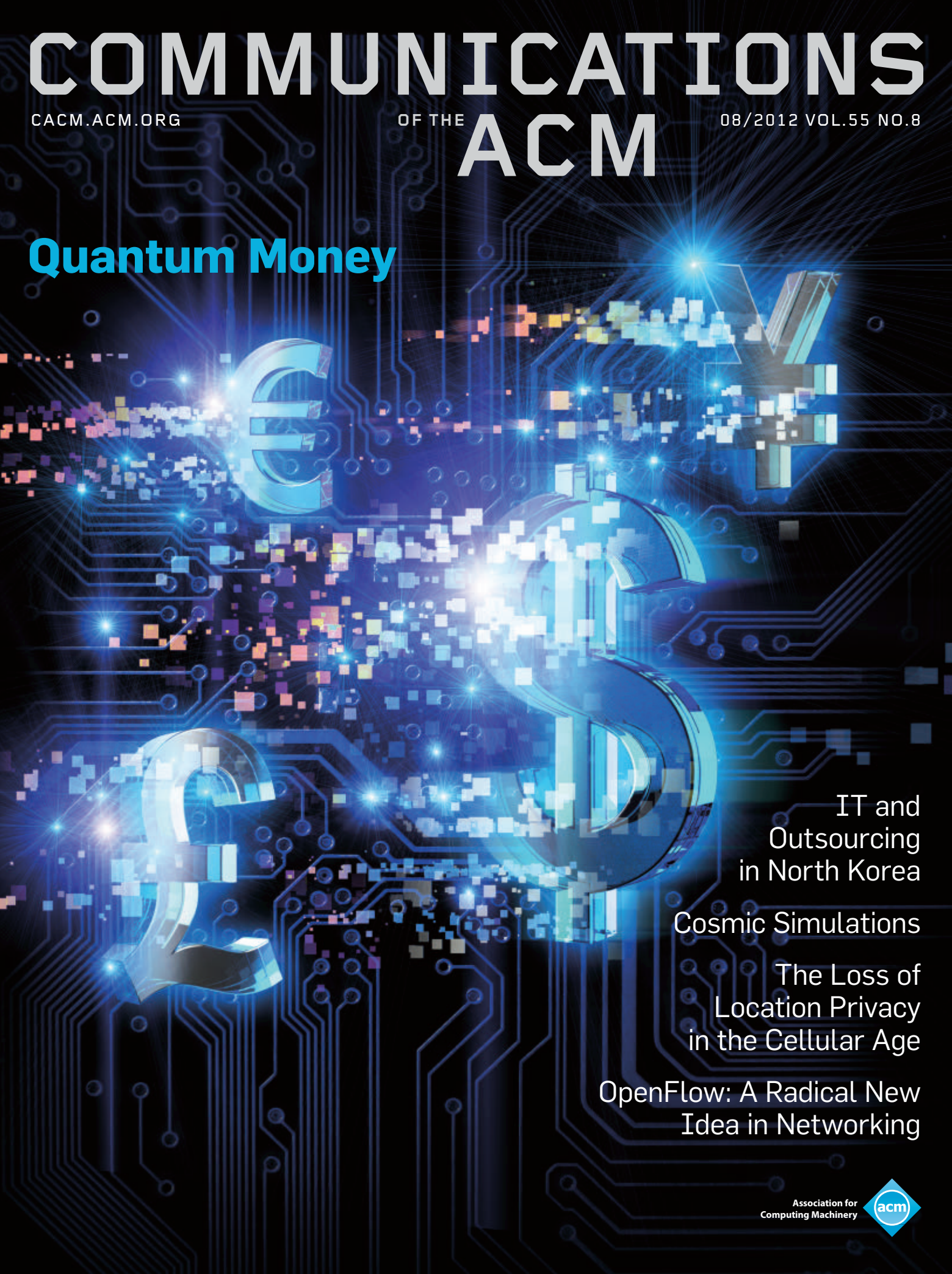
COMMUNICATIONS OF THE ACM
CACM.ACM.ORG
08/2012 VOL.55 NO.8

Quantum Money



IT and Outsourcing in North Korea

Cosmic Simulations

The Loss of Location Privacy in the Cellular Age

OpenFlow: A Radical New Idea in Networking

Association for Computing Machinery

MATCH YOUR SERVER TO YOUR BUSINESS.

ONLY PAY FOR WHAT YOU NEED!

With a 1&1 Dynamic Cloud Server, you can change your server configuration in real time.

