock valuation is arbitrarily close to the distinguished corner:

Similar to the refinement of regions, the transitions in the region automaton have to be refined to keep track of the corners. In the example above, there is a (delay) transition from region-corner pair *(a)* to *(b)*, whereas *(c)* cannot be reached neither from *(a)* nor from *(b)*. Figure 4 illustrates the corner-point abstraction of an example priced timed automaton. This graph has two types of delay edges: either within a region, from one corner to another one, or from a corner of a region to the corresponding corner in the subsequent region. The first case corresponds to a delay of "almost" one time unit, while the second case corresponds to a delay of "almost" zero time units. In addition, there are edges representing transitions of the timed automaton (which reset clock x in our example of Figure 4). In that case as well, there is a natural mapping between corners.

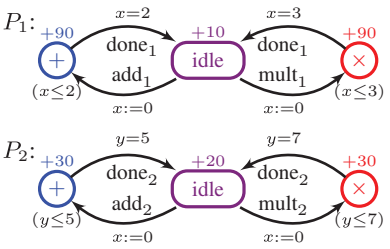
The edges of the corner-point abstraction are labeled with discrete cost information: if the cost rate in the current location is +3, all one time-unit edges have label +3, and all zero time-unit edges get label 0. Edges coming from discrete transitions are labeled with the cost of the transition (+5 in the example).

The corner-point abstraction can be used to solve many optimization problems, as it can be shown that in these cases, optimal total cost is obtained for runs that always take transitions close to integer clock values. Hence the optimization problem reduces to a problem on a finite graph that can be solved using different standard techniques. This is the case for the mean-cost optimization problem[13] and the discounted-cost problem.[25] For optimal reachability, another technique (*priced regions*) has been used[10] that also extends to a setting of more than one cost variable.[34]

As for algorithm and tool support, the zone-based approach has been successfully extended to solve the optimal reachability problem,[35] by introducing *priced zones*, and tool support is available in UPPAAL CORA. For mean-cost and discounted-cost optimization, active research is being conducted in developing efficient zone-based algorithms, or alternatively showing that no such algorithms exist.

Task graph scheduling: energy optimality. Reconsidering our running task graph scheduling problem, cost-optimal reachability for priced timed automata may be used to provide energy-optimal schedules. The energy needed for performing computation steps is modeled as cost in the priced timed automaton model, and optimal reachability techniques can be used for finding an energy-optimal schedule.

For the task graph scheduling instance of Figure 3, energy consumption of the two processors is reflected in the respective timed automata by suitable cost-rates in the locations corresponding to the processor being idle or in use. The processors can then be represented by the following two priced timed automata:



Managing the resources. Up to this point we have only employed priced timed automata as a formalism for modeling time-dependent *consumption* of resources. However, in several situations resources may not just be consumed but also occasionally regained, for example, in autonomous robots with rechargeable batteries, or in tanks which may not only be emptied but also filled. Extending priced timed automata to allow for both positive (regaining) and negative (consumption) rates provides a natural modeling formalism.[21]

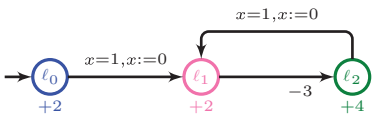
However, a new question now

emerges related to the appropriate management of resources: "Is it possible to maintain the level of resources within fixed bounds?" Such resource-bound problems are highly relevant to the analysis of several embedded systems, for example, it is natural to plan the usage of a device with rechargeable batteries so that one never runs out of energy, nor exceeds the maximum capacity for energy storage. Figure 5 shows a priced timed automaton together with some resource management problems.

Few results have been obtained on this problem so far: only the case of *one-clock* priced timed automata has been investigated.[15] This restriction has two important consequences: cycle detection can be done statically, as each resetting transition leads to a configuration with clock value 0, and the region automaton can be coarsened so that the partition consists of intervals with end-points given by the constants in the automaton's guards. As a consequence, there are only polynomially many regions.

Under the additional assumption that the cost cannot be updated during transitions (hence cost evolves only in locations), it can be shown[15] that for finding runs that satisfy a global lower-bound constraint, with or without soft upper bound, one can restrict oneself to look for runs with *integral* delays. Hence, the corner-point abstraction can be used for this, and the problems are solvable in polynomial time.

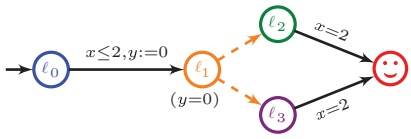
For priced timed automata with more than one clock, no results are known, but even for one-clock automata with cost updates during transitions, there are some difficulties that mean abstractions like the above are insufficient. As an example, consider the following priced timed automaton:



Assuming that we start with initial cost 0, this automaton has exactly one feasible execution in which the cost level remains non-negative: after spending one time unit in location ℓ_0, we alternately spend half a time unit in ℓ_1 and half a time unit in ℓ_2. Any other execution eventually violates the lower bound. Hence in this case, runs satisfying the lower bound cannot be found using the corner-point abstraction.

Priced Timed Games

A model for uncertainties. The systems we have considered so far are *closed* in the sense that we have a complete description of the system. This is not sufficient to model embedded systems where interaction with the environment is crucial, or systems with some imprecision. These can be modeled using (two-player) *timed games*,[8] in which some actions are triggered by the environment (we can think of signals received by sensors, or of unexpected events). The aim is to *control*, or *guide*, the system so that it will be safe or correct regardless of the way the environment interferes. An example of a timed game is depicted below:

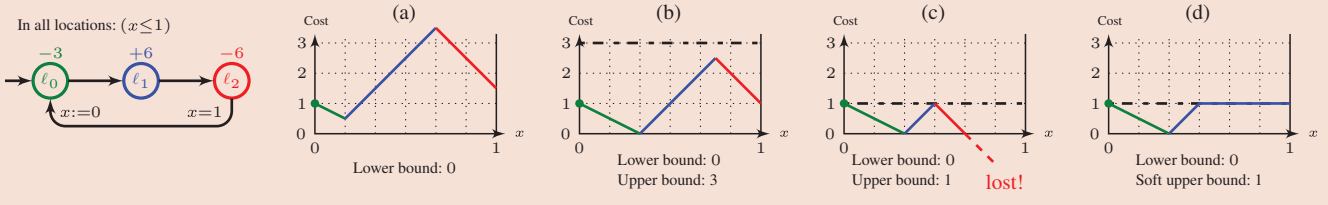


Dashed edges belong to the environment (they are *uncontrollable*): when they are fireable, the system cannot prevent (nor force) them to be fired. Here, the system cannot decide whether it goes through ℓ_2 or through ℓ_3.

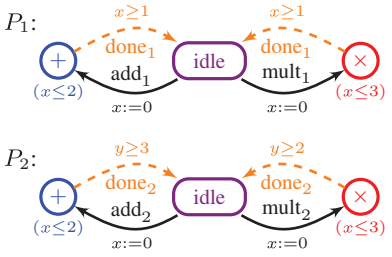
For simple correctness criteria, for example, reachability or safety, the set of *winning states* (that is, states from which the system can be controlled under the safety constraint) and also *winning strategies* (that is, policies for how to control the system) can be computed using the region abstraction.[8] Also computability of time-optimal strategies,[7] as well as strategies under partial observability, has been demonstrated. For the latter, decisions are based on discrete observations giving only partial information of the system state, depending on the availability and precision of sensors.[19] For efficient algorithms, a zone-based approach for solving timed games with reachability and safety objectives has been developed,[18, 38] and tool support is available in UPPAAL-TIGA.

Task graph scheduling: timing uncertainty. Returning to our running task graph scheduling example, we can use the formalism of timed games to model uncertainty in precisely how much time a certain computation on a given processor takes. Previously, we modeled computation times by precise numbers, whereas we now can make the model more realistic by only providing *interval bounds* within which computation times are prescribed to lie. The timed game models below provide versions of the processors P_1 and P_2 from Figure 3 in which computation times are prescribed to lie in the intervals [1, 2] for addition and [1, 3]

Figure 5. *The resource management problem* asks whether it is possible to maintain the cost level within fixed bounds. There can be a lower bound only (a), a lower and an upper bound (b, c), or a lower bound and a soft upper bound above which cost level cannot increase. Figures (a), (b), and (d) represent solutions to the respective problems for the priced timed automaton depicted on the left: there is an infinite run that satisfies the global constraint. In case (a) for instance, we have depicted a possible schedule for the first cycle, and this run can be repeated because at the start of the second cycle, the cost level is larger than at the start of the first cycle. In Figure (c), the proposed schedule violates the lower bound, and it can be shown that there exists no infinite run which maintains cost level within the specified bounds.

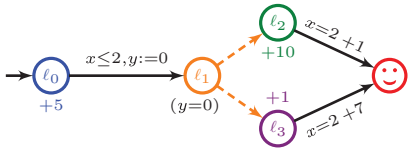


for multiplication on P_1, and similarly for P_2.



Using these models, a computed time-optimal schedule will no longer be a simple fixed assignment of tasks and time slots to processors, but rather a flexible dynamic assignment, where task scheduling can be adapted online according to actual completion times of previous tasks. (Hence, we cannot display the solution here.)

Cost-optimal strategies. It is natural to extend the timed game framework with cost information, hence making it possible to model uncertainty as well as resource use, and to ask for *controllability under resource constraints*, or for *optimal* controllability. The model of *priced timed games* is a synthesis of priced timed automata and timed games; we show an example below:



In this example we may, for example, want to compute the minimum cost for reaching location ☺ regardless of the moves of the environment (which is in charge of the edges out of ℓ_1 as before). As the system cannot control whether execution goes through ℓ_2 or ℓ_3, the minimum cost is given by the term

$$\inf_{0\le t\le 2} \max\begin{pmatrix} 5t+10(2-t)+1 \\ 5t+(2-t)+7 \end{pmatrix} = 14+\frac{1}{3}$$

where t is the delay spent in location ℓ_0. Solving this, one arrives at a minimum cost of $14\frac{1}{3}$, which is attained for $t=\frac{4}{3}$. As this is not an integer, one sees that techniques based on the corner-point abstraction are not sufficient for computing optimal-reachability strategies, even in case of one-clock priced timed games.

Generally, priced timed games are much more difficult to analyze than priced timed automata. Using reductions from the Halting problem for two-counter machines, one can show that cost-optimal strategies are undecidable,[17] even when restricted to priced timed games with only three clocks.

Decidability has been shown for classes of priced timed games with strong conditions on the cost evolution[3] and for one-clock priced timed games.[14] The reason for the latter is the same as for one-clock priced timed automata above: resetting the clock leads to a configuration with a known clock valuation.

Applications and Tools

Timed automata and their extensions have been applied to the modeling, analysis, and optimization of numerous real-time applications. Here, we give a few examples, not aiming at being exhaustive but rather to illustrate the wide range of application domains.

A variety of mature tools are available that provide important computer-aided support for applications. Well-known tools include UPPAAL, KRONOS, and HYTECH, but there is a large number of other tools available. The electronic version of this article contains an extra section that aims to give an overview together with references to the individual tools.

The timed automata formalism is now routinely applied to the modeling and analysis of real-time control programs, including a wide class of programmable logic controller (PLC) control programs[23,36] and timing analysis and code generation of vehicle control software,[39] and the timed automaton approach has also demonstrated its viability to the timing analysis of certain classes of asynchronous circuits.[16]

Similarly, numerous real-time communication protocols have been analyzed using timed automata technology, often with inconsistencies being revealed: for example, using real-time model checking, the cause of a 10-year-old bug in the IR-link protocol used by Bang & Olufsen was identified and corrected.[27] Most recently, real-time model checking has been applied to the clock synchronization algorithm currently used in a wireless sensor network that has been developed by the Dutch company CHESS.[37] Here it is shown that in certain cases a static, fully synchronized network may eventually become unsynchronized if the current algorithm is used, even in a setting with infinitesimal clock drifts.

During the last years, timed automata modeling of multitasking applications running under real-time operating systems has received substantial research effort. Here the goals are multiple: to obtain less pessimistic worst-case response time analysis compared with classical methods for single-processor systems[40]; to relax the constraints of period task arrival times of classical scheduling theory to task arrival patterns that can be described using timed automata[26]; to allow for schedulability analysis of tasks in terms of concurrent objects executing on multiprocessor or distributed platforms (for example, MPSoC).[22]

Just as symbolic reachability checking of finite-state models has led to very efficient planning and scheduling algorithms, reachability checking for (priced) timed automata has demonstrated competitive and complementary performance with respect to classical approaches such as MIPL on optimal scheduling problems involving real-time constraints, for example, job-shop and task-graph scheduling[1,9] and aircraft landing problems.[35] In fact a translation of the variant PDDL3 of PDDL (Planning Domain Definition Language) into priced timed automata has been made[24] allowing optimal planning questions to be answered by cost-optimal reachability checking. Industrial applications include planning a wafer scanner from semiconductor industry[28] and computation of optimal paper paths for printers.[31]

Most recently, computation of winning strategies for timed games has been applied to controller synthesis for embedded systems, including synthesis of most general non-preemptive online schedulers for real-time systems with sporadic tasks,[2] synthesis of climate control for pig stables provided by the company Skov A/S,[32] and automatic synthesis of robust and near-optimal controllers for industrial hydraulic pumps.[20]

Conclusion

Timed automata and their priced and game extensions provide a uniform and expressive formalism for dynamic resource allocation problems with hard real-time constraints, that is, timing constraints that must be satisfied under all circumstances. This is in contrast to soft real-time constraints, which only need to be met with a certain probability, .999 say, and which require stochastic modeling formalisms such as discrete-time or continuous-time Markov chains, queueing models. While hard real-time focuses on worst-case analysis, soft real-time addresses more refined properties such as average-case performance.

However, within the setting of hard real-time, timed automata and their extensions allow for analysis of a wide collection of performance and optimization problems, with results competitive with respect to more traditional approaches such as mixed-integer linear programming or others.

Particularly challenging problems remaining to be settled include decidability of synthesis for priced timed games under partial observability, as well as a range of resource management problems in the setting of priced timed automata and games with both consumption and regaining of resources.

Acknowledgments

The authors are partly supported by the European project QUASIMODO (FP7-ICT-STREP-214755). The French authors are supported by project DOTS (ANR-06-SETI-003). The Danish authors are supported by the Danish Center of Excellence MT-LAB. C

References

1. Abdeddaïm, Y., Asarin, E., Maler, O. Scheduling with timed automata. *Theor. Comput. Sci. 354*, 2 (2006), 272–300.
2. Altisen, K. et al. A framework for scheduler synthesis. In *IEEE Real-Time Systems Symposium*, 1999, 154–163.
3. Alur, R., Bernadsky, M., Madhusudan, P. Optimal reachability in weighted timed games. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP '04)*, Lecture Notes in Computer Science 3142, Springer, 2004, 122–133.
4. Alur, R., Courcoubetis, C., Dill, D.L. Model-checking for real-time systems. In *Proceedings of the 5th Annual Symposium on Logic in Computer Science (LICS '90)*, IEEE Computer Society Press, 1990, 414–425.
5. Alur, R., Dill, D.L. Automata for modeling real-time systems. In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming (ICALP '90)*, Lecture Notes in Computer Science 443, Springer, 1990, 322–335.
6. Alur, R., La Torre, S., Pappas, G.J. Optimal paths in weighted timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, Lecture Notes in Computer Science 2034, Springer, 2001, 49–62.
7. Asarin, E., Maler, O. As soon as possible: Time optimal control for timed automata. In *Proceedings of the 2nd International Workshop on Hybrid Systems: Computation and Control (HSCC'99)*, Lecture Notes in Computer Science 1569, Springer, 1999, 19–30.
8. Asarin, E. et al. Controller synthesis for timed automata. In *Proceedings of IFAC Symposium on System Structure and Control*, Elsevier Science, 1998, 469–474.
9. Behrmann, G. et al. Efficient guiding towards cost-optimality in UPPAAL. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '01)*, Lecture Notes in Computer Science 2031, Springer, 2001, 174–188.
10. Behrmann, G. et al. Minimum-cost reachability for priced timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control (HSCC '01)*, Lecture Notes in Computer Science 2034, Springer, 2001, 147–161.
11. Bérard, B., Dufourd, C. Timed automata and additive clock constraints. *Inf. Process. Lett. 75*, 1–2 (2000), 1–7.
12. Bouyer, P. Forward analysis of updatable timed automata. *Form. Methods Syst. Des. 24*, 3 (2004), 281–320.
13. Bouyer, P., Brinksma, E., Larsen, K.G. Optimal infinite scheduling for multi-priced timed automata. *Form. Methods Syst. Des. 32*, 1 (2008), 2–23.
14. Bouyer, P. et al. Almost optimal strategies in one-clock priced timed automata. In *Proceedings of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS '06)*, Lecture Notes in Computer Science 4337, Springer, 2006, 345–356.
15. Bouyer, P. et al. Infinite runs in weighted timed automata with energy constraints. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'08)*, Lecture Notes in Computer Science 5215, Springer, 33–47.
16. Bozga, M. et al. Verification of asynchronous circuits using timed automata. *Electron. Notes Theor. Comput. Sci. 65*, 6 (2002), 47–59.
17. Brihaye, Th., Bruyère, V., Raskin, J.-F. On optimal timed strategies. In *Proceedings of the 3rd International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS '05)*, Lecture Notes in Computer Science 3821, Springer, 2005, 49–64.
18. Cassez, F. et al. Efficient on-the-fly algorithms for the analysis of timed games. In *Proceedings of the 16th International Conference on Concurrency Theory (CONCUR '05)*, Lecture Notes in Computer Science 3653, Springer, 2005, 66–80.
19. Cassez, F. et al. Timed control with observation based and stuttering invariant strategies. In *Proceedings of the 5th International Symposium on Automated Technology for Verification and Analysis (ATVA '07)*, Lecture Notes in Computer Science 4762, Springer, 2007, 192–206.
20. Cassez, F. et al. Automatic synthesis of robust and optimal controllers—An industrial case study. In *Proceedings of the 12th International Workshop on Hybrid Systems: Computation and Control (HSCC '09)*, Lecture Notes in Computer Science 5469, Springer, 2009.
21. Chakrabarti, A. et al. Resource interfaces. In *Proceedings of the 3rd International Workshop on Embedded Software (EMSOFT '03)*, Lecture Notes in Computer Science 2855, Springer, January 2003.
22. David, A. et al. Model-based framework for schedulability analysis using UPPAAL 4.1, chapter 4. *Model-Based Design for Embedded Systems*. G. Nicolescu and P.J. Mosterman, eds. CRC Press, Boca Raton, FL, 2009, 93–119.
23. Dierks, H. PLC-automata: A new class of implementable real-time automata. *Theor. Comput. Sci. 253*, 1 (2001), 61–93.
24. Dierks, H. Finding optimal plans for domains with continuous effects with UPPAAL CORA. In *Proceedings of the ICAPS '05 Workshop on Verification and Validation of Model-Based Planning and Scheduling Systems*, 2005.
25. Fahrenberg, U., Larsen, K.G. Discount-optimal infinite runs in priced timed automata. *Electron. Notes Theor. Comput. Sci. 239* (2009), 179–191.
26. Fersman, E. et al. Schedulability analysis of fixed-priority systems using timed automata. *Theor. Comput. Sci. 354*, 2 (2006), 301–317.
27. Havelund, K. et al. Formal modeling and analysis of an audio/video protocol: An industrial case study using UPPAAL. In *Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS '97)*, IEEE Computer Society Press, 1997, 2–13.
28. Hendriks, M., van den Nieuwelaar, B., Vaandrager, F.W. Model checker aided design of a controller for a wafer scanner. *STTT 8*, 6 (2006), 633–647.
29. Henzinger, Th.A. et al. What's decidable about hybrid automata? *J. Comput. Syst. Sci. 57*, 1 (1998), 94–124.
30. Henzinger, Th.A. et al. Symbolic model-checking for real-time systems. *Inf. Comput. 111*, 2 (1994), 193–244.
31. Igna, G. et al. Formal modeling and scheduling of datapaths of digital document printers. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS '08)*, Lecture Notes in Computer Science 5215, Springer, 2008, 170–187.
32. Jessen, J.J. et al. Guided controller synthesis for climate controller using UPPAAL-TIGA. In *Proceedings of the 5th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS '07)*, Lecture Notes in Computer Science 4763, Springer, 2007, 227–240.
33. Kesten, Y. et al. Decidable integration graphs. *Inf. Comput. 150*, 2 (1999), 209–243.
34. Larsen, K.G., Rasmussen, J.I. Optimal reachability for multi-priced timed automata. *Theor. Comput. Sci. 390*, 2–3 (2008), 197–213.
35. Larsen, K.G. et al. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proceedings of the 13th International Conference on Computer Aided Verification (CAV '01)*, Lecture Notes in Computer Science 2102, Springer, 2001, 493–505.
36. Mader, A., Wupper, H. Timed automaton models for simple programmable logic controllers. In *Proceedings of the 11th Euromicro Conference on Real-Time Systems (ECRTS '99)*, IEEE Computer Society, 1999, 106–113.
37. Schuts, M. et al. Modelling clock synchronization in the Chess gMAC WSN protocol. *CoRR*, abs/0912.1901, 2009.
38. Tripakis, S., Altisen, K. Controller synthesis for discrete and dense-time systems. In *Proceedings of the World Congress on Formal Methods in the Development of Computing Systems (FM '99)*, Lecture Notes in Computer Science 1708, Springer, 233–252, 1999.
39. Tripakis, S., Yovine, S. Timing analysis and code generation of vehicle control software using TAXYS. *Electron. Notes Theor. Comput. Sci. 55*, 2 (2001), 277–286.
40. Waszniowski, L., Hanzálek, Z. Formal verification of multitasking applications based on timed automata model. *Real-Time Syst. 38*, 1 (2008), 39–65.