- Independently configure CPU, RAM, and storage
- Control costs with pay-per-configuration and hourly billing
- Up to 6 Cores, 24 GB RAM, 800 GB storage
- 2000 GB of traffic included free
- Parallels® Plesk Panel 10 for unlimited domains, reseller ready
- Up to 99 virtual machines with different configurations

1&1 DYNAMIC CLOUD SERVER

3 MONTHS FREE!*

Base Configuration, Starting at $49.99/month

- **NEW:** Monitor and manage your cloud server through 1&1 mobile apps for Android™ and iPhone®.

1&1®

MEMBER OF united internet

www.1and1.com

*Offer valid for a limited time only. 3 months free only applies to $49.99 base configuration. Set-up fee of $49.00 applies. Base configuration includes 1 processor core, 1 GB RAM, 100 GB Storage. Other terms and conditions may apply. Visit www.1and1.com for full promotional offer details. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners.

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

Please see our website for exact eligibility rules. 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, 2012** 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.

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 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 2013.

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

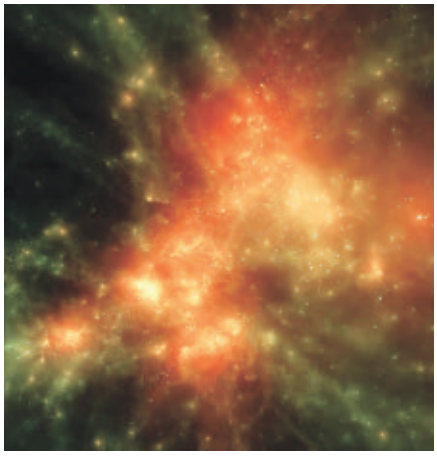
Association for Computing Machinery

COMMUNICATIONS OF THE ACM

Departments

Last Byte

News



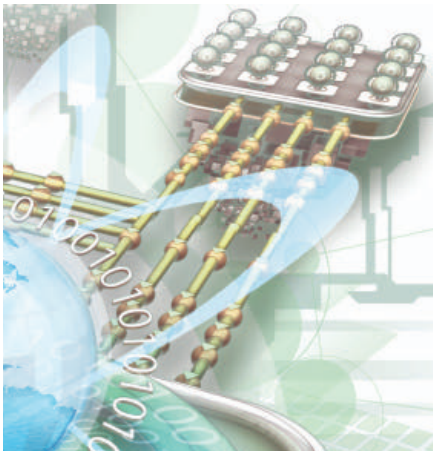
Viewpoints

Association for Computing Machinery
Advancing Computing as a Science & Profession

IMAGE COURTESY OF MULTIDARK RESIMULATION PROJECT (GUSTAVO YEPES, SPLOTCH)

08/2012
VOL. 55 NO. 08

Practice



Articles' development led by acmqueue
queue.acm.org

Contributed Articles

Review Articles

Research Highlights

About the Cover:
Imagine money you can spend without a trace and without the worry of loss or counterfeit. Enter quantum information as the basis for a better kind of money. This month's cover story (p. 84) explores what it would take to realize quantum money. Cover illustration by Spooky Pooka at Début Art.

ILLUSTRATIONS BY JASON COOK, BRIAN GREENBERG, SPOOKY POOKA AT DÉBUT ART

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:10.1145/2240236.2240237

Bill Poucher

Giving Students the Competitive Edge

The annual ACM International Collegiate Programming Contest (ICPC) shines the spotlight on the next generation of problem solvers during their university years, engaging

them in a competition that develops teamwork, programming skills, and algorithmic mastery. Doors are opened for students to measure their prowess among their peers as they push human problem-solving performance beyond accepted norms. ICPC is a competition of global proportions; participation is open to every student at every university on the planet.

This year over 25,000 students from over 2,200 universities competed in regional contests that spanned the globe. The top 112 teams of three competed in the 36th Annual ACM-ICPC World Finals sponsored by IBM and hosted by the University of Warsaw. For full coverage of this event, which took place May 14–18, I encourage you to visit ICPC Digital at http://icpc.baylor.edu/digital/ for full coverage.

Allow me to recap the highlights here:

Officials from the University of Warsaw, the City of Warsaw, along with leading lights from Poland's science and economics communities worked together to roll out the red carpet, giving the event national exposure with full TV coverage. Leading the opening events for ACM-ICPC World Finals Week was the President of the Republic of Poland, Bronislaw Komorowski.