Patricia Bouyer (bouyer@lsv.ens-cachan.fr), LSV–CNRS & ENS Cachan, France.

Uli Fahrenberg (uli@cs.aau.dk), Department of Computer Science, Aalborg Universitet, Aalborg, Denmark.

Kim G. Larsen (kgl@cs.aau.dk), Department of Computer Science, Aalborg Universitet, Aalborg, Denmark.

Nicolas Markey (markey@lsv.ens-cachan.fr), LSV-CNRS & ENS Cachan, France.

CAREERS

Michigan State University
Tenure-Stream Faculty -
Computer Science and Engineering

The Department of Computer Science and Engineering (CSE) at Michigan State University invites applications for a tenure-stream faculty position in the area of computer vision, image processing, and its applications to biometric recognition. Candidates at all ranks will be considered. The appointment starts in August 2012.

The CSE Department conducts leading-edge research in many areas, with particular strength in software engineering and formal methods, computer networks and security, computer graphics and visualization, bioinformatics and digital revolution, data mining, machine learning and pattern recognition, and natural language processing. The department's external research awards have nearly doubled in the last couple of years. Multidisciplinary research across a broad range of disciplines is strongly encouraged and is being actively pursued by the faculty. Partnering with several other departments and universities, the CSE department is a major contributor and plays an important role in the NSF Science and Technology Center for the study of Evolution in Action (BEACON) on our campus.

Candidates should have a Ph.D. in Computer Science or a closely related field with evidence of research accomplishments, teaching skills, and an ability to work effectively with other researchers. The successful candidate will be expected to develop an externally funded research program of national prominence that includes fundamental research, publications in high quality conferences and journals, and training graduate students. Leadership is expected in development of educational programs to provide state-of-the-art knowledge to both undergraduate and graduate students.

MSU enjoys a large, park-like campus with outlying research facilities and natural areas. The greater Lansing area has approximately 450,000 residents. The local communities have excellent school systems and place a high value on education. The University is proactive in exploring opportunities for the employment of spouses, both inside and outside the University.

Candidates should submit an application for this position through: https://jobs.msu.edu/. Refer to posting #4905. Closing date is December 1, 2011. Applications will be reviewed on a continuing basis until the position is filled. For full consideration, applications should be received by the closing date.

MSU is an affirmative action, equal opportunity employer. MSU is committed to achieving excellence through cultural diversity. The university actively encourages applications and/or nominations of women, persons of color, veterans and persons with disabilities.

Faculty Search Committee
Department of Computer Science and Engineering
3115 Engineering Building
Michigan State University
East Lansing, Michigan 48824-1226
Apply at: https://jobs.msu.edu/
http://www.cse.msu.edu

Ozarks Technical Community College
Programmer

The Programmer is responsible for the design, implementation, maintenance, and monitoring of complex systems on the college's Student Information System (SIS). The Programmer implements and maintains assigned modules on the SIS. The Programmer coordinates with other areas of Information Technology to integrate external systems with the data contained on the SIS.

Bachelors Degree in Computer Science or a related program from a regionally accredited institution of higher learning or 3 years of object oriented programming experience. Significant experience in a modern programming language such as Java, .NET or C# (Applicants may be required to demonstrate programming proficiency) Significant experience with relational databases. Effective communication skills.

Apply URL: http://www.otc.edu/jobs/jobs.php

OZ Management
SharePoint Developer

SharePoint Developer, New York, NY. Oversee SharePoint architecture w/in firm, & design, dvlp & implement individual projects incl information portals, self service portals, document libraries, BI portals, & forms-based workflows. Requires Master's degree in Comp Sci or foreign equiv w/3 yrs web dvlpt exp, focused on MS SharePoint (in lieu of Master's and 3 years, employer will accept Bachelor's degree or foreign equivalent in Comp Sci w/5 yrs progressively resp web dvlpt exp, of which at least 3 must have focused on Microsoft SharePoint). Exp must incl proficiency w/MS SharePoint 2003/2007 & MOSS 2007 platform incl: creating page layouts & master pages, site definitions & templates, Features & Solutions, web parts, dash-boarding & BI & BDC. Solid understanding of SP Object Model, Solution Framework, Administration, Configuring SP Webservices , Design & programming exp. using C#, ASP.net, CAML, CSS, XSLT, Windows Workflow Foundation. Expertise in .NET Framework, SQL Server 2000/2005, SSRS, SP Designer, InfoPath, IIS & Active Directory reqd. Mail CV to OZ Management LP, ref job code: "ACMkr" Attn: K. Cubberly, 9 West 57th St, 39th Fl., NY, NY 10019.

Providence College
Assistant Professor of Computer Science

The Mathematics and Computer Science Department at Providence College invites applicants for a tenure-track Assistant Professor position in Computer Science, commencing Fall 2012. Candidate must hold a Ph.D. from an accredited institution in computer science (or earn one by August 1, 2012). Preference will be given to a candidate who specializes in database management. The position requires a commitment to undergraduate teaching (9 credit hours per semester) and continuing scholarship. Details and application instructions are available on the College website at:

http://www.providence.edu/About+PC/Employment+Opportunities/

Preference will be given to applications completed by November 30, 2011. Providence College is a Roman Catholic four-year liberal arts institution conducted under the auspices of the Dominican Friars and seeks candidates who can affirm and contribute to its Mission. An AA/EOE, the College especially encourages applications of women and minorities.

Texas State University-San Marcos
Department of Computer Science

Applications are invited for a tenure-track position at the rank of Assistant Professor. Applicants must have completed all requirements for a PhD with specialization in software engineering by start of employment. Consult the department recruiting page at http://www.cs.txstate.edu/recruitment/faculty_recruit.php for job duties, qualifications, application procedures, and information about the university and the department.

Texas State University-San Marcos will not discriminate against any person in employment or exclude any person from participating in or receiving the benefits of any of its activities or programs on any basis prohibited by law, including race, color, age, national origin, religion, sex, disability, veterans' status, or on the basis of sexual orientation. Texas State University-San Marcos is a member of the Texas State University System.

Toyota Technological Institute Chicago
Computer Science at TTI Chicago
Faculty Positions at All Levels

Toyota Technological Institute at Chicago (TTIC) is a philanthropically endowed degree-granting institute for computer science located on the University of Chicago campus. Applications are being accepted in all areas, but we are particularly interested in machine learning, speech processing, computational linguistics, Computer vision, computational biology and optimization. Positions are available at all ranks, and we have a large number of three year limited term positions currently available. For all positions we require a Ph.D. Degree or Ph.D. candidacy, with the degree conferred prior to date of hire. Submit your application electronically at:

http://ttic.uchicago.edu/facapp/

Toyota Technological Institute at Chicago is an Equal Opportunity Employer

U.S. Air Force Academy
Distinguished Visiting Professor

U. S. AIR FORCE ACADEMY Department of Computer Science is accepting applications for the 2012-2013 Distinguished Visiting Professor position. See http://www.usafa.edu/df/dfcs/index.cfm or call (719) 333-7377 for details. U.S. Citizenship required.

research highlights

P. 90
Technical Perspective
Making Browser Extensions Secure
By Christopher Kruegel

P. 91
Vetting Browser Extensions for Security Vulnerabilities with VEX
By Sruthi Bandhakavi, Nandit Tiku, Wyatt Pittman, Samuel T. King, P. Madhusudan, and Marianne Winslett

P. 100
Technical Perspective
Abstracting Abstract Machines
By Olivier Danvy and Jan Midtgaard

P. 101
Abstracting Abstract Machines
A Systematic Approach to Higher-Order Program Analysis
By David Van Horn and Matthew Might

DOI:10.1145/1995376.1995397

Technical Perspective
Making Browser Extensions Secure

By Christopher Kruegel

THE WORLD WIDE Web has grown tremendously over the last years. To make the rich and dynamic content on the Web accessible to end users, Web browsers have evolved rapidly as well, and new functionalities, often in the form of extensions and plug-ins, are added continuously. As is frequently the case with software, the significant increase in the size and complexity of the code that drives browsers and their extensions has resulted in an increase of program flaws (bugs). Some bugs simply crash the browser. Others, unfortunately, are security vulnerabilities that attackers can use to compromise end users' machines, install malware, and steal sensitive information. Indeed, browser and extension vulnerabilities have become the primary venue through which cyber criminals compromise the security of Web users and, ultimately, earn money.

To prevent attackers from exploiting program flaws, it is critical to identify and fix bugs before the software is deployed. This is particularly important as users are slow in upgrading, even when patches are eventually made available. Expecting developers to write software that is free of any errors is unrealistic. Hence, we need tools that can automatically detect bugs, especially those that can be exploited by attackers.

The following paper describes VEX, a system that specifically focuses on the identification of security vulnerabilities in browser extensions for Firefox. These extensions are JavaScript add-ons that provide new functionality or augment existing features for the Firefox browser. In contrast to the core browser, extensions are often developed by programmers who have less experience in writing secure, robust code. Thus, a tool such as VEX is of particular importance to ensure a secure Web experience.

At the core of VEX is a static source code analysis component that checks for security vulnerabilities that are the result of potentially unsafe data flows. That is, the system scans the JavaScript code of an extension for program paths over which untrusted input, possibly controlled by an attacker, might reach security-relevant functions. If such a path exists, it could be possible for an attacker to craft malicious input that tricks the security-relevant function to do something that was not intended by the developer. For example, attackers could include malicious code in inputs, and this code is later executed by the extension in the context of the browser. This can lead to all kinds of security problems—for example, the attacker could steal a cookie and take over the session between the victim and an online banking site, or the attacker could steal passwords directly from form fields, or he could display a convincing phishing site to the user. Of course, VEX does not find all possible security vulnerabilities, but it covers an important class of common and critical bugs. As always with security, there is no single approach that solves everything, and this system is an important step into the right direction.

The crucial challenge the creators of VEX had to overcome is that static code analysis is a difficult problem. While precise static analysis is hard in general, it is particularly difficult for programs written in JavaScript. The reason is that JavaScript is a very dynamic language. It can execute code dynamically; that is, some parts of the program that will be executed during runtime do not exist in the source code. Instead, they are built by the application while it is running. Moreover, a Firefox extension does not work in isolation, but is tightly integrated with the browser. This means it calls many functions offered by the browser, for example, to access Web pages. Thus, the static analysis cannot look at the program in isolation but must take into account these interactions with the browser as well. I encourage you to read this paper to discover how the authors achieved this analysis.

A tool such as VEX is of particular importance to ensure a secure Web experience.

Static analysis is great because it covers all program paths. However, sound static analysis is also known to raise many false positives (that is, the system claims there is a vulnerability when there is none). VEX strives to strike a balance between trying to cover as many vulnerabilities as possible while making sure that false alarms are minimized. That is, although mistakes are possible, the false positive rates are low. This makes the system useful in practice. After all, for each alert, a human must manually investigate the reported problem.

The authors have demonstrated that VEX works well in practice by running it over 2,460 extensions. The system found a number of security problems, including seven vulnerabilities that were previously unknown. Examples of these bugs, as well as the details of VEX, are detailed in the paper. C

Christopher Kruegel (chris@cs.ucsb.edu) is an associate professor and holder of the Eugene Aas Chair in Computer Science at the University of California, Santa Barbara.

DOI:10.1145/1995376.1995398

Vetting Browser Extensions for Security Vulnerabilities with VEX

By Sruthi Bandhakavi, Nandit Tiku, Wyatt Pittman, Samuel T. King, P. Madhusudan, and Marianne Winslett

Abstract

The browser has become the de facto platform for everyday computation and a popular target for attackers of computer systems. Among the many potential attacks that target or exploit browsers, vulnerabilities in browser extensions have received relatively little attention. Currently, extensions are vetted by manual inspection, which is time consuming and subject to human error. In this paper, we present VEX, a framework for applying static information flow analysis to JavaScript code to identify security vulnerabilities in browser extensions. We describe several patterns of flows that can lead to privilege escalations in Firefox extensions. VEX analyzes Firefox extensions for such flow patterns using high-precision, context-sensitive, flow-sensitive static analysis. We subject 2460 browser extensions to the analysis, and VEX finds 5 of the 18 previously known vulnerabilities and 7 previously unknown vulnerabilities.

1. INTRODUCTION

Driving the Internet revolution is the modern Web browser, which has evolved from a relatively simple client application designed to display static data into a complex networked operating system tasked with managing many facets of a user's online experience. To help meet the varied needs of a broad user population, *browser extensions* expand the functionality of browsers by interposing on and interacting with browser-level events and data. Some extensions are simple and make only small changes to the appearance of Web pages or the browser itself. Other extensions provide more sophisticated functionality, such as NOSCRIPT that provides fine-grained control over page JavaScript execution,[15] or GREASE-MONKEY that provides a full-blown programming environment for scripting browser behavior.[3] These are just a few of the thousands of extensions currently available for Firefox, the second most popular browser today.

Extensions written with benign intent can have subtle security-related bugs, called vulnerabilities, that expose users to devastating attacks from the Web, often just by viewing a Web page. Firefox extensions run with full browser privileges, so attackers can exploit extension weaknesses to take over the browser, steal cookies or protected passwords, compromise confidential information, or even hijack the host system, without revealing their actions to the user. Unfortunately, dozens of extension vulnerabilities have been discovered in the last few years, and capable attacks against buggy extensions have already been demonstrated.[11]

In this paper, we propose VEX, a system for finding vulnerabilities in browser extensions using static information-flow analysis. Our key insight is that *extension vulnerabilities often translate into explicit information flows from injectable sources to executable sinks*. For extensions written with benign intent, most attacks involve the attacker injecting JavaScript into a data item that is subsequently executed by the extension under full browser privileges. We identify key flows of this nature that can lead to security vulnerabilities, and we check extensions for the presence of such flows using a high-precision static analysis that is both path-sensitive and context-sensitive, to minimize the number of false positive suspect flows. VEX has special features to handle the quirks of JavaScript (e.g., VEX does a constant string analysis for expressions that flow into the `eval` statement that execute dynamically generated code).

Determining whether extensions are malicious or harbor security vulnerabilities is a hard problem. Extensions are typically complex artifacts that interact with the browser in subtle and hard to understand ways. For example, the ADBLOCK PLUS extension performs the seemingly simple task of filtering out ads based on a list of ad servers. However, the ADBLOCK PLUS implementation consists of over 11K lines of JavaScript code. Similarly, the NOSCRIPT extension provides fine-grained control over which domains are allowed to execute JavaScript and basic cross-site scripting protection. The NOSCRIPT extension implementation consists of over 19K lines of JavaScript code. Also, ADBLOCK PLUS had 41 releases in January 1, 2006 to October 6, 2011, and NOSCRIPT had 48 releases just in January 1, 2011 to October 6, 2011. While Mozilla uses volunteers to vet each new extension and revision before posting it on their official list of approved Firefox extensions, examining an extension to find a vulnerability requires a detailed understanding of the code to reason about anything beyond the most basic type of information flow. Thus tools to help vet browser extensions can be very useful for improving the security of extensions.

We show that VEX identifies five previously known vulnerabilities, and identifies other flows that led to the discovery of seven previously unknown vulnerabilities, including vulnerabilities in the extensions WIKIPEDIA TOOLBAR, MOUSE GESTURES, and KAIZOU.

2. THREAT MODEL, ASSUMPTIONS, AND USAGE MODEL

In this article, we focus on finding security vulnerabilities in buggy browser extensions. We do not try to identify malicious extensions, bugs in the browser itself, or bugs in other browser extensibility mechanisms, such as plug-ins. We assume that the developer is neither malicious nor trying to

A previous version of this paper was published in the USENIX Security Symposium, Aug. 2010.

obfuscate extension functionality, but we assume the developer could write incorrect code that contains vulnerabilities.

We use two attack models. First, we consider attacks that originate from Web sites, and we assume the attacker can send arbitrary HTML and JavaScript to the user's browser, modeling the usage model that assumes the user can navigate to any page on the internet. We focus on attacks where this untrusted data can lead to code injection or privilege escalation through buggy extensions. In the second attack model, we assume the same model as above, but we consider certain Web sites as trusted. For example, if an extension gleans information from the Facebook Web site, we assume that the Facebook data will *not* include arbitrary HTML and JavaScript, but only well formatted and trusted data.

According to the Mozilla developer site, Mozilla has a team of volunteers who help vet extensions manually. They run new and updated extensions isolated in a virtual machine to test the user experience. The editors also use a validation tool, which uses *grep* to look for key indicators of bugs. Many of the patterns they search for involve interactions between extensions and Web pages, and they use their understanding of these patterns to help guide their inspection of the code. Our goal is to help automate this process, so that analysts can quickly hone in on particular snippets of code that are likely to contain security vulnerabilities. Figure 1 shows our overall work flow for using VEX: when extensions are subject to analysis by VEX, it reports precise code paths from untrusted sources to executable sinks in the extensions' code, which an expert must manually examine to check whether they can be used to mount an attack.

3. VEX INFORMATION FLOW PATTERNS

Firefox has two privilege levels: *page* for the Web page displayed in the browser's content pane, and *chrome* for elements belonging to Firefox and its extensions. Page privileges are more restrictive than chrome privileges. For example, a page loaded from `illinois.edu` can only access content from `illinois.edu`. Firefox code and extensions run with full chrome privileges, which enable them to access all browser states and events, OS resources like the file system and network, and all Web pages. Extensions also can include their own user-interface components via a *chrome document*, which can run with full chrome privileges.

Firefox has APIs for extension code to communicate across protection domains and these interactions are one cause of extension security vulnerabilities. As the Mozilla developer site explains, "One of the most common security issues with extensions is execution of remote code in privileged context. A typical example is an RSS reader extension that would take the content of the RSS feed (HTML code), format it nicely and insert into the extension window. The issue that is commonly overlooked here is that the RSS feed could contain some malicious JavaScript code and it would then execute with the privileges of the extension—meaning that it would get full access to the browser (cookies, history, etc.) and to user's files" [sic].

We characterize these cross-protection-domain interactions as information-flow patterns from JavaScript objects that include page content (untrusted sources) to JavaScript objects and methods that execute content with chrome privileges (executable sinks). In this section we discuss the sources and sinks that VEX tracks. Flows between these sources and sinks are sometimes benign, and represent an incomplete list of possible extension security bugs, but these are the patterns that VEX considers suspicious.

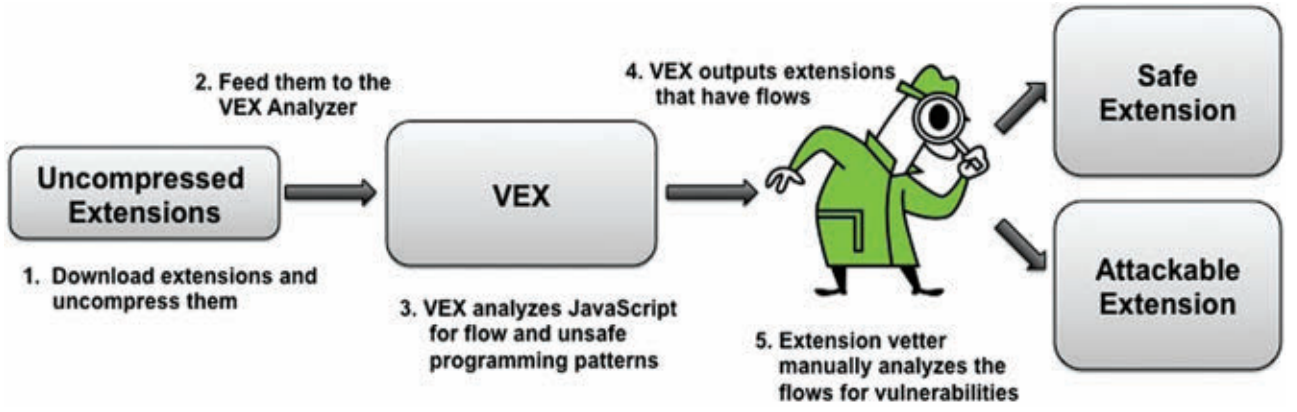
3.1. Untrusted sources

We now describe the untrusted JavaScript objects that extensions can access. Untrusted objects might contain foreign scripts that can lead to attacks if run with chrome privileges.

The JavaScript content-document object (`window.content.document`) accesses the browser's content page directly, and hence is an untrusted source. Also, the browser sets JavaScript pop-up nodes (`document.popupNode`) when the user right-clicks on document object model (DOM) elements. If this DOM element is part of the page content, then it includes untrusted page content.

One API that extensions use to access persistent state is the Resource Description Framework (RDF). RDF is a model for describing hierarchical relationships between browser resources[17] and is used by the browser to store persistent data, like bookmarks. Extension developers can store persistent extension data in an RDF file, or access browser resources stored in RDF format. However, RDF data can

Figure 1. The overall analysis process of VEX.



come from untrusted sources. For example, when a user stores a bookmark, Firefox records the un-sanitized title of the bookmarked page, which is controlled by the Web page, in an RDF file. Extensions can also access un-sanitized bookmark URLs using the `nsILivemarkService` interface and the `BookmarksUtils` object.

Extensions access Firefox preferences through the `nsIPrefService` interface. Any extension can set values in the preferences, and extensions have unchecked access to all preference settings. Some extensions use this service to store untrusted strings obtained from Web page content; hence using this service is also treated as an untrusted source.

In summary, the VEX treats the following as untrusted source objects: `window.content.document`, `document.popupNode`, `BookmarksUtils`, and access to the new instances of the objects `nsIRDFService`, `nsILivemarkService`, and `nsIPrefService`.

3.2. Executable sinks

Now we describe the set of executable sinks, which are JavaScript objects and methods that provide a way to parse and execute JavaScript dynamically. VEX considers these executable sinks to be potentially dangerous when they execute untrusted JavaScript code with chrome privileges.

The `eval` function call interprets string data as JavaScript, which it executes dynamically. This flexible mechanism can be used to generate JavaScript code dynamically, for example to deserialize JavaScript Object Notation (JSON) objects. However, this flexibility can lead to code injection vulnerabilities in extensions. If extensions execute `eval` functions on unsanitized strings that come from untrusted sources, an attacker can inject JavaScript code that runs with full chrome privileges.

Each HTML element in a page has an `innerHTML` property that defines the text that occurs between that element's opening and closing tags. Extensions can change the `innerHTML` property to alter existing DOM elements, or to add new DOM elements, because the browser parses the modified text after JavaScript code modifies this property. Thus, passing specially crafted strings (e.g., <img> tags with JavaScript in their `onload` attribute) into `innerHTML` can lead to code injection attacks.

Extensions can add a new DOM element to a content page or chrome page by using the `appendChild` method. This method causes the browser to parse and process the data within the element, similar to the `innerHTML` property. Therefore, this feature can also be used to execute injected code.

In summary, the executable sinks that we consider in VEX are calls to the functions `eval` and `appendChild`, and assignments to `innerHTML` property.

4. STATIC INFORMATION FLOW ANALYSIS

The core component of VEX is a static analysis tool for detecting explicit information flows in browser extensions written in JavaScript. VEX computes flows between different sources and sinks, including all those described in Section 3. To support fine-grained information-flow analysis, VEX tracks the flows from source objects to the sinks encountered in the JavaScript extension, using a taint-based analysis. Motivated by the fact that every flow reported needs to be checked manually for attacks, which can take considerable human effort, we aim for an analysis that admits as few false positives as possible, where false positives are flows reported by VEX that cannot actually occur at run time.

Statically analyzing JavaScript extensions for flows is a nontrivial task. Object properties in JavaScript change dynamically, in the sense that new object properties can be created dynamically at run time. Functions are objects in JavaScript, and hence can be created, redefined dynamically, and passed as parameters. In addition to the objects defined in the program, the extensions can also access the browser's DOM API and the Firefox Extension API provided by XPCOM components, and the static analysis must handle them correctly. JavaScript browser extensions also have a large number of objects and functions that need to be tracked. The challenge is to accurately keep track of such objects, properties, and the corresponding flows to them.

The analysis engine in VEX is a static taint analysis to detect explicit flows, where taint propagation for JavaScript is achieved by adapting an operational semantics for JavaScript proposed by Maffeis et al.[13] In the analysis, we replace concrete heaps by *abstract heaps*, where abstract heaps accurately track objects and their properties, but abstract the primitive values stored. An abstract heap can be seen as a directed graph, where every object and function in the JavaScript program is represented as a node, while the edges in the heap represent the field relationships between different objects. Additionally, every node in the abstract heap is associated with a taint value, which is used by VEX's analysis to compute the information flows from the source objects to the sinks.

In the analysis, VEX handles only loop-free programs, and translates programs with loops to loop-free programs first by unrolling loops a bounded number of times (hence the analysis is not *sound—see* Section 4.3). The VEX abstract semantics computes and tracks the abstract heap on (loop-free) programs fairly precisely by mimicking the operational semantics for JavaScript. Unlike common abstraction domains used in the literature, at any point during the analysis, an abstract heap does not have a single node representing two objects; hence VEX is quite accurate in keeping track of the precise heap nodes and field relations and the corresponding flows, ignoring only the exact primitive values in the heap (like integers). Since programs are unrolled into loop-free code, the abstract heaps have a bounded size, leading to a terminating algorithm.

4.1. Abstract semantics of JavaScript

In this section, the abstract heap is described in detail, followed by a description of the data structures used for the static analysis. The high-level ideas behind VEX's static analysis are also described.

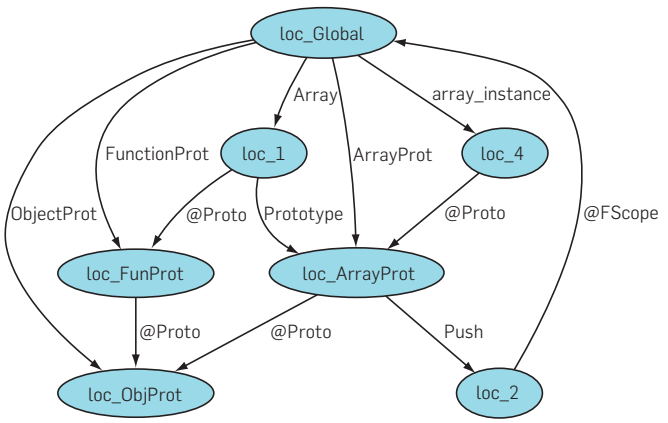
Abstract heaps: We model the state of a JavaScript program using the notion of an abstract heap. Every object is stored on the heap. The heap is modeled as a set of (location, object) pairs. A location is an arbitrarily generated name created whenever a new object is created in the program. An object is a set of (property name, value) pairs. The property names could either be identifiers or strings. An abstract value could

be a heap location (if the property points to another object), a function declaration, a security type, or a primitive value. Security types keep track of taints; a sink object's security type acquires a taint associated with a source object, if there is an explicit flow from the source object to the sink object. A security type is modeled as a pair (taint value, source string); the taint value could either be LOW or HIGH and the taint source is a string identifying the source object of the taint. The primitive string values are preserved and propagated through string operations, whenever they evaluate to constant strings. All other primitive values are abstracted.

Figure 2 gives an example of a sample JavaScript heap computed using the VEX analysis. Every object and function in the JavaScript program is represented as a node in the heap, while the properties of the object are represented using edges in the graph. In the figure, the global object `loc_Global` has five properties `ObjectProt`, `FunctionProt`, `Array`, `ArrayProt`, and `array_instance` pointing to the nodes `loc_ObjProt`, `loc_FunProt`, `loc_1`, `loc_ArrayProt`, and `loc_4` respectively. Every node in the heap is associated with a taint value, HIGH or LOW—HIGH representing the untrusted objects and LOW representing the trusted objects. High taints and low taints are represented by red and blue nodes, respectively, in the figures (all nodes in Figure 2 are LOW). Figure 3 shows the *initial* abstract heap representation of the `window.content.document` object and the `window.document` object; notice that one of the nodes `loc_document` has a high taint.

The analysis: VEX analysis is based on a set of rules that transform abstract heaps according to each statement in the program, and it works by essentially over-approximating the effect of the statements on the abstract heap. These rules closely follow the small-step operational semantics proposed by Maffeis et al.,[13] which covers the ECMA-262 standard for JavaScript. JavaScript core objects and functions are summarized to have only the essential functionality; an example summary is given in Section 4.2. Variables and functions that are not initialized in the current program execution or through summaries, are initialized to point to placeholder dummy objects with HIGH taints. The default taint of an object created in the extension is set to LOW unless the analysis explicitly sets the value to HIGH or a variable is uninitialized.

Figure 2. Sample JavaScript heap—Array object.



The loops in the program are unrolled a bounded number of times and function calls are inlined for a bounded unrolling of recursive calls, and every path of the resulting program is explored. Thus VEX may overlook certain flows, as discussed in Section 4.3. The static analysis does not evaluate the conditions in conditional statements of the program because of the abstraction. Whenever it reaches a conditional statement, both branches are traversed, in a depth-first manner, to ensure that the entire program is covered. The analysis is flow-sensitive and, due to inlining, also context-sensitive.

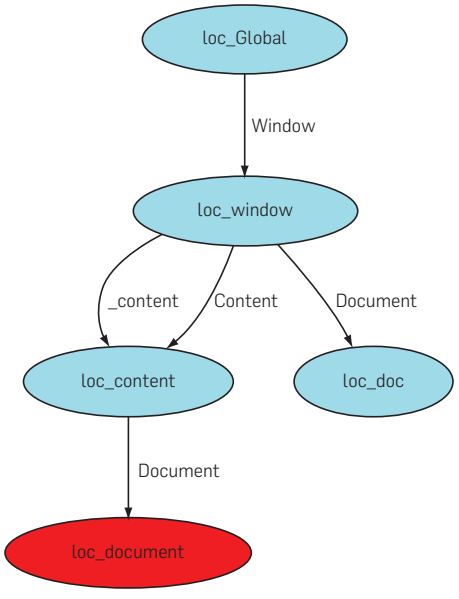
Prototypes: JavaScript uses prototype-based inheritance.[9] Every object in the JavaScript heap has a special `@Proto` property, which is used to specify inheritance chains. Additionally, every function (that can be used as a constructor in `new`) has a `prototype` property. This `prototype` property is used to instantiate the `@Proto` property when a new object is created using the function constructor. An object inherits all the properties of its `@Proto` and of all the objects in the prototype's `@Proto` chain.

Figure 2 illustrates how VEX handles prototype-based inheritance. The `Array` object in JavaScript is represented as the node `loc_1` in the figure. Since the `Array` object is a constructor, which can be used to create new instances of the array, it has a `prototype` field pointing to the object, `ArrayProt`, represented in the graph by the node `loc_ArrayProt`. A new `Array` instance, `array_instance` object, is created in the program using the statement: `array_instance = new Array ()`. In Figure 2, `loc_4` represents the `array_instance` object. The `@Proto` field of this object points to the object `loc_ArrayProt`. Therefore, the push method is accessible to the `array_instance` object and can be called using the `array_instance.push`.

4.2. Handling other features of JavaScript

Function and object summaries: Natively supported functions and objects are replaced with stubs that summarize

Figure 3. window.content.document object.



the effect on the heap and the taints when accessing them. VEX function and object summaries are hence simplified JavaScript objects and functions containing only the essential functionality of the objects. For example, a JavaScript `Array` object is defined in Figure 4 to be a function object with the `@Class`, `prototype`, and `@Proto` properties initialized to the string "Function", identifier `ArrayProt`, and identifier `FunctionProt`, respectively. The variables `FunctionProt` and `ArrayProt` point to the prototype objects, which contain the various functions like `length` and `push`.

Browser's DOM API and XPCOM components: VEX treats most of the browser's DOM API, and XPCOM components as uninitialized variables, fields, and functions. However, VEX provides explicit function summaries for the API components and objects that VEX needs to keep track of in order to trace the flows to and from the objects. VEX analysis sets the taint of the objects that represent insecure sources or those that are dependent on insecure sources to HIGH.

Higher-order functions: VEX analysis accurately keeps track of the objects and implements function calls by inlining the function bodies according to the JavaScript semantics. Higher-order functions calls are also inlined. Additionally, VEX provides summaries for some higher-order functions in the JavaScript API. For example, the `settimeout` function in JavaScript takes a callback function as its first argument. This function is represented in VEX as a function in which the function body invokes the callback function in the first argument.