IBM, completing 15 years of a 20-year sponsor commitment, kicked off the week by introducing their latest cognitive computing and big data research in Warsaw's newly completed Copernicus Science Center—one of the most advanced science museums in Europe, hosting over 450 interactive exhibits.

The opening ceremony featured an extraordinary celebration of 5D Arts in the Palace of Culture and Science. We are indebted to the University of Warsaw, alumni, and supporters—specifically Finals Director Jan Madey and Rector Chałasińska-Macukow—for transforming the university into an ICPC Village with extraordinary hospitality.

The 2012 World Finals, held on May 17, 2012 at the University of Warsaw School of Management, was indeed action packed. With only seconds to go, the top two teams were neck-and-neck. Both teams solved nine of the 12 problems posed, with St. Petersburg State University of IT, Mechanics and Optics edging out the University of Warsaw to earn the coveted title of ACM-ICPC World Champions. Congratulations to Eugeny Kapun, Mikhail Kever, Niyaz Nigmatullin, and coach Andrey Stankevich!

ICPC is a competition of global proportions; participation is open to every student at every university on the planet.

Throughout the event there was spectacular TV coverage within Poland featuring highlights and interviews with the hometown team from the University of Warsaw: Jakub Pachocki, Tomasz Kulczyński, and Wojciech Śmietanka.

And the coverage did not stop there. On May 22, 2012, Russia's President Vladimir Putin invited the 2012 ICPC World Champions to the annual meeting of the Russian Academy of Science. In a speech unfolding the national focus on science in the Russian Federation over the next five years, Mr. Putin remarked:

"Incidentally, present here at this meeting today are team members of the St. Petersburg State University of Information Technologies, Mechanics and Optics, which won the 2012 ACM International Collegiate Programming Contest. So our victories are not limited to hockey but extend to such academic disciplines as well. I congratulate them on this achievement.

The brilliant success of our student team is a prime example of effective integration of science and education, quality training of creative and intelligent young people who will doubtlessly be in demand in all areas of life in our country such as Russian science, including fundamental [research]."

The 2013 World Finals will be held in St. Petersburg, Russia, in St. Isaac's Square, courtesy of St. Petersburg State University of Information Technologies, Mechanics and Optics, the Russian Duma, and IBM.

Now for the editorial.

Poland gets it.

Russia gets it, too.

Bill Poucher (poucherw@acmicpc.org) is Executive Director of ICPC and a professor of computer science at Baylor University, Waco, TX.

The Ultimate Online Resource for Computing Professionals & Students

Association for Computing Machinery

Advancing Computing as a Science & Profession

DOI:10.1145/2240236.2240238

Composable Trees for Configurable Behavior

I CONCUR WHOLEHEARTEDLY with the composability benefits Brian Beckman outlined in his article "Why LINQ Matters: Cloud Composability Guaranteed" (Apr. 2012) due to my experience using composability principles to design and implement the message-dissemination mechanism for a mobile ad hoc router in a proprietary network. In it, the message-dissemination functionality of the router emerges from the aggregation of approximately 1,500 nodes in a composable tree that resembles a large version of the lambda-tree diagrams in the article. However, instead of being LINQ-based, each node represents a control element (such as if/else, for-loop, and Boolean operations nodes), as well as nodes that directly access message attributes. Each incoming message traverses the composable tree, with control nodes directing it through pertinent branches based on message attributes (such as message type, timestamp, and sender's location) until the message reaches processing nodes that complete the dissemination.

Since assembling and maintaining a 1,500-node tree within the code base would be daunting, a parser assembles the tree from a 1,300-line routing-rule specification based on a domain-specific language (DSL).[a] Defining the routing rules through this DSL-assembled composable tree also provides these additional benefits:

Nodes verified independently. Verifying the if/else, message-timestamp, and other nodes can be done in isolation;

Routing rules modified for unit testing. As the routing rules mature, their execution requires a full lab- or field-configuration environment, making it difficult to test new features; a quick simplification of a local copy of the DSL specification defines routing rules that bypass irrelevant lab/field constraints while focusing on the feature being tested on the developer's desktop;

Scalable and robust. New routing rules can be added to DSL specification; new routing concepts can be added through the definition of new node types; and new techniques can be added to the overall design; and

Each message traversal recorded by the composable tree. Each node in the composable tree logs a brief one-line statement describing what it was doing and why the message chose a particular traversal path; the aggregation of these statements provides an itinerary describing the journey of each message traversal through the composable tree for confirmation or debugging.