Dynamically generated code: The `eval` method in JavaScript allows execution of dynamically formed code, and is widely used in browser extensions. While an accurate analysis of the structure of dynamically created code is a research topic in itself, and quite out of the scope of this paper, the analysis cannot simply ignore `eval` statements. VEX analysis performs a *constant-string analysis* for strings and string operations. If the actual parameters to the `eval` statement evaluate to a constant string, VEX's static analysis engine parses these constant strings and inserts them into the program flow just after the `eval` statement. This ensures that these newly parsed statements are included in the computation of the taint. In most correct extensions, an `eval`-ed statement is dynamically chosen from a set of constant-strings or taken from trusted sources, and hence evaluate to a constant string on the path explored (and tracked accurately by VEX). Parameters to `eval`, whose exact string values are not statically inferred by VEX along the path explored, are tested to check if they are tainted. If there is a flow from an *untrusted* source to an `eval`, VEX will report this flow, as it corresponds to a vulnerable flow pattern.

Object properties accessed in the form of associative arrays: In JavaScript, objects are treated as associative arrays. This means that any property of the object can be accessed using the array notation. Array indices could be constant strings, which are then evaluated to get the actual property being accessed; or they could be numbers, which indicate the property number that is being accessed; or they could be variables, that could be instantiated at run time. If VEX cannot evaluate the array index to a property name for any reason, the array access conservatively gets the taints of every property in the parent array object.

Functions that take arbitrary number of arguments: Some functions in JavaScript can have variable numbers of arguments. For example, the `push` method of an array can be called with any number of arguments and the arguments will be appended to the end of that array. To handle this in VEX, the object representing the `push` method has a special property indicating that it can take a variable number of arguments and when the method is called, VEX analysis conservatively appends the taints of all the arguments to the `push` method to the array object on which the method is called.

Figure 4. `Array` object summary in VEX.

```
varArray=function(){
  this.@Class="Function";
  this.prototype=ArrayProt;
  this.@Proto=FunctionProt;};
```

4.3. A note on soundness

Most static analysis tools, such as those used in compilers and those used in abstract interpretation, *over-approximate* the concrete semantics, and hence are *sound*. In the context of flow analysis, a sound tool never reports that a program has no flows when it has one. Soundness often entails a large number of *false positives*, i.e., flows that are reported by the tool but may not actually ever happen during execution.

VEX is not sound. We believe that a sound state-of-the-art analysis tool for JavaScript extensions would overwhelm and frustrate the tool's users with a torrent of false positives. Thus to handle certain features of JavaScript without producing excessively many false positives, we chose not to make VEX sound. As a consequence, for example, a maliciously written extension could quite easily evade detection by VEX. On the other hand, a maliciously written extension can easily harm its users directly, without any input from untrusted Web pages. This underlies the reason why our threat model assumes that the extension author is not malicious.

Instead of aiming for soundness, we concentrated on making VEX fairly accurate on paths in the program, without *collapsing* (merging) the nodes of the heap in any way. However, since VEX can only analyze a finite number of paths in the program (obtained by unrolling recursion a bounded number of times) in this accurate manner, the analysis VEX performs is inherently not sound.

False positives are also, of course, still possible in VEX, i.e., VEX may report flows that actually do not exist in the program. This stems from the fact that the analysis uses an abstraction. In particular, not having precise enough information for evaluating conditionals, not precisely being able to determine the values of strings being subject to `eval` statements, etc. are common sources for false positives. Compared to classical heap analysis in programs that merges nodes in heaps, VEX performs a much more accurate analysis that reduces the number of false positives considerably. In experiments, we found that VEX produces very few false positives.

Overall, our choices were determined mainly by the complexity of JavaScript analysis and our aim at building a *useful* tool, which in turn led us to sacrifice soundness.

5. EVALUATION

Vex is implemented in Java (~7000 LOC), and utilizes a JavaScript parser built using the ANTLR parser generator for the JavaScript 1.5 grammar provided by ANTLR.[1] ANTLR outputs Java-based Abstract Syntax Trees (AST) for JavaScript sources obtained from the pre-processing of the extension's XUL and JavaScript files. The XUL files add different UI elements to the browser's chrome. When any one of the user-interface elements is invoked and clicked, the corresponding event is triggered and the event-handler is called. We extract all such calls to the event-handlers from the XUL files and run them using Vex's abstract operational semantics.

During the execution of the program using the abstract operational semantics outlined in Section 4, if the program reaches a vulnerable sink, it checks if the inputs or assignments to the sink are tainted. If they are tainted, Vex reports the occurrence of the flow along with the source objects and sink locations in the code. The source objects are the objects described in Section 3 and the sink locations are the points where the sinks described in Section 3 are encountered during the execution. The rest of this section summarizes our results.

The number of loop unrollings can be set as a parameter in the Vex analysis engine (in our experiments, a bound of just *one* was used). The Vex implementation has a number of optimizations to improve memory usage and speed. To save memory, abstract heaps are freed when backtracking in the depth-first search. But to save time, abstract heaps at join points are cached and compared when other paths hit these points, to avoid exploring paths unnecessarily.

5.1. Evaluation methodology

The extensions we analyzed were chosen as follows. First, in October 2008, we built a suite of extensions using a random sample of 1827 extensions from the Mozilla add-ons Web site, by downloading the first extensions in alphabetical order for all subject categories. This extension suite had two extensions with known vulnerabilities. In November 2009, we downloaded 699 of the most popular extensions and 8 extensions with known vulnerabilities. The random sample and the popular extensions had 74 extensions in common, for a total of 2460 extensions. Our suite includes *multiple versions* of some extensions, allowing cross-version comparisons. For instance, we found a new version of the Fizzle (see Bandhakavi et al.[2]), to be vulnerable even though its authors tried to fix the vulnerabilities in the previous version.

We extracted the JavaScript files from these extensions and ran Vex on them, using a 2.4 GHz 64 bit × 86 processor with a maximum heap size of 16GB for the JVM.

To evaluate the effectiveness of Vex, we perform two kinds of experiments. First, we run Vex on the downloaded extensions and check if any of them have one of the malicious flow patterns. Second, we check if Vex can detect known extension vulnerabilities.

5.2. Experimental results

Finding flows from injectible sources to executable sinks: Figure 5 summarizes the experimental results for flows that are from injectible sources to executable sinks (flows for which the sinks are `eval` and `innerHTML`). Of the 2460 extensions analyzed by Vex, a `grep` showed that a total of 977 extensions had the occurrence of either the string "`eval`" or the string "`innerHTML`" or both.

The first column of Figure 5 indicates the exact source to sink flow pattern checked by Vex. The second column indicates the number of extensions on which Vex reports an alert with corresponding flows. On an average, Vex took 11.5 s per extension. It took about a week to analyze all the extensions with flows from untrusted sources to `eval` and `innerHTML` sinks.

To look for potential attacks, we manually analyzed the extensions with suspect flows found by Vex, spending about 20 min per extension on average. The next column reports the number of extensions on which we could engineer an attack based on the flows reported by Vex. We were able to attack nine extensions, of which only two extensions (Fizzle version 0.5 and Beatnik v-1.0) were already known to be vulnerable. The rest of the attacks are new.

The next column shows the extensions where the source is provided either by the extension user or the extension developer or computed from the system parameters by the

Figure 5. Flows from injectible sources to executable sinks.

Flow Pattern	Vex Alerts	Attackable			Not Attackable		
		Confirmed	*Source from user/ extension/system*	*Source is a trusted Web site*	*Sanitized input*	*Non-chrome sinks*	*Non-existent flows*
Content Doc to `eval`	13	2*	8	0	0	0	3
Prefs to `eval`	10	1*	7	2	0	0	0
Unknown var to `eval`	28	0	12	2	3	0	11
Content Doc to `innerHTML`	22	1*	0	3	2	14	2
RDF to `innerHTML`	7	5*	1	1	0	0	0
Prefs to `innerHTML`	6	0	6	0	0	0	0
`popupNode` to `innerHTML`	2	0	0	0	1	1	0
Unknown to `innerHTML`	27	0	11	7	3	5	1
Total	115	9	45	15	9	20	17

* Attackable Extensions are listed in Section 5.2

extension. The values are either stored in the preferences or in a local file. Since we trust the users and extension developers in our trust model, these extensions are considered to be non-vulnerable. However, if the preferences file or the local file system is corrupted in any way, these extensions can be attacked.

The fifth column shows the extensions where the source is code from a Web site, and where an attack *is possible* provided the Web site can be attacked. In other words, these extensions rely on a *trusted Web site* assumption (e.g., that the code on the Facebook Web site is safe). We think that these are valid warnings that users of an extension (and Mozilla) should be aware of; trusted Web sites can after all be compromised, and the code on these sites can be changed leading to an attack on all users of such an extension.

Not all flows lead to attacks—the next set of columns describe the alerts that we were unable to convert to concrete attacks. Some extensions were not exploitable as the input is *sanitized* correctly (either by the extension or the browser), preventing JavaScript injection. Other extensions were not exploitable as the sinks were not in chrome executable contexts. These extensions are noted in the next two columns. Finally, Vex, being a static flow-analysis tool, does report alerts about flows that do not actually exist—there were very few of these, and are noted under the column "Nonexistent flows." Section 5.4 discusses the flows that do not lead to attacks.

New vulnerabilities discovered: The number of security vulnerabilities discovered is shown in column 3 in Figure 5, of which 7 are new. Wikipedia Toolbar versions V-0.5.7 and V-0.5.9 have flows from `window.content.document` to `eval`, which leads to attacks. Mouse Gestures Redox v-2.0.3 has flows from `nsIPrefService` to `eval`, which also led to an attack. Beatnik V-1.2, Fizzle v-0.5.1, and Fizzle v-0.5.2 are also attackable, and have flows from `nsIRDFService` to `innerHTML`. Kaizou v-0.5.8 has a flow from `window.content.document` to `innerHTML` which leads to attacks. Section 5.3 gives some details about the flows and the attacks in some of the vulnerable extensions. Details about Fizzle (and Beatnik) vulnerabilities can be found in the previous version of this article.[2]

Known vulnerabilities detected: Apart from the new vulnerabilities found by Vex, there are several extensions that have been reported to be vulnerable in the past. In the course of our research, we found 18 unique extensions that were reported to be vulnerable in various databases like CVE, Secunia, etc. Of these 18, we did not find the source code for 5 extensions (Greasemonkey $v \leq 0.3.5$, Wizz Rss $v < 3.1.0.0$, Skype $v \leq 3.8.0.188$, MouseoverDictionary $v < 0.6.2$, POW $v < 0.0.9$), so we did not analyze them. Of the remaining 13 extensions, we found that 10 of them can potentially be found using explicit information flow analysis techniques, like Vex.

Currently, Vex can detect 5 of the above 10 known extension that have flow-based vulnerabilities: Fizzle v-0.5, Beatnik V-1.0, CoolPreviews v-2.7, 2.7.2, infoRSS V-<=1.1.4.2, and Sage v- < 1.3.9, <=1.4.3. CoolPreviews has flows from `document.popupNode` to `appendChild`. infoRSS has flows from `nsIRDFService` to `appendChild`. Sage has flows from `BookmarksUtils` to an object accessing the local file system using the `nsIFile` interface.

The remaining five extensions have flow vulnerabilities but were not found by Vex for the following reasons. For FeedSidebar V< 3.2, FireBug v-1.01, Scribefire V<= 3.4.2, and Update Scanner V< 3.0.3 the trigger of the flow was in an event handler or a function call which was called outside the extension's code base. In Yoono version $\leq$ 6.1.1 an un-sanitized JavaScript element like an image or link is rendered in the chrome context. However, it was difficult to find the source and sink objects from its source code.

Finally, there were three extension vulnerabilities (for which we had the source) that cannot be found by Vex because they are not flow vulnerabilities. These vulnerabilities include attacks on a file server (e.g., FireFTP V < 0.97.2, < 1.04), and directory traversal attacks (e.g., Navigational Sounds version-1.0.2, Ajax Yahoo Mail Viamatic Webmail version-0.9) when a chrome package is "flat" rather than contained in a jar. In both the above cases, an attacker can escape from the extension's directory and read files in a predictable location on the disk. Since such attacks are not related to chrome privilege escalations, and Vex does not handle them.

5.3. Successful attacks

Attack scripts: All our attack scenarios involve a user who has installed a vulnerable extension who visits a malicious page, and either automatically or through invoking the extension, triggers script written on the malicious page to execute in the chrome context. Figure 6 illustrates an attack payload that can be used in such attacks: this script displays the folders and files in the root directory.

The attack payloads could be much more dangerous, where the attacker could gain complete control of the affected computer using XPCOM API functions. More examples of such payloads are enumerated in the white-paper given in Freeman and Liverani[7] In this section, we illustrate a few

Figure 6. Attack script to display directories.

```
<script>
var root = Components.classes
["@mozilla.org/file/local;1"].createInstance
(Components.interfaces.nsILocalFile);
try {
root.initWithPath("/."); // for Linux or Mac
} catch (er){
root.initWithPath("\\\\."); // for Windows
}
var drivesEnum = root.directoryEntries, drives
= [];
while (drivesEnum.hasMoreElements()) {
drives.push(drivesEnum.getNext().
QueryInterface(Components.interfaces.
nsILocalFile).path);
}
alert(drives);
</script>
```

Figure 7. Wikipedia toolbar code.

```
script = window._content.document.
getElementsByTagName("script")[0].innerHTML;
eval(script);
```

attacks on extensions with previously unknown vulnerabilities.

Wikipedia Toolbar, up to version 0.5.9: If a user visits a Web page with the directory display attack script in its <head> tag, and clicks on one of the Wikipedia toolbar buttons (unwatch, purge, etc.), the script executes in the chrome context. The attack works because the extension has the code given in Figure 7 in its toolbar.js file.

The first line gets the first <script> element from the Web page and executes it using `eval`. The extension developer assumes the user only clicks the buttons when a Wikipedia page is open, in which case <script> may not be malicious. But the user might be fooled by a malicious Wikipedia spoof page, or accidentally press the button on some other page. VEX led us to this previously unknown attack, which we reported to the developers, who acknowledged it, patched it, and released a new version. This resulted in a new **CVE vulnerability (CVE-2009-41-27).** The fix involved inserting a conditional in the program to check if the URL of the page is in Wikipedia's domain and evaluating the script only if this is true.

Kaizou v-0.5.8: Kaizou is a Web development extension that allows users to open the source of any Web page in a separate window, modify the contents and render it again in the current window by pressing a button. However, this separate window has chrome privileges, and when the user saves the changes he made to the page source, the scripts in the page are executed with chrome privileges. A malicious Web page can have an attack script, which could result in an attack when modified using KAIZOU.

Mouse Gestures Redox v-2.0.3: The MOUSE GESTURES REDOX extension allows users to create shortcuts for frequently used commands without using keyboard, menu, or toolbars. The users can either create new gestures or download them from an online source. The new gestures are scripts, which are stored in the browser's preferences file. When the gestures are enabled, they are retrieved from the prefs.js file and sent as arguments to the `eval()` function, thereby activating the gestures. If any of the gestures downloaded from the internet contain attack scripts, they would be executed in the chrome context when `eval` is called.

5.4. Flows that do not result in attacks

Figure 8 gives several examples of the suspect flows that we manually analyzed and for which either trusted sources were assumed by the extension or we could not find attacks.

The first set has extensions accessing values from Web sites or sources it trusts, and the values flow to `eval` or `innerHTML`. Of course, if the trusted sources are compromised, then the extensions may become vulnerable. The second set illustrates examples where the input was sanitized between the source and the sink. We do not know for sure that the sanitization is adequate, but we were unable to attack it. The third set of extensions had non-chrome sinks. The last set has two examples that show false positives where the flows reported by VEX do not exist in the code.

6. RELATED WORK

Maffeis et al.[13] proposed a small-step operational semantics for JavaScript, using which they analyze security properties of Web applications. They also use their operational semantics for generating safe subsets of JavaScript and to manually prove that the so-called safe subsets of JavaScript are in fact vulnerable to certain attacks.[14] Our operational semantics follows their operational semantics, but works on an abstract heap. Guha et al.[9] propose an alternate operational semantics.

Louw et al.[12] highlight some of the potential security risks posed by browser extensions, and propose run time support for restricting the interactions between browsers and extensions. Our analysis technique is complementary to their

Figure 8. Extensions that could not be attacked.

Classification	Extension	Flow pattern	Explanation
Source is trusted Web site	WORLD SMS V-2.2	Unknown var to `eval`	The source is a Web site: http://worldsms.co.cc/json?get=info, which is hardcoded into the extension code.
	BROWSER BACKGROUNDS V-0.3.5	`nsIRDFService` to `innerHTML`	The user installs background images from the Web site http://browserbackgrounds.com/
Sanitized input	ALPHANUMERATOR V-0.2	Content Doc to `innerHTML`	The input string is converted to numbers effectively sanitizing it.
	VIEW SOURCE CHART V-2.7	Content Doc to `innerHTML`	Input HTML tags are sanitized into custom tags
Non-chrome sinks	PAGEDIFF V-1.3.0	Content Doc to `innerHTML`	The display page has a chrome url but is marked to be "content-type."
Non-existent flows	LINK_ALERT V-0.8.2.1	Unknown var to `eval`	`eval`'s argument is a packed and minified JavaScript string that VEX could not parse properly. VEX finds an unknown variable in this incorrectly parsed string.
	SKIPSCREEN V-0.1.09102009	Unknown var to `innerHTML`	During the execution, the extension will never follow the branch that has the sink, as the conditional variable is never initialized in the program.

restrictions since even restricted interfaces can still be susceptible to security vulnerabilities.

More recently, researchers have developed static information flow analysis methods for JavaScript.[4, 8] In Chugh et al.[4] the authors essentially perform a *context-insensitive* and *flow-insensitive* static analysis on the code, and delegate analysis of dynamic code to runtime checks. Guarnieri and Livshits[8] propose a mostly static enforcement for JavaScript analysis, which is *context-sensitive* but *flow-insensitive*. In contrast, our analysis is both *flow-sensitive* and *context-sensitive*, thereby reducing the number of false positives.

Several *dynamic* analysis techniques with static instrumentation have been proposed for JavaScript to check information-flow properties.[10, 18] *SABRE*[5] is a framework for dynamically tracking in-browser information flows for analyzing JavaScript-based browser extensions. The taints are tracked by modifying the JavaScript interpreter. In contrast, Djeric and Goel[6] dynamically track taints in both the browser's native code and the script interpreter. Although dynamic techniques are useful in preventing certain types of script injection attacks if they are enforced by the Web browser, they suffer from a few drawbacks. When a questionable flow is detected dynamically, the browser has to either choose an appropriate action (which might be overly restrictive) or ask the user to choose an action (which might lead to an attack if the user chooses a wrong option). Additionally, dynamic techniques impose a performance and memory overhead on the browser because of the need to keep track of the security label for every JavaScript object inside the browser. One of our main motivations was to facilitate a static analysis that scales to thousands of extensions, to circumvent these problems.

7. CONCLUSION

We have presented Vex, a tool for detecting potential security vulnerabilities in browser extensions using static analysis. Vex helps in automating the difficult manual process of analyzing browser extensions, by identifying and reasoning about subtle and potentially malicious flows. Experiments on thousands of extensions indicate that Vex is successful at identifying flows that indicate potential vulnerabilities and greatly reducing the number of flows that must be vetted manually. Using Vex, we identified seven previously unknown security vulnerabilities and five known vulnerabilities, together with a variety of instances of unsafe programming practices.

An interesting future direction is to develop automatic ways to synthesize attacks that exploit flows reported by Vex. A technique based on *constraint solving* to generate attack inputs that satisfy the path constraints in the flow seems appropriate.

In the broader context, there is an increasing number of settings where small software teams (consisting of even one or two people) write software that is downloaded and used by hundreds of thousands of people. Browser extensions fall in this category, but several others have emerged, including mobile phone applications (for iPhone/Android/Windows) and Facebook applications. The teams writing these software do not always think about security carefully, leaving their users with potential privacy and integrity risks. We believe that precise static analysis tools, such as the one presented in this paper, combined with more precise and adaptable access control policies, can help address this security concern.

Acknowledgments

We thank Chris Grier and Mike Perry who directed us to the Firefox extension vulnerabilities. This research was funded in part by NSF CAREER award #0747041, NSF grant CNS #0917229, NSF grant CNS #0831212, grant N0014-09-1-0743 from the Office of Naval Research, and AFOSR MURI grant FA9550-09-01-0539.

References

1. ANTLR Parser Generator. http://www.antlr. org, 2008.
2. Bandhakavi, S., King, S.T., Madhusudan, P., Winslett, M. VEX: Vetting browser extensions for security vulnerabilities. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security '10 (Berkeley, CA, 2010), USENIX Association, 339–354.
3. Boodman, A. The Greasemonkey Firefox extension. https://addons.mozilla.org/en-US/ firefox/addon/748, 2005.
4. Chugh, R., Meister, J.A., Jhala, R., Lerner, S. Staged information flow for JavaScript. In *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '09 (New York, NY, 2009), ACM, 50–62.
5. Dhawan, M., Ganapathy, V. Analyzing information flow in JavaScript-based browser extensions. In *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09 (Washington, DC, 2009), IEEE Computer Society, 382–391.
6. Djeric, V., Goel, A. Securing script-based extensibility in web browsers. In *Proceedings of the 19th USENIX Conference on Security*, USENIX Security' 10 (Berkeley, CA, 2010), USENIX Association, 355–370.
7. Freeman, N. Liverani, R.S. Exploiting cross context scripting vulnerabilities in Firefox (April 2010). http: //www.security-assessment.com/files/whitepapers/Exploiting_Cross_Context_ Scripting_vulnerabilities_in_Firefox.pdf
8. Guarnieri, S. Livshits, B. GATEKEEPER: Mostly static enforcement of security and reliability policies for javascript code. In *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM '09 (Berkeley, CA, 2009), USENIX Association, 151–168.
9. Guha, A., Saftoiu, C., Krishnamurthi, S. The essence of JavaScript. In *ECOOP, Lecture Notes in Computer Science*. Springer, 2010.
10. Kikuchi, H., Yu, D., Chander, A., Inamura, H., Serikov, I. JavaScript instrumentation in practice. In Ramalingam *Programming Languages and Systems, Proceedings of the 6th Asian Symposium, APLAS 2008* (Bangalore, India, December 9–11, 2008), volume 5356 of *Lecture Notes in Computer Science*. Springer, 2008, 326–341.
11. Liverani, R.S., Freeman, N. Abusing Firefox extensions, Defcon (July 17, 2009).
12. Louw, M.T., Lim, J.S., Venkatakrishnan, V.N. Extensible web browser security. In B. M. Hämmerli and R. Sommer, eds., *DIMVA*, volume 4579 of *Lecture Notes in Computer Science*, Springer, 2007, 1–19.
13. Maffeis, S., Mitchell, J.C., Taly, A. An operational semantics for JavaScript. In Ramalingam *Programming Languages and Systems, Proceedings of the 6th Asian Symposium, APLAS 2008* (Bangalore, India, December 9–11, 2008), volume 5356 of *Lecture Notes in Computer Science*. Springer, 2008, 307–325.
14. Maffeis, S. Taly, A. Language-based isolation of untrusted JavaScript. In *Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium* (Washington, DC, 2009), IEEE Computer Society, 77–91.
15. Maone, G. NoScript Firefox extension. http://noscript.net/
16. Ramalingam, G. ed. *Programming Languages and Systems*, In *Proceedings of the 6th Asian Symposium, APLAS 2008* (Bangalore, India, December 9–11, 2008), volume 5356 of *Lecture Notes in Computer Science*. Springer, 2008.
17. Waterson, C. RDF in fifty words or less. https://developer.mozilla.org/en/RDF_in_ Fifty_Words_or_Less (June 9, 2008).
18. Yu, D., Chander, A., Islam, N., Serikov, I. JavaScript instrumentation for browser security. In *Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '07, (New York, NY, 2007), ACM, 237–249.

Sruthi Bandhakavi, (sbandha2@illinois.edu), Department of Computer Science, University of Illinois at Urbana, Champaign.

P. Madhusudan, (madhu@illinois.edu), Department of Computer Science, University of Illinois at Urbana, Champaign.

Samuel T. King, (kingst@illinois.edu), Department of Computer Science, University of Illinois at Urbana, Champaign.

Wyatt Pittman, (wpittma2@illinois.edu), Department of Computer Science, University of Illinois at Urbana, Champaign.

Nandit Tiku, P. (tiku1@illinois.edu), Department of Computer Science, University of Illinois at Urbana, Champaign.

Marianne Winslett, (winslett@illinois.edu), Department of Computer Science, University of Illinois at Urbana, Champaign.

DOI:10.1145/1995376.1995399

Technical Perspective
Abstracting Abstract Machines

By Olivier Danvy and Jan Midtgaard

THE GOAL OF program analysis is to statically predict runtime properties of programs without running them. The semantic approach to program analysis originates in Cousot's path-breaking work on abstract interpretation: start from a formal mathematical model of program execution—a *semantics*—and approximate it with Galois connections (or similar means) into a computable model based on lattices of runtime properties that accounts for all possible execution paths. Each program gives rise to a collection of equations that are then typically solved by fixed-point iteration.

Semantics-based program analysis therefore requires one to (1) start from a "friendly" semantics; design a "congenial" lattice of runtime properties; (3) associate a "relevant" set of equations to a program; and (4) solve these equations efficiently.

Each of these requirements is fraught with difficulties:

1. Among the varieties of formal semantics that exist (operational, denotational, axiomatic, among others) and their sub-varieties (for example, small step or big step), where is your friendly semantics? Ideally, it should lend itself to a good approximation into a computable model.

2. What is a congenial lattice of runtime properties? How wide should it be? How high? Ideally, it should lend itself to a good widening operator that accelerates the convergence of fixed-point iteration without compromising the precision of its result.

3. What is a relevant set of equations? Ideally, each equation should mimic the friendly semantics as closely as possible.

4. What is the best representation of equations and the most efficient way to solve them? This is an algorithmics problem.

Effective answers to each of these questions have been found before, but it is like each of them is a tour de force.

In the following paper, David Van Horn and Matthew Might take a radical bet of simplicity and effectiveness:

▸ Since most semantic artifacts are inter-derivable, without loss of generality, they select abstract machines—deterministic state-transition systems with potentially infinite state spaces—as their friendly semantics.

We find Van Horn and Might's scientific contribution to be an effective tutorial on how to develop a higher-order program analysis by abstracting an abstract machine.

▸ They then refactor each abstract machine into a non-deterministic state-transition system with a finite state space.

Their methodology is concretely useful: it enables program-analysis designers to start from an existing abstract machine rather than from an ad hoc, tailored one, and then factor it uniformly into an abstraction-friendly semantic artifact. Their methodology is effective: it scales to a variety of computational situations involving realistic programming-language constructs, for example, exceptions. Their methodology is structural and generic: it enables program-analysis designers to concentrate on what is specific to their analysis and is still difficult—their lattice of runtime properties, their widening operator, how to represent their equations, and how to solve them efficiently—instead of being forced to perform one global tour de force after another, from scratch, every time.

As such, we find Van Horn and Might's scientific contribution to be a significant stepping stone conceptually and practically as well as an effective tutorial on how to develop a higher-order program analysis by abstracting an abstract machine. We also found their article a pleasure to read. C

Olivier Danvy (danvy@cs.au.dk) is an associate professor and **Jan Midtgaard** (jmi@cs.au.dk) is a post-doctoral researcher in the Department of Computer Science at Aarhus University, Aarhus, Denmark.

DOI:10.1145/1995376.1995400

Abstracting Abstract Machines

A Systematic Approach to Higher-Order Program Analysis

By David Van Horn and Matthew Might

Abstract

Predictive models are fundamental to engineering reliable software systems. However, designing conservative, computable approximations for the behavior of programs (static analyses) remains a difficult and error-prone process for modern high-level programming languages. What analysis designers need is a principled method for navigating the gap between semantics and analytic models: analysis designers need a method that tames the *interaction* of complex languages features such as higher-order functions, recursion, exceptions, continuations, objects and dynamic allocation.

We contribute a *systematic approach to program analysis* that yields novel and transparently sound static analyses. Our approach relies on existing derivational techniques to transform high-level language semantics into low-level deterministic state-transition systems (with potentially infinite state spaces). We then perform a series of simple machine refactorings to obtain a sound, computable approximation, which takes the form of a *non-deterministic* state-transition systems with *finite* state spaces. The approach scales up uniformly to enable program analysis of realistic language features, including higher-order functions, tail calls, conditionals, side effects, exceptions, first-class continuations, and even garbage collection.

1. INTRODUCTION

Software engineering, compiler optimizations, program parallelization, system verification, and security assurance depend on program analysis, a ubiquitous and central theme of programming language research. At the same time, the production of modern software systems employs expressive, higher-order languages such as Java, JavaScript, C#, Python, Ruby, etc., implying a growing need for fast, precise, and scalable higher-order program analyses.

Program analysis aims to soundly predict properties of programs before being run. (*Sound* in program analysis means "conservative approximation": if a sound analysis says a program must not exhibit behavior, then that program will not exhibit that behavior; but if a sound analysis says a program may exhibit a behavior, then it may or may not exhibit that behavior.) For over 30 years, the research community has expended significant effort designing effective analyses for higher-order programs.[13] Past approaches have focused on connecting high-level language semantics such as structured operational semantics, denotational semantics, or reduction semantics to equally high-level but dissimilar analytic models. These models are too often far removed from their programming language counterparts and take the form of constraint languages specified as relations on sets of program fragments.[12, 18, 25] These approaches require significant ingenuity in their design and involve complex constructions and correctness arguments, making it difficult to establish soundness, design algorithms, or grow the language under analysis. Moreover, such analytic models, which focus on "value flow," i.e., determining which syntactic values may show up at which program sites at run-time, have a limited capacity to reason about many low-level intensional properties such as memory management, stack behavior, or trace-based properties of computation. Consequently, higher-order program analysis has had limited impact on large-scale systems, despite the apparent potential for program analysis to aid in the construction of reliable and efficient software.

In this paper, we describe a *systematic approach to program analysis* that overcomes many of these limitations by providing a straightforward derivation process, lowering verification costs and accommodating sophisticated language features and program properties.

Our approach relies on leveraging existing techniques to transform high-level language semantics into *abstract machines*—low-level deterministic state-transition systems with potentially infinite state spaces. Abstract machines,[11] and the paths from semantics to machines,[5, 7, 20] have a long history in the research on programming languages.

From an abstract machine, which represents the idealized core of a realistic run-time system, we perform a series of basic machine refactorings to obtain a *non-deterministic* state-transition system with a *finite* state space. The refactorings are simple: (1) variable bindings and the control stack are redirected through the machine's store and (2) the store is bounded to a finite size. Due to finiteness, store updates must become merges, leading to the possibility of multiple values residing in a single store location. This in turn requires store look-ups be replaced by a non-deterministic choice among the multiple values at a given location. The derived machine computes a sound approximation of the original machine, and thus forms an *abstract interpretation* of the machine and the high-level semantics.

The approach scales up uniformly to enable program analysis of realistic language features, including

The original version of this paper was published in *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*.

higher-order functions, tail calls, conditionals, side effects, exceptions, first-class continuations, and even garbage collection. Thus, we are able to refashion semantic techniques used to model language features into abstract interpretation techniques for reasoning about the behavior of those very same features.

Background and notation: We present a brief introduction to reduction semantics and abstract machines. For background and a more extensive introduction to the concepts, terminology, and notation employed in this paper, we refer the reader to *Semantics Engineering with PLT Redex*.[7]

2. FROM SEMANTICS TO MACHINES AND MACHINES TO ANALYSES

In this section, we demonstrate our systematic approach to analysis by stepping through a derivation from the high-level semantics of a prototypical higher-order programming language to a low-level abstract machine, and from the abstract machine to a sound and computable analytic model that predicts intensional properties of that machine. As a prototypical language, we choose the call-by-value λ-calculus,[19] a core computational model for both functional and object-oriented languages. We choose to model program behavior with a simple operational model given in the form of a reduction semantics. Despite this simplicity, reduction semantics scale to full-fledged programming languages,[22] although the choice is somewhat arbitrary since it is known how to construct abstract machines from a number of semantic paradigms.[5] In subsequent sections, we demonstrate the approach handles richer language features such as control, state, and garbage collection, and we have successfully employed the same method to statically reason about language features such as laziness, exceptions, and stack-inspection, and programming languages such as Java and JavaScript. In all cases, analyses are derived following the systematic approach presented here.

2.1. Reduction semantics

To begin, consider the following language of expressions:

$$e \in Exp = x \mid (ee) \mid (\lambda x.e)$$
$$x \in Var \quad \text{an infinite set of identiers.}$$

The syntax of expressions includes variables, applications, and functions. Values v, for the purposes of this language, include only function terms, $(\lambda x.e)$. We say x is the *formal parameter* of the function $(\lambda x.e)$, and e is its *body*. A *program* is a closed expression, i.e., an expression in which every variable occurs within some function that binds that variable as its formal parameter. Call-by-value *reduction* is characterized by the relation **v**:

$$((\lambda x.e)v) \quad \mathbf{v} \quad [v/x]e,$$

which states that a function applied to a value reduces to the body of the function with every occurrence of the formal parameter replaced by the value. The expression on the left-hand side is a known as a *redex* and the right-hand side is its *contractum*.

Reduction can occur within a context of an *evaluation context,* defined by the following grammar:

$$E = [\,] \mid (Ee) \mid (vE).$$

An evaluation context can be thought of as an expression with a single "hole" in it, which is where a redex may be reduced. It is straightforward to observe that for all programs, either the program is a value, or it decomposes uniquely into an evaluation context and redex, written $E[((\lambda x.e)v)]$. Thus the grammar as given specifies a deterministic reduction strategy, which is formalized as a *standard reduction relation* on programs:

$$E[e] \mapsto_{\mathbf{v}} E[e'], \quad \text{if } e \,\mathbf{v}\, e'.$$

The *evaluation* of a program is defined by a partial function relating programs to values (p. 67 of Felleisen et al.[7]):

$$eval(e) = v \text{ if } e \mapsto\!\!\!\!\rightarrow_{\mathbf{v}} v, \quad \text{for some } v,$$

where $\mapsto\!\!\!\!\rightarrow_{\mathbf{v}}$ denotes the reflexive, transitive closure of the standard reduction relation.

We have now established the high-level semantic basis for our prototypical language. The semantics is in the form of an evaluation function defined by the reflexive, transitive closure of the standard reduction relation. However, the evaluation function as given does not shed much light on a realistic implementation. At each step, the program is traversed according to the grammar of evaluation contexts until a redex is found. When found, the redex is reduced and the contractum is plugged back into the context. The process is then repeated, again traversing from the beginning of the program. Abstract machines offer an extensionally equivalent but more realistic model of evaluation that shortcuts the plugging of a contractum back into a context and the subsequent decomposition.[6]

2.2. CEK machine

The CEK machine (Interpreter III of Reynolds[20]) (p. 100 of Felleisen et al.[7]) is a state transition system that efficiently performs evaluation of a program. There are two key ideas in its construction, which can be carried out systematically.[2] The first is that substitution, which is not a viable implementation strategy, is instead represented in a delayed, explicit manner as an *environment* structure. So a substitution $[v/x]e$ is represented by e and an environment that maps x to v. Since e and v may have previous substitutions applied, this will likewise be represented with environments. So in general, if ρ is the environment of e and ρ' is the environment of v, then we represent $[v/x]e$ by e in the environment ρ extended with a mapping of x to (v, ρ'), written $\rho[x \mapsto (v, \rho')]$. The pairing of a value and an environment is known as a *closure*.[11]

The second key idea is that evaluation contexts are constructed inside-out and represent continuations:

1. [ ] is represented by **mt**;
2. $E[([\,]e)]$ is represented by $\mathbf{ar}(e', \rho, \kappa)$ where ρ closes e' to represent e and κ represents E; and
3. $E[(v[\,])]$ is represented by $\mathbf{fn}(v', \rho, \kappa)$ where ρ closes v' to represent v and κ represents E.