My experience with composable trees defined through a DSL has been so positive I would definitely consider using the technique again to solve problems that are limited in scope but unlimited in variation.

Jim Humelsine, Neptune, NJ

a http://en.wikipedia.org/wiki/Domain-specific_language

Model Dependence in Sample-Size Calculation

We wish to clarify and expand on several points raised by Martin Schmettow in his article "Sample Size in Usability Studies" (Apr. 2012) regarding sample-size calculation in usability engineering, emphasizing the challenges of calculating sample size for binomial-type studies and identifying promising methodologies for future investigation.

Schmettow interpreted "overdispersion" as an indication of the variability of the parameter p; that is, when n Bernoulli trials are correlated (dependent), the variance can be shown as $np(1-p)(1+C)$, where C is the correlation parameter, and when $C>0$ the result is overdispersion. When the Bernoulli trials are negatively correlated, or $C<0$, the result is "underdispersion." If the trials are independent, then $C=0$, corresponding to the binomial model. Bernoulli trials may thus result in overdispersion or underdispersion; in practice, overdispersion is more common due to the heterogeneity of populations/samples.

A widely used approach for modeling an overdispersed binomial model is to consider p as a random variable to account for all uncertainty. A common model for p is the beta distribution that leads to a well-known prototype model for overdispersion, the "beta-binomial distribution." Note this model assumes a particular parametric distribution of the random variable p. However, sample-size calculations based on this paradigm also involve computational challenges; M'Lan et al.[1] concluded that choosing the criterion for sample-size determination from the many criteria in the literature is ultimately based on personal taste. Note, too, that Schmettow's "zero-truncated logit-normal binomial model" follows this scheme. To the best of our knowledge, the Bernstein–Dirichlet process is a promising family for such a modeling framework; a nice feature of the related distribution of p is that any density in `(0, 1]` can be approximated by the Bernstein polynomial.

A more common approach is to fix p through widely used methods involving fixed p based on the confidence-interval formulas derived from normal approximations to the binomial distribution requiring an estimate of p as input into the sample-size formula. However, the normal-based-interval approximation is well known for being erratic for small sample sizes, and even for large samples when p is near the boundaries 0 or 1.

Current sample-size-calculation procedures are thus highly model-dependent, so results will differ. This means a universal procedure that works for a particular binomial-type process is as yet nonexistent, and more studies are needed. We hope Schmettow's article and our discussion here inspire more researchers to take on the subject of sample-size calculation for usability studies.

Dexter Cahoy and Vir Phoha, Ruston, LA

Reference

1. M'Lan, C.E., Joseph, L., and Wolfson, D.B. Bayesian sample size determination for binomial proportions. *Bayesian Analysis 3*, 2 (Feb. 2008), 269–296.

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

Global Reach for Global Opportunities in Computing

Dear Colleague,

Today's computing professionals are at the forefront of the technologies that drive innovation across diverse disciplines and international boundaries with increasing speed. In this environment, ACM offers advantages to computing researchers, practitioners, educators and students who are committed to self-improvement and success in their chosen fields.

ACM members benefit from a broad spectrum of state-of-the-art resources. From Special Interest Group conferences to world-class publications and peer-reviewed journals, from online lifelong learning resources to mentoring opportunities, from recognition programs to leadership opportunities, ACM helps computing professionals stay connected with academic research, emerging trends, and the technology trailblazers who are leading the way. These benefits include:

Timely access to relevant information

- ***Communications of the ACM*** magazine
- ***ACM Queue*** website for practitioners
- Option to subscribe to the ***ACM Digital Library***
- ACM's ***50+ journals and magazines*** at member-only rates
- ***TechNews***, tri-weekly email digest
- ***ACM SIG conference*** proceedings and discounts

Resources to enhance your career

- **ACM Tech Packs**, exclusive annotated reading lists compiled by experts
- **Learning Center** books, courses, podcasts and resources for lifelong learning
- Option to join **37 Special Interest Groups** (SIGs) and **hundreds of local chapters**
- **ACM Career & Job Center** for career-enhancing benefits
- ***CareerNews***, email digest
- **Recognition of achievement** through Fellows and Distinguished Member Programs

As an ACM member, you gain access to ACM's worldwide network of more than 100,000 members from nearly 200 countries. ACM's global reach includes councils in Europe, India, and China to expand high-quality member activities and initiatives. By participating