In this way, evaluation contexts form a program stack: **mt** is the empty stack, and **ar** and **fn** are frames.

States of the CEK machine are triples consisting of an expression, an environment that closes the control string, and a continuation:

$$\begin{aligned} \varsigma \in \textstyle\sum &= Exp \times Env \times Cont \\ v \in Val &= (\lambda x.e) \\ \rho \in Env &= Var \rightharpoonup_{\text{fin}} Val \times Env \\ \kappa \in Cont &= \mathbf{mt} \mid \mathbf{ar}(e, \rho, \kappa) \mid \mathbf{fn}(v, \rho, \kappa). \end{aligned}$$

The transition function for the CEK machine is defined in Figure 1. The initial machine state for a program e is given by the inj_{CEK} function:

$$inj_{CEK}(e) = \langle e, \emptyset, \mathbf{mt} \rangle.$$

Evaluation is defined by the reflexive, transitive closure of the machine transition relation and a "*real*" function (p. 129 of Plotkin[19]) that maps closures to the term represented:

$$eval_{CEK}(e) = real(v, \rho), \quad \text{where } inj_{CEK}(e) \mapsto\!\!\!\!\to_{\text{v}} \langle v, \rho, \mathbf{mt} \rangle,$$

which is equivalent to the *eval* function of Section 2.1:

LEMMA 1 (CEK Correctness[7]) $eval_{CEK} = eval$.

We have now established a correct low-level evaluator for our prototypical language that is extensionally equivalent to the high-level reduction semantics. However, program analysis is not just concerned with the result of a computation, but also with *how* it was produced, i.e., analysis should predict intensional properties of the machine as it runs a program. We therefore adopt a reachable states semantics that relates a program to the set of all its intermediate steps:

$$CEK(e) = \{\varsigma \mid inj_{CEK}(e) \mapsto\!\!\!\!\to_{CEK} \varsigma\}.$$

Membership in the set of reachable states is straightforwardly undecidable. The goal of analysis, then, is to construct an *abstract interpretation*[4] that is a sound and computable approximation of the *CEK* function.

Figure 1. CEK machine.

$\varsigma \longmapsto_{CEK} \varsigma'$

$\langle x, \rho, \kappa \rangle$	$\langle v, \rho', \kappa \rangle$ where $\rho(x) = (v, \rho')$
$\langle (e_0 e_1), \rho, \kappa \rangle$	$\langle e_0, \rho, \mathbf{ar}(e_1, \rho, \kappa) \rangle$
$\langle v, \rho, \mathbf{ar}(e, \rho', \kappa) \rangle$	$\langle e, \rho', \mathbf{fn}(v, \rho, \kappa) \rangle$
$\langle v, \rho, \mathbf{fn}((\lambda x.e), \rho', \kappa) \rangle$	$\langle e, \rho'[x \mapsto (v, \rho)], \kappa \rangle$

We can do this by constructing a machine that is similar in structure to the CEK machine: it is defined by an *abstract state transition* relation, $\mapsto_{\widehat{CEK}}$, which operates over *abstract states,* $\hat{\Sigma}$, that approximate states of the CEK machine. Abstract evaluation is then defined as

$$\widehat{CEK}(e) = \{\hat{\varsigma} \mid inj_{\widehat{CEK}}(e) \mapsto_{\widehat{CEK}} \hat{\varsigma}\}.$$

1. *Soundness* is achieved by showing transitions preserves approximation, so that if $\varsigma \mapsto_{CEK} \varsigma'$ and $\hat{\varsigma}$ approximates ς, then there exists an abstract state $\hat{\varsigma}'$ such that $\hat{\varsigma} \mapsto_{\widehat{CEK}} \hat{\varsigma}'$ and $\hat{\varsigma}'$ approximates ς'.
2. *Decidability* is achieved by constructing the approximation in such a way that the state space of the abstracted machine is finite, which guarantees that for any program e, the set $\widehat{CEK}(e)$ is finite.

An attempt at approximation: A simple approach to abstracting the machine's state space is to apply a *structural abstraction,* which lifts approximation across the structure of a machine state, i.e., expressions, environments, and continuations. The problem with the structural abstraction approach for the CEK machine is that both environments and continuations are recursive structures. As a result, the abstraction yields objects in an abstract state space with recursive structure, implying the space is infinite.

Focusing on recursive structure as the source of the problem, our course of action is to add a level of indirection, forcing recursive structure to pass through explicitly allocated addresses. Doing so unhinges the recursion in the machine's data structures, enabling structural abstraction via a single point of approximation: the store.

The next section covers the first of the two steps for refactoring the CEK machine into its computable approximation: a store component is introduced to machine states and variable bindings and continuations are redirected through the store. This step introduces no approximation and the constructed machine operates in lock-step with the CEK machine. However, the machine is amenable to a direct structural abstraction.

2.3. CESK* machine

The states of the CESK* machine extend those of the CEK machine to include a *store,* which provides a level of indirection for variable bindings and continuations to pass through. The store is a finite map from *addresses* to *storable values,* which includes closures and continuations, and environments are changed to map variables to addresses. When a variable's value is looked-up by the machine, it is now accomplished by using the environment to look up the variable's address, which is then used to look up the value. To bind a variable to a value, a fresh location in the store is allocated and mapped to the value; the environment is extended to map the variable to that address.

To untie the recursive structure associated with continuations, we likewise add a level of indirection through the store and replace the continuation component of the machine with a *pointer* to a continuation in the store. We term the resulting machine the CESK* (control, environment, store,

continuation pointer) machine.

$$\varsigma \in \sum = Exp \times Env \times Store \times Addr$$
$$s \in Storable = Val \times Env + Cont$$
$$\kappa \in Cont = \mathbf{mt} \mid \mathbf{ar}(e, \rho, a) \mid \mathbf{fn}(v, \rho, a).$$

The transition function for the CESK* machine is defined in Figure 2. The initial state for a program is given by the inj_{CESK^*} function, which combines the expression with the empty environment and a store with a single pointer to the empty continuation, whose address serves as the initial continuation pointer:

$$inj_{CESK^*}(e) = \langle e, \emptyset, [a_0 \mapsto \mathbf{mt}], a_0 \rangle.$$

An evaluation function based on this machine is defined following the template of the CEK evaluation given in Section 2.2:

$$eval_{CESK^*}(e) = real(v, \rho, \sigma), \quad \text{where}$$
$$inj_{CESK^*}(e) \mapsto\!\!\!\!\rightarrow_{CESK^*} \langle v, \rho, \sigma, a_0 \rangle,$$

where the *real* function is suitably extended to follow the environment's indirection through the store.

We also define the set of reachable machine states:

$$CESK^*(e) = \{\varsigma \mid inj_{CESK^*}(e) \mapsto\!\!\!\!\rightarrow_{CESK^*} \varsigma\}.$$

Observe that for any program, the CEK and CESK* machines operate in lock-step: each machine transitions, by the corresponding rule, if and only if the other machine transitions.

LEMMA 2 $CESK^*(e) \simeq CEK(e)$

The above lemma implies correctness of the machine.

LEMMA 3 (CESK* Correctness) $eval_{CESK^*} = eval$.

Addresses, abstraction and allocation: The CESK* machine, as defined in Figure 2, nondeterministically chooses addresses when it allocates a location in the store, but because machines are identified up to consistent renaming of addresses, the transition system remains deterministic.

Looking ahead, an easy way to bound the state space of this machine is to bound the set of addresses. But once the store is finite, locations may need to be reused and when multiple values are to reside in the same location; the store will have to soundly approximate this by *joining* the values.

In our concrete machine, all that matters about an allocation strategy is that it picks an unused address. In the abstracted machine however, the strategy *will all but certainly have to reuse previously allocated addresses*. The abstract allocation strategy is therefore crucial to the design of the analysis—it indicates when finite resources should be doled out and decides when information should deliberately be lost in the service of computing within bounded resources. In essence, the allocation strategy is the heart of an analysis.

For this reason, concrete allocation deserves a bit more attention in the machine. An old idea in program analysis is that dynamically allocated storage can be represented by the state of the computation at allocation time[10]; Section 1.2.2 of Midtgaard.[13] That is, allocation strategies can be based on a (representation) of the machine history. Since machine histories are always fresh, we call them *time-stamps*.

A common choice for a time-stamp, popularized by Shivers,[21] is to represent the history of the computation as *contours,* finite strings encoding the calling context. We present a concrete machine that uses a general time-stamp approach and is parameterized by a choice of *tick* and *alloc* functions.

2.4. TIME-STAMPED CESK* MACHINE

The machine states of the time-stamped CESK* machine include a *time* component, which is intentionally left unspecified:

$$t, u \in Time$$
$$\varsigma \in \sum = Exp \times Env \times Store \times Addr \times Time.$$

The machine is parameterized by the functions:

$$tick : \sum \to Time \qquad alloc : \sum \to Addr.$$

The *tick* function returns the next time; the *alloc* function allocates a fresh address for a binding or continuation. We require of *tick* and *alloc* that for all t and ς, $t \neq tick(\varsigma)$ and $alloc(\varsigma) \notin \sigma$ where $\varsigma = \langle \_, \_, \sigma, \_, t \rangle$.

The time-stamped CESK* machine is defined in Figure 3. Note that occurrences of ς on the right-hand side of this definition are implicitly bound to the state occurring on the left-hand side. The evaluation function $eval_{CESK^*_t}$ and reachable states $CESK^*_t$ are defined following the same outline as before and omitted for space. The initial machine state is defined as

$$inj_{CESK^*_t}(e) = \langle e, \emptyset, [a_0 \mapsto \mathbf{mt}], a_0, t_0 \rangle.$$

Satisfying definitions for the parameters are

$$Time = Addr = \mathbb{Z}$$
$$a_0 = t_0 = 0 \qquad tick\langle \_, \_, \_, \_, t \rangle = t + 1 \qquad alloc\langle \_, \_, \_, \_, t \rangle = t.$$

Figure 2. CESK* machine.

$$\varsigma \longmapsto_{CESK^*} \varsigma', \text{ where } \kappa = \sigma(a), b \notin dom(\sigma)$$

$\langle x, \rho, \sigma, a \rangle$	$\langle v, \rho', \sigma, a \rangle$ where $(v, \rho') = \sigma(\rho(x))$
$\langle (e_0 e_1), \rho, \sigma, a \rangle$	$\langle e_0, \rho, \sigma[b \mapsto \mathbf{ar}(e_1, \rho, a)], b \rangle$
$\langle v, \rho, \sigma, a \rangle$ if $\kappa = \mathbf{ar}(e, \rho', c)$	$\langle e, \rho', \sigma[b \mapsto \mathbf{fn}(v, \rho, c)], b \rangle$
if $\kappa = \mathbf{fn}((\lambda x.e), \rho', c)$	$\langle e, \rho'[x \mapsto b], \sigma[b \mapsto (v, \rho)], c \rangle$

Figure 3. Time-stamped CESK* machine.

$\varsigma \longmapsto_{CESK^*_t} \varsigma'$, where $\kappa = \sigma(a), b = alloc(\varsigma), u = tick(\varsigma)$

$\langle x, \rho, \sigma, a, t\rangle$	$\langle v, \rho', \sigma, a, u\rangle$ where $(v, \rho') = \sigma(\rho(x))$
$\langle (e_0 e_1), \rho, \sigma, a, t\rangle$	$\langle e_0, \rho, \sigma[b \mapsto \mathbf{ar}(e_1, \rho, a)], b, u\rangle$
$\langle v, \rho, \sigma, a, t\rangle$ if $\kappa = \mathbf{ar}(e, \rho, c)$	$\langle e, \rho, \sigma[b \mapsto \mathbf{fn}(v, \rho, c)], b, u\rangle$
if $\kappa = \mathbf{fn}((\lambda x.e), \rho', c)$	$\langle e, \rho'[x \mapsto b], \sigma[b \mapsto (v, \rho)], c, u\rangle$

Under these definitions, the time-stamped CESK* machine operates in lock-step with the CESK* machine, and therefore with the CEK machine, implying its correctness.

LEMMA 4 $CESK^*_t(e) \simeq CESK^*(e)$.

The time-stamped CESK* machine forms the basis of our abstracted machine in the following section.

2.5. ABSTRACT TIME-STAMPED CESK* MACHINE

As alluded to earlier, with the time-stamped CESK* machine, we now have a machine ready for direct abstract interpretation via a single point of approximation: the store. Our goal is a machine that resembles the time-stamped CESK* machine, but operates over a finite state space and it is allowed to be nondeterministic. Once the state space is finite, the transitive closure of the transition relation becomes computable, and this transitive closure constitutes a static analysis. Buried in a path through the transitive closure is a possibly infinite traversal that corresponds to the concrete execution of the program.

The abstracted variant of the time-stamped CESK* machine comes from bounding the address space of the store and the number of times available. By bounding the address space, the whole state space becomes finite. (Syntactic sets like *Exp* are infinite, but finite for any given program.) For the purposes of soundness, an entry in the store may be forced to hold several values simultaneously:

$$\hat{\sigma} \in \widehat{Store} = Addr \rightarrow_{\text{fin}} \mathcal{P}(Storable).$$

Hence, stores now map an address to a *set* of storable values rather than a single value. These collections of values model approximation in the analysis. If a location in the store is reused, the new value is joined with the current set of values. When a location is dereferenced, the analysis must consider any of the values in the set as a result of the dereference.

The abstract time-stamped CESK* machine is defined in Figure 4. The non-deterministic abstract transition relation changes little compared with the concrete machine. We only have to modify it to account for the possibility that multiple storable values, which includes continuations, may reside together in the store. We handle this situation by letting the machine non-deterministically choose a particular value from the set at a given store location.

The analysis is parameterized by abstract variants of the functions that parameterized the concrete version:

Figure 4. Abstract time-stamped CESK* machine.

$\hat{\varsigma} \mapsto_{\widehat{CESK}^*_t} \hat{\varsigma}'$, where $\kappa \in \hat{\sigma}(a), b = \widehat{alloc}(\hat{\varsigma}, \kappa), u = \widehat{tick}(\hat{\varsigma}, \kappa)$

$\langle x, \rho, \hat{\sigma}, a, t\rangle$	$\langle v, \rho', \hat{\sigma}, a, u\rangle$ where $(v, \rho') \in \hat{\sigma}(\rho(x))$
$\langle (e_0 e_1), \rho, \hat{\sigma}, a, t\rangle$	$\langle e_0, \rho, \hat{\sigma} \sqcup [b \mapsto \mathbf{ar}(e_1, \rho, a)], b, u\rangle$
$\langle v, \rho, \hat{\sigma}, a, t\rangle$ if $\kappa = \mathbf{ar}(e, \rho', c)$	$\langle e, \rho', \hat{\sigma} \sqcup [b \mapsto \mathbf{fn}(v, \rho, c)], b, u\rangle$
if $\kappa = \mathbf{fn}((\lambda x.e), \rho', c)$	$\langle e, \rho'[x \mapsto b], \hat{\sigma} \sqcup [b \mapsto (v, \rho)], c, u\rangle$

$$\widehat{tick} : \hat{\Sigma} \times Cont \rightarrow Time, \quad \widehat{alloc} : \hat{\Sigma} \times Cont \rightarrow Addr.$$

In the concrete, these parameters determine allocation and stack behavior. In the abstract, they are the arbiters of precision: they determine when an address gets re-allocated, how many addresses get allocated, and which values have to share addresses.

Recall that in the concrete semantics, these functions consume states—not states and continuations as they do here. This is because in the concrete, a state alone suffices since the state determines the continuation. But in the abstract, a continuation pointer within a state may denote a multitude of continuations; however the transition relation is defined with respect to the choice of a particular one. We thus pair states with continuations to encode the choice.

The *abstract* semantics is given by the reachable states:

$$\widehat{CESK}^*_t(e) = \{\hat{\varsigma} \mid \alpha(inj_{CESK^*_t}(e)) \mapsto\!\!\!\!\rightarrow_{\widehat{CESK}^*_t} \hat{\varsigma}\}.$$

Soundness and decidability: We have endeavored to evolve the abstract machine gradually so that its fidelity in soundly simulating the original CEK machine is both intuitive and obvious. To formally establish soundness of the abstract time-stamped CESK* machine, we use an abstraction function, defined in Figure 5, from the state space of the concrete time-stamped machine into the abstracted state space.

The abstraction map over times and addresses is defined so that the parameters $\widehat{alloc}$ and $\widehat{tick}$ are sound simulations of the parameters *alloc* and *tick,* respectively. We also define the partial order ($\sqsubseteq$) on the abstract state space as the natural point-wise, element-wise, component-wise and memberwise lifting, wherein the partial orders on the sets *Exp* and *Addr* are flat. Then, we can prove that abstract machine's transition relation simulates the concrete machine's transition relation.

THEOREM 1 (Soundness)
If $\varsigma \mapsto_{CEK} \varsigma'$ *and* $\alpha(\varsigma) \sqsubseteq \hat{\varsigma}$, *then there exists an abstract state* $\hat{\varsigma}'$, *such that* $\hat{\varsigma} \mapsto_{\widehat{CESK}_t} \hat{\varsigma}'$ *and* $\alpha(\varsigma') \sqsubseteq \hat{\varsigma}'$.

PROOF. By Lemmas 3 and 4, it suffices to prove soundness with respect to $\mapsto_{\widehat{CESK}^*_t}$. Assume $\varsigma \mapsto_{CESK^*_t} \varsigma'$ *and* $\alpha(\varsigma) \sqsubseteq \hat{\varsigma}$. Because ς transitioned, exactly one of the rules from the definition of ($\mapsto_{CESK^*_t}$) applies. We split by cases on these rules. The rule for the second case is deterministic and follows by

Figure 5. Abstraction map, $\alpha : \sum_{CESK^\star_t} \to \sum_{\widehat{CESK}_t}$.

$$
\begin{aligned}
\alpha(e,\rho,\sigma,a,t) &= (e,\alpha(\rho),\alpha(\sigma),\alpha(a),\alpha(t)) && \text{[states]}\\
\alpha(\rho) &= \lambda x.\alpha(\rho(x)) && \text{[environments]}\\
\alpha(\sigma) &= \lambda\hat{a}.\bigsqcup_{\alpha(a)=\hat{a}}\{\alpha(\sigma(a))\} && \text{[stores]}\\
\alpha((\lambda x.e),\rho) &= ((\lambda x.e),\alpha(\rho)) && \text{[closures]}\\
\alpha(\mathbf{mt}) &= \mathbf{mt} && \text{[continuations]}\\
\alpha(\mathbf{ar}(e,\rho,a)) &= \mathbf{ar}(e,\alpha(\rho),\alpha(a))\\
\alpha(\mathbf{fn}(v,\rho,a)) &= \mathbf{fn}(v,\alpha(\rho),\alpha(a))
\end{aligned}
$$

calculation. For the remaining (nondeterministic) cases, we must show an abstract state exists such that the simulation is preserved. By examining the rules for these cases, we see that all three hinge on the abstract store in $\hat{\varsigma}$ soundly approximating the concrete store in ς, which follows from the assumption that $\alpha(\varsigma) \sqsubseteq \hat{\varsigma}$. □

THEOREM 2 (Decidability)
Membership of $\hat{\varsigma}$ in $\widehat{CESK}^\star_t(e)$ is decidable.

PROOF. The state space of the machine is non-recursive with finite sets at the leaves on the assumption that addresses are finite. Hence reachability is decidable since the abstract state space is finite. □

3. ABSTRACT STATE AND CONTROL

We have shown that store-allocated continuations make abstract interpretation of the CESK* machine straightforward. In this section, we want to show that the tight correspondence between concrete and abstract persists after the addition of language features such as conditionals, side effects, and first-class continuations. We tackle each feature, and present the additional machinery required to handle each one. In most cases, the path from a canonical concrete machine to pointer-refined abstraction of the machine is so simple we only show the abstracted system. In doing so, we are arguing that this abstract machine-oriented approach to abstract interpretation represents a flexible and viable framework for building program analyses.

To handle conditionals, we extend the language with a new syntactic form, (if *e e e*), and introduce a base value #f, representing false. Conditional expressions induce a new continuation form: **if** (e'_0, e'_1, ρ, a), which represents the evaluation context E[(if [ ] e_0 e_1)] where ρ closes e'_0 to represent e_0, ρ closes e'_1, to represent e_1, and a is the address of the representation of E.

Side effects are fully amenable to our approach; we introduce Scheme's set! for mutating variables using the (set! *x e*) syntax. The set! form evaluates its subexpression *e* and assigns the value to the variable *x*. Although set! expressions are evaluated for effect, we follow Felleisen et al. and specify set! expressions evaluate to the value of *x* before it was mutated (p. 166 of Felleisen et al.[7]). The evaluation

Figure 6. Abstract extended CESK* machine.

$\hat{\varsigma} \longmapsto_{\widehat{CESK}^\star_t} \hat{\varsigma}'$, where $\kappa \in \hat{\sigma}(a), b = \widehat{alloc}(\hat{\varsigma},\kappa), u = \widehat{tick}(\hat{\varsigma},\kappa)$

$\langle (\texttt{if}\ e_0\ e_1\ e_2), \rho, \hat{\sigma}, a, t\rangle$	$\langle e_0, \rho, \hat{\sigma} \sqcup [b \mapsto \mathbf{if}(e_1, e_2, \rho, a)], b, u\rangle$
$\langle \texttt{\#f}, \rho, \hat{\sigma}, a, t\rangle$ if $\kappa = \mathbf{if}(e_0, e_1, \rho', c)$	$\langle e_1, \rho', \sigma, c, u\rangle$
$\langle v, \rho, \hat{\sigma}, a, t\rangle$ if $\kappa = \mathbf{if}(e_0, e_1, \rho', c)$, and $v \neq \texttt{\#f}$	$\langle e_0, \rho', \hat{\sigma}, c, u\rangle$
$\langle (\texttt{set!}\ x\ e), \rho, \hat{\sigma}, a, t\rangle$	$\langle e, \rho, \hat{\sigma} \sqcup [b \mapsto \mathbf{set}(\rho(x), a)], b, u\rangle$
$\langle v, \rho, \hat{\sigma}, a, t\rangle$ if $\kappa = \mathbf{set}(a', c)$	$\langle v', \rho, \hat{\sigma} \sqcup [a' \mapsto v], c, u\rangle$ where $v' \in \hat{\sigma}(a')$
$\langle (\lambda x.e), \rho, \hat{\sigma}, a, t\rangle$ if $\kappa = \mathbf{fn}(\texttt{callcc}, \rho', c)$	$\langle e, \rho[x \mapsto b], \hat{\sigma} \sqcup [b \mapsto c], c, u\rangle$ where $c = \widehat{alloc}(\hat{\varsigma},\kappa)$
$\langle c, \rho, \hat{\sigma}, a, t\rangle$ if $\kappa = \mathbf{fn}(\texttt{callcc}, \rho', a')$	$\langle a, \rho, \hat{\sigma}, c, u\rangle$
$\langle v, \rho, \hat{\sigma}, a, t\rangle$ if $\kappa = \mathbf{fn}(c, \rho', a')$	$\langle v, \rho, \hat{\sigma}, c, u\rangle$

context E[(set! x [ ])] is represented by **set**(a_0, a_1), where a_0 is the address of x's value and a_1 is the address of the representation of E.

First-class control is introduced by adding a new base value callcc which reifies the continuation as a new kind of applicable value. Denoted values are extended to include representations of continuations. Since continuations are store-allocated, we choose to represent them by address. When an address is applied, it represents the application of a continuation (reified via callcc) to a value. The continuation at that point is discarded and the applied address is installed as the continuation.

The resulting grammar is

$$
\begin{aligned}
e \in Exp &= \ldots \mid (\text{if } e\ e\ e) \mid (\text{set! } x\ e)\\
\kappa \in Cont &= \ldots \mid \text{if}(e, e, \rho, a) \mid \text{set}(a, a)\\
v \in Val &= \ldots \mid \#\text{f} \mid \text{callcc} \mid a.
\end{aligned}
$$

We show only the abstract transitions (Figure 6), which result from store-allocating continuations, time-stamping, and abstracting the concrete transitions for conditionals, mutation, and control. The first three machine transitions deal with conditionals; here we follow the Scheme tradition of considering all non-false values as true. The fourth and fifth transitions deal with mutation.

The remaining three transitions deal with first-class control. In the first of these, callcc is being applied to a closure value *v*. The value *v* is then "called with the current continuation," i.e., *v* is applied to a value that represents the continuation at this point. In the second, callcc is being applied to a continuation (address). When this value is applied to the reified continuation, it aborts the current computation, installs itself as the current continuation, and puts the reified continuation "in the hole." Finally, in the third, a continuation is being

applied; c gets thrown away, and v gets plugged into the continuation b. In all cases, these transitions result from pointer-refinement, time-stamping, and abstraction of the usual machine transitions.

4. ABSTRACT GARBAGE COLLECTION

Garbage collection determines when a store location has become unreachable and can be re-allocated. This is significant in the abstract semantics because an address may be allocated to multiple values due to finiteness of the address space. Without garbage collection, the values allocated to this common address must be joined, introducing imprecision in the analysis (and inducing further, perhaps spurious, computation). By incorporating garbage collection in the abstract semantics, the location may be proved to be unreachable and safely *overwritten* rather than joined, in which case no imprecision is introduced.

Like the rest of the features addressed in this paper, we can incorporate abstract garbage collection into our static analyzers by a straightforward pointer-refinement of textbook accounts of concrete garbage collection, followed by a finite store abstraction.

Concrete garbage collection is defined in terms of a GC machine that computes the reachable addresses in a store (p. 172 of Felleisen et al.[7]):

$$\langle \mathcal{G}, \mathcal{B}, \sigma \rangle \mapsto_{GC} \langle (\mathcal{G} \cup LL_\sigma(\sigma(a))) \setminus (\mathcal{B} \cup \{a\})), \mathcal{B} \cup \{a\}, \sigma \rangle \quad \text{if } a \in \mathcal{G}.$$

This machine iterates over a set of reachable but unvisited "grey" locations $\mathcal{G}$. On each iteration, an element is removed and added to the set of reachable and visited "black" locations $\mathcal{B}$. Any newly reachable and unvisited locations, as determined by the "live locations" function LL_σ, are added to the grey set. When there are no grey locations, the black set contains all reachable locations. Everything else is garbage.

The live locations function computes a set of locations which may be used in the store. Its definition varies based on the machine being garbage collected, but the definition appropriate for the CESK* machine of Section 2.3 is

$$\begin{aligned} LL_\sigma(e) &= \emptyset \\ LL_\sigma(e, \rho) &= LL_\sigma(\rho|\mathbf{fv}(e)) \\ LL_\sigma(\rho) &= rng(\rho) \\ LL_\sigma(\mathbf{mt}) &= \emptyset \\ LL_\sigma(\mathbf{fn}(v, \rho, a)) &= \{a\} \cup LL_\sigma(v, \rho) \cup LL_\sigma(\sigma(a)) \\ LL_\sigma(\mathbf{ar}(e, \rho, a)) &= \{a\} \cup LL_\sigma(e, \rho) \cup LL_\sigma(\sigma(a)). \end{aligned}$$

We write $\rho|\mathbf{fv}(e)$ to mean ρ restricted to the domain of free variables in e. We assume the least-fixed-point solution in the calculation of the function LL in cases where it recurs on itself.

The pointer-refinement requires parameterizing the LL function with a store used to resolve pointers to continuations. A nice consequence of this parameterization is that we can re-use LL for *abstract garbage collection* by supplying it an abstract store for the parameter. Doing so only necessitates extending LL to the case of sets of storable values:

$$LL_\sigma(S) = \bigcup_{s \in S} LL_\sigma(s)$$

The CESK* machine incorporates garbage collection by a transition rule that invokes the GC machine as a subroutine to remove garbage from the store (Figure 7). The garbage collection transition introduces non-determinism to the CESK* machine because it applies to any machine state and thus overlaps with the existing transition rules. The non-determinism is interpreted as leaving the choice of *when* to collect garbage up to the machine.

The abstract CESK* incorporates garbage collection by the *concrete garbage collection transition*, i.e., we reuse the definition in Figure 7 with an abstract store, $\hat{\sigma}$, in place of the concrete one. Consequently, it is easy to verify abstract garbage collection approximates its concrete counterpart.

The CESK* machine may collect garbage at any point in the computation, thus an abstract interpretation must soundly approximate *all possible choices* of when to trigger a collection, which the abstract CESK* machine does correctly. This may be a useful analysis *of* garbage collection, however it fails to be a useful analysis *with* garbage collection: for soundness, the abstracted machine must consider the case in which garbage is never collected, implying no storage is reclaimed to improve precision.

However, we can leverage abstract garbage collection to reduce the state space explored during analysis and to improve precision and analysis time. This is achieved (again) by considering properties of the *concrete* machine, which abstract directly; in this case, we want the concrete machine to deterministically collect garbage. Determinism of the CESK* machine is restored by defining the transition relation as a non-GC transition (Figure 2) followed by the GC transition (Figure 7). This state space of this concrete machine is "garbage free" and consequently the state space of the abstracted machine is "abstract garbage free."

In the concrete semantics, a nice consequence of this property is that although continuations are allocated in the store, they are deallocated as soon as they become unreachable, which corresponds to when they would be popped from the stack in a non-pointer-refined machine. Thus the concrete machine really manages continuations like a stack.

Figure 7. GC transition for the CESK* machine.

$$\varsigma \longmapsto_{CESK^*} \varsigma'$$

$$\langle e, \rho, \sigma, a \rangle \quad | \quad \langle e, \rho, \{\langle b, \sigma(b) \rangle \mid b \in \mathcal{L}\}, a \rangle$$

$$\text{if } \langle LL\sigma(e, \rho) \cup LL\sigma(\sigma(a)), \{a\}, \sigma \rangle \longmapsto\!\!\!\!\!\rightarrow_{GC} \langle \emptyset, \mathcal{L}, \sigma \rangle$$

Similarly, in the abstract semantics, continuations are deallocated as soon as they become unreachable, which often corresponds to when they would be popped. We say often, because due to the finiteness of the store, this correspondence cannot always hold. However, this approach gives a good finite approximation to infinitary stack analyses that can always match calls and returns.

5. RELATED WORK

The study of abstract machines for the λ-calculus began with Landin's SECD machine,[11] the systematic construction of machines from semantics with Reynolds's definitional interpreters,[20] the theory of abstract interpretation with the seminal work of Cousot and Cousot,[4] and static analysis of the λ-calculus with Jones's coupling of abstract machines and abstract interpretation.[9] All have been active areas of research since their inception, but only recently have well-known abstract machines been connected with abstract interpretation by Midtgaard and Jensen.[14, 15] We strengthen the connection by demonstrating a general technique for abstracting abstract machines.

The approximation of abstract machine states for the analysis of higher-order languages goes back to Jones,[9] who argued abstractions of regular tree automata could solve the problem of recursive structure in environments. We reinvoked that wisdom to eliminate the recursive structure of continuations by allocating them in the store.

Midtgaard and Jensen present a 0CFA for a CPS language.[14] The approach is based on Cousot-style calculational abstract interpretation,[3] applied to a functional language. Like the present work, Midtgaard and Jensen start with a known abstract machine for the concrete semantics, the CE machine of Flanagan et al.,[8] and employ a reachable-states model. They then compose well-known Galois connections to reveal a 0CFA with reachability in the style of Ayers.[1] The CE machine is not sufficient to interpret direct-style programs, so the analysis is specialized to programs in continuation-passing style.

Although our approach is not calculational like Midtgaard and Jensen's, it continues in their vein by applying abstract interpretation to well-known machines, extending the application to direct-style machines to obtain a parameterized family of analyses that accounts for polyvariance.

Static analyzers typically hemorrhage precision in the presence of exceptions and first-class continuations: they jump to the top of the lattice of approximation when these features are encountered. Conversion to continuation- and exception-passing style can handle these features without forcing a dramatic ascent of the lattice of approximation.[21] The cost of this conversion, however, is lost knowledge—both approaches obscure static knowledge of stack structure, by desugaring it into syntax.

Might and Shivers introduced the idea of using abstract garbage collection to improve precision and efficiency in flow analysis.[16] They develop a garbage collecting abstract machine for a CPS language and prove it correct. We extend abstract garbage collection to direct-style languages interpreted on the CESK machine.

6. CONCLUSIONS AND PERSPECTIVE

We have demonstrated a derivational approach to program analysis that yields novel abstract interpretations of languages with higher-order functions, control, state, and garbage collection. These abstract interpreters are obtained by a straightforward pointer refinement and structural abstraction that bounds the address space, making the abstract semantics safe and computable. The technique allows concrete implementation technology, such as garbage collection, to be imported straightforwardly into that of static analysis, bearing immediate benefits. More generally, an abstract machine based approach to analysis shifts the focus of engineering efforts from the design of complex analytic models such as involved constraint languages back to the design of programming languages and machines, from which analysis can be *derived*. Finally, our approach uniformly scales up to richer language features such as laziness, stack-inspection, exceptions, and object-orientation. We speculate that store-allocating bindings and continuations is sufficient for a straightforward abstraction of most existing machines.

Looking forward, a semantics-based approach opens new possibilities for design. Context-sensitive analysis can have daunting complexity,[24] which we have made efforts to tame,[17] but modular program analysis is crucial to overcome the significant cost of precise abstract interpretation. Modularity can be achieved without needing to design clever approximations, but rather by designing *modular semantics* from which modular analyses follow systematically.[23] Likewise, *push-down analyses* offer infinite state space abstractions with perfect call-return matching while retaining decidability. Our approach expresses this form of abstraction naturally: the store remains bounded, but continuations stay on the stack.

Acknowledgments

We thank Matthias Felleisen, Jan Midtgaard, Sam Tobin-Hochstadt, and Mitchell Wand for discussions, and the anonymous reviewers of *ICFP'10* for their close reading and helpful critiques; their comments have improved this paper.

Van Horn's work is supported by the National Science Foundation under grant 0937060 to the Computing Research Association for the CIFellow Project. Might's research is based upon work supported by the National Science Foundation under Grant No. 1035658. C

References

1. Ayers, A.E. Abstract analysis and optimization of Scheme. PhD thesis, Massachusetts Institute of Technology (1993).
2. Biernacka, M., Danvy, O. A concrete framework for environment machines. *ACM Trans. Comput. Logic 9*, 1 (2007) 1–30.
3. Cousot, P. The calculational design of a generic abstract interpreter. In M. Broy and R. Steinbrüggen. eds. *Calculational System Design*. NATO ASI Series F. IOS Press, Amsterdam (1999).
4. Cousot, P., Cousot, R. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL '77: Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (New York, 1977), ACM, 238–252.
5. Danvy, O. An analytical approach to program as data objects. DSc thesis, Department of Computer Science, Aarhus University (October, 2006).
6. Danvy, O., Nielsen, L.R. Refocusing in reduction semantics. *Research Report BRICS* RS-04-26, Department of Computer Science, Aarhus University

(November 2004).
7. Felleisen, M.R., Findler, B., Flatt, M. *Semantics Engineering with PLT Redex*. MIT Press (August, 2009).
8. Flanagan, C., Sabry, A., Duba, B.F., Felleisen, M. The essence of compiling with continuations. In *PLDI '93: Proceedings of the ACM SIGPLAN 1993 Conference on Programming Language Design and Implementation* (New York, June 1993), ACM, 37–247
9. Jones, N.D. Flow analysis of lambda expressions (preliminary version). In *Proceedings of the 8th Colloquium on Automata, Languages and Programming* (Springer-Verlag, 1981), 14–128.
10. Jones, N.D., Muchnick, S.S. A flexible approach to interprocedural data flow analysis and programs with recursive data structures. In *POPL '82: Proceedings of the 9th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '82 (New York, 1982), ACM, 66–74.
11. Landin, P.J. The mechanical evaluation of expressions. *Comput. J. 6*, 4 (1964), 308–320.
12. Meunier, P.R., Findler, B., Felleisen, M. Modular set-based analysis from contracts. In *POPL '06: Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (New York, January, 2006), ACM, 218–231.
13. Midtgaard, J. Control-flow analysis of functional programs. *ACM Computing Surveys*, (2012), Forthcoming.
14. Midtgaard, J., Jensen, T. A calculational approach to control-flow analysis by abstract interpretation. In M. Alpuente and G. Vidal. eds. *SAS*, volume 5079 of *LNCS*, Springer (2008), 347–362.
15. Midtgaard, J., Jensen, T.P. Control-flow analysis of function calls and returns by abstract interpretation. In *ICFP '09: Proceedings of the 14th ACM SIGPLAN International Conference on Functional Programming* (New York, 2009), ACM, 287–298.
16. Might, M., Shivers, O. Improving flow analyses via ΓCFA: Abstract garbage collection and counting. In *Proceedings of the 11th ACM International Conference on Functional Programming (ICFP 2006)*, (New York, September, 2006), 13–25.
17. Might, M., Smaragdakis, Y., Van Horn, D. Resolving and exploiting the *k*-CFA paradox: Illuminating functional vs. object-oriented program analysis. In *PLDI '10: Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation* (New York, 2010), ACM, 305–315.
18. Nielson, F., Nielson, H.R., Hankin, C. *Principles of Program Analysis*. Springer-Verlag, New York (1999).
19. Plotkin, G. Call-by-name, call-by-value and the λ-calculus. *Theoret. Comput. Sci. 1*, 2 (December 1975), 125–159.
20. Reynolds, J.C. Definitional interpreters for higher-order programming languages. In *ACM '72: Proceedings of the ACM Annual Conference* (1972), ACM, 717–740.
21. Shivers, O. Control-flow analysis of higher-order languages. PhD thesis, Carnegie Mellon University (1991).
22. Sperber, M. Dybvig, R.K., Flatt, M., van Straaten, A., Findler, R. Matthews, J. *Revised [6] Report on the Algorithmic Language Scheme*. Cambridge University Press (2010).
23. Tobin-Hochstadt, S., Horn, D.V. Modular analysis via specifications as values. *CoRR*, abs/1103.1362, (2011).
24. Van Horn, D., Mairson, H.G. Deciding *k*CFA is complete for EXPTIME. In *ICFP '08: Proceedings of the 13th ACM SIGPLAN International Conference on Functional Programming* (New York, 2008), ACM, 275–282.
25. Wright, A.K., Jagannathan, S. Polymorphic splitting: An effective polyvariant flow analysis. *ACM Trans. Program. Lang. Syst. 20*, 1 (1998), 166–207.

David Van Horn (dvanhorn@ccs.neu.edu), Northeastern University, Boston, MA.

Matthew Might (might@cs.utah.edu), University of Utah, Salt lake City, UT.

DOI:10.1145/1995376.1995401

Peter Winkler

Puzzled
Solutions and Sources

Last month (Aug. 2011, p. 120) we posted a trio of brainteasers, including one as yet unsolved, concerning divisibility of numbers. Here, we offer solutions to two of them and a remark about the third. How did you do?

1. Multiples with just zeroes and ones.

Solution. You were asked to determine whether every positive integer divides a number containing only zeroes and ones in its base-10 representation. Seems reasonable. Suppose your number is n, and someone gives you a large random number m. If you now compute the remainder when m is divided by n, you get some number between 0 and n–1; the remainder is denoted m mod n. If m mod $n = 0$, m is a multiple of n, you might expect this to happen about one time in n. There are infinitely many numbers with base-10 representations containing only zeroes and ones, so unless there is some good reason why not, lots of them ought to be multiples of n. But how to prove it?

One clever way was suggested by Muthu Muthukrishnan of Rutgers University: Consider the numbers 1, 11, 111, 1111, etc. up to 111... 1, where the last number has n+1 digits. Call these numbers m_1, m_2, ... , m_n+1. Each has a remainder when divided by n, and two of these remainders must be the same. Why? Because there are n+1 of them but only n values a remainder can take. This is an application of the famous and useful "pigeonhole principle"; that is, if n+1 items are put into n boxes, some box must contain at least two items.

Suppose the two numbers with the same remainder are m_i and m_j, with $i < j$. Now subtract the smaller from the larger. The resulting number, $m_i - m_j$, consisting of $j - i$ ones followed by i zeroes, must be a multiple of n.

Now try it for $n = 12$; the numbers m_1 through m_{13} and their remainders begin:

1	1
11	11
111	3
1111	7
11111	11

We can stop here because we've already found two numbers with the same remainder, 11. Subtracting them gives 11100, which must then have remainder 0; indeed, 11100 = 12 x 925.

2. Multiples that are Fibonacci numbers.

Solution. Does every n divide some Fibonacci number? Again, since there are infinitely many Fibonacci numbers, it seems plausible that the answer would be "yes." We can tackle it the same way as in the first solution, using remainders mod n.

This time, it makes sense to keep track of remainders mod n for each consecutive pair of Fibonacci numbers. Note, if the remainders are, say, r and s, then the remainders for the next (overlapping) pair of Fibonacci numbers are s and r+s mod n, and the remainders for the previous pair of Fibonacci numbers are r and s-r mod n. Now try this for $n = 7$; the remainder pairs are (1,1); (1,2); (2,3); (3,5); (5,1); (1,6); (6,0)... Having hit a zero, you now have our multiple of 7.

How do you know you will eventually hit a zero? It follows from three observations: there are only finitely many (n^2, to be exact) possible pairs of remainders; since the process can run backward, as well as forward, it must eventually cycle back to where it began; and the process can start with the pair (0,1). The point behind the third observation is that it does no harm to imagine that the Fibonacci numbers start with 0, 1, instead of the customary 1, 1.

This cute puzzle was given to me over lunch by Richard Stanley of MIT. For more, Gregg Musiker of the University of Minnesota recommends a paper by D.D. Wall: "Fibonacci Series Modulo m" in *American Mathematical Monthly 67* (1960), 525–532.

3. Perfect number *m*.

Solution. The problem was to determine whether there are any odd perfect numbers, a famously difficult question. But why has it attracted so much attention over the centuries? One possible answer is that the odd-perfect-number problem is an example of looking for ways in which numbers do, or do not, behave randomly. But maybe the best answer is that such a question is like a disease to which some of us are immune and others highly susceptible. You probably know in which category you belong.

Peter Winkler (puzzled@cacm.acm.org) is William Morrill Professor of Mathematics and Computer Science at Dartmouth College, Hanover, NH.

All readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

[CONTINUED FROM P. 112] also tend to look in health care, where it's relatively easy to get good problem definitions and good deployment coverage.

Is it difficult to get funding?

Funding the equipment isn't difficult, because the equipment isn't very expensive. It's harder to fund the research. Research funding was doing better before the 2008 financial crisis, but I am optimistic about relationships with foundations and nonprofits, which are more into these things than they used to be. We're also starting to see interest from nontraditional parts of the government, like the U.S. Department of State.

What about your wireless hypothesis, which posits that it's more useful to provide communications and computing capabilities to developing nations than more traditional infrastructure?

In terms of the percentages, I think there are signs that the wireless hypothesis is coming true—that countries that have cellular infrastructure are getting things like roads, too. It's hard to know what it means, but it's certainly correlated, and we'll probably know in another 10 years or so.

What's your take on the larger ICTD community?

I'm very happy with our progress. There's the ACM SIGDEV [ACM's Symposium on Computing for Development, Univ. of London, Dec. 17–18, 2011) that's coming, and ACM India has interest in this space. There are also several conferences that cover different disciplines, as well as workshops from many different fields—for AI, for networking and systems, for HCI. This is the right model, because we're trying to solve problems that require many disciplines. So people in this space have both a community position and a position in which they specialize and teach and do traditional work in their discipline.

How can scientists balance those two roles?

It varies by discipline. I would say that HCI is the easiest because they have a long history of looking at their users as part of the focus of the domain. For other fields that are perhaps more narrowly defined, I would say these projects tend to be about 20% new technology and 80% other stuff. You need a nugget and a strong insight for your domain, and then you have to do all this other work like understanding the problem, maybe even discovering the problem, and trying something in the field, because that's the litmus test. It's unpredictable how things are going to work.

"Computer science needs to be a part of almost every discipline now, and it's not clear to me that computer scientists have stepped up to that role yet."

Earlier this year, you began a two-year stint at Google. How did that happen?

I was doing some consulting with Google, and in particular with Google.org, which is its philanthropic arm. Some of the senior Google management asked me about my opinion on some things, and I guess they wanted my opinion on a lot more things, which led to an offer I couldn't refuse.

What are you working on?

I'm looking quite broadly at ways that Google can improve its own infrastructure to make it more innovative in the long term. There's great hardware coming, and things like flash storage that really change some of the equations. Looking long term, I would also like to see how we can provide the cloud to another billion or two billion people. There's bandwidth coming into Africa at unprecedented levels, particularly to east Africa, because of the undersea cables. We still have to figure out how to get it inland, and we need to figure out how to build and operate mobile phone and cloud-based services in developing countries. Right now there are very few data centers in Africa. You end up having to go to Europe and the U.S., and that's a long way to go for every object on a Web page.

You've been involved with a good range of environments throughout your career, from startups to academia to traditional industry.

I like both academia and industry for different reasons. There are certain things where academia is a better place to have an impact, where things are a little longer term or where there's not a clear market yet. There are other places where I prefer industry, especially when you want to get something from an idea stage to affecting a billion people. So I will continue to cross that line back and forth.

What are some of the things you think the field still needs to work on?

One thing that needs more thought is how to make computer science a good player in multi-disciplinary research. Computer science needs to be a part of almost every discipline now, and it's not clear to me that computer scientists have stepped up to that role yet. It's not easy to do. Tenure cases are still based on a single discipline. Funding at NSF is single discipline. As I mentioned earlier, a lot of projects end up being 20% technology and 80% other stuff—so you need reviewers that respect that other stuff and understand what's hard or valuable about it.

I imagine you've learned a lot about interdisciplinary research through your ICTD projects.

I've learned that it's hard. It's harder for faculty than for grad students. Here at Berkeley, we've been able to train a generation of graduate students that really know both social science and computer science. It's much easier to learn in grad school, when you have the time. I've been learning as I go—learning from my students, from colleagues, sometimes sitting in on classes. But the future of multi-disciplinary research will be through students who have been trained in multiple disciplines. C

Leah Hoffmann is a technology writer in Brooklyn, NY.

DOI:10.1145/1995376.1995402

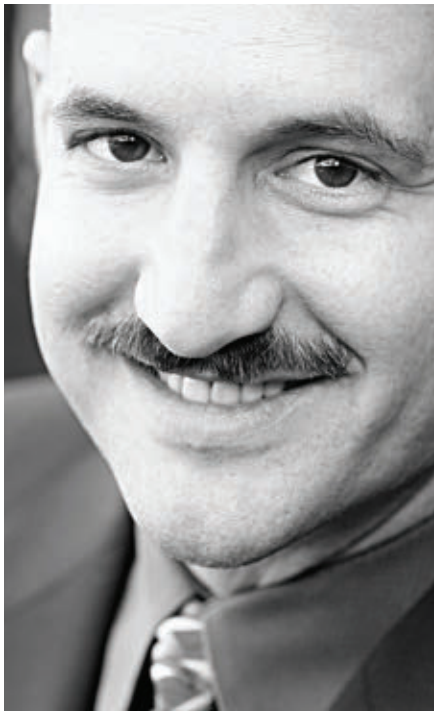
Leah Hoffmann

Q&A
Scaling Up

Eric Brewer talks about infrastructure, connectivity, and computing for developing nations.

THE UNIVERSITY OF California, Berkeley's Eric Brewer has covered a lot of ground in his 20-year career. He was among the earliest to recognize the need for large-scale Web services, building scalable servers with clusters of commodity nodes and laying the foundation for contemporary cloud computing. He co-founded Inktomi, a search engine startup that peaked at $241 per share and $300 million in annual revenue in 2000 before collapsing as clients like Exodus filed for bankruptcy. (It was sold to Yahoo! in 2003.)

He has also been deeply involved with Information and Communication Technologies for Development, spearheading projects to bring telemedicine to Indian villages and develop long-distance Wi-Fi networks in rural areas. In May, he began yet another chapter with a two-year assignment at Google, where he is working on developing the company's next-generation infrastructure.



For the past 10 years, you've been involved with a number of computing projects that benefit developing countries. Tell us about your recent work in that domain.

One of the things we're working on is building a low-cost GSM base station that's appropriate for rural villages. Rural connectivity is expensive. Base stations take a lot of power, so you need a big diesel generator. Then you need to bring diesel to the generator, which means you need a road—often, you've got to build it—and you need trucks to bring the diesel to the generator. On top of that, if you're building a road, you probably want to be in a relatively flat area, and you need a very tall tower to get coverage.

The base station we're building takes only roughly 50 watts, which means it can be run on solar or wind power and can be located up on a hill, in a place that has good visibility to villages.

Does it leverage your previous work on low-cost, long-distance Wi-Fi?

Our previous work in solar solutions and long-distance Wi-Fi are both very relevant as they greatly reduce the cost of the power system and backhaul solution; we use long-distance Wi-Fi instead of microwave links to backhaul the traffic into an urban area that has relatively low-cost bandwidth.

What inspired you to tackle the project? In the past, I understand you were less convinced that cellular connectivity is the best solution for rural areas since Wi-Fi is cheaper to implement.

The strong urban success of cellular means that many rural folks have phones, even if their village does not have coverage. Some use these phones when in coverage, others use them as FM radios, and still others have them mostly as a status symbol. Nonetheless, the demand for rural cellular is very clear, and the phones are often already there and waiting.

Much of your work is done through Technology and Infrastructure for Emerging Regions, or TIER, a research group you founded at the University of California, Berkeley. How did TIER get started?

The biggest influences on the founding of TIER came out of Inktomi. First, I was traveling quite a bit, and I'd been invited to the World Economic Forum, where I had the privilege of meeting a wide variety of very sharp people from developing nations. Many of these folks were articulate about the problems in their country, and almost all the time my reaction was that technology had a role to play in solving them. And another factor was that Inktomi had done so well that I was, at least on paper, extremely wealthy, and starting to think more seriously about addressing some of these problems. Of course, I don't have that money anymore, so I decided to focus on solutions via research by creating a community within computer science that could address these great challenges.

What's your process for finding new projects?

I tend to prefer infrastructure problems. I like to have at least half of my students working on core infrastructure, things like connectivity and power in particular. I [CONTINUED ON P. 111]

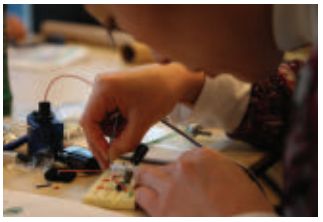
PHOTOGRAPH BY PETER BURANZON

Discover the Future of CG at SIGGRAPH Asia 2011

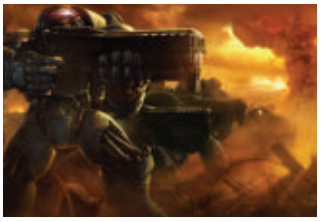
Be inspired by this year's spectacular line-up of conference programs. Establish connections with top-notch industry players on the exhibition floor and with academics at SIGGRAPH Asia 2011.

Screenings of Animation and Visual Effects

Demonstrations, Displays and Installations

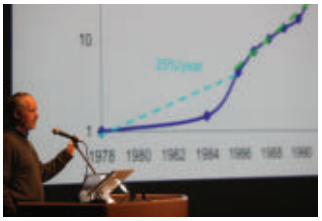


Instructional Sessions



Production Talks

Products, Services and Recruitment



Industry Insights and Forecasts

Networking Opportunities

Research and Development Presentations

REGISTER NOW AND SAVE

Network with and reach out to the industry's movers and shakers.

Choose from Full Conference or Basic Conference Passes, with admission to all or a variety of programs and events.

Join us in Hong Kong this December by registering online from August at http://www.siggraphasia2011.com

Register by 31 October to enjoy Early Bird discounts!

SIGGRAPHASIA2011 HONG KONG

The 4th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia

Conference: 12-15 December | **Exhibition:** 13-15 December Hong Kong Convention and Exhibition Centre

Sponsored by ACM SIGGRAPH

www.SIGGRAPH.org/ASIA2011

MODULARITY
aosd • 2012

Hasso-Plattner-Institut Potsdam, Germany

March 25—30, 2012

General Chair
Robert Hirschfeld,
Hasso-Plattner-Institut Potsdam, Germany

Organizing Chair
Michael Haupt,
Oracle Labs, Potsdam, Germany

Research Results Chair
Éric Tanter, *Universidad de Chile, Chile*

Modularity Visions Chair
Kevin Sullivan, *University of Virginia, USA*

Heart of Technology Lectures Chairs
Richard P. Gabriel, *IBM Research, USA*
Robert Krahn, *Hasso-Plattner-Institut Potsdam, Germany*

Workshops Chairs
Sven Apel, *University of Passau, Germany*
Bastian Steinert,
Hasso-Plattner-Institut Potsdam, Germany

Demonstrations and BoFs Chairs
Carl Friedrich Bolz,
Heinrich-Heine-Universität Düsseldorf, Germany
Damien Cassou,
University of Bordeaux, France

Industry Chair
Bogdan Franczyk,
Universität Leipzig, Germany

Student Events Chairs
Hidehiko Masuhara,
The University of Tokyo, Japan
Michael Perscheid,
Hasso-Plattner-Institut Potsdam, Germany

Publicity Chairs
Eric Bodden, *Technische Universität Darmstadt, Germany*
Monica Pinto, *Universidad de Málaga, Spain*

Student Volunteers Chairs
Ruzanna Chitchyan,
Lancaster University, UK
Jens Lincke, *Hasso-Plattner-Institut Potsdam, Germany*

Administrative Coordinator
Sabine Wagner,
Hasso-Plattner-Institut Potsdam, Germany

Web Chair
Tobias Pape,
Hasso-Plattner-Institut Potsdam, Germany

Design & Layout
Constanze Langer,
University of Applied Sciences Magdeburg-Stendal, Germany

SIGSOFT
SPECIAL INTEREST GROUP ON SOFTWARE ENGINEERING

acm SIGPLAN

>>>>> www.aosd.net/2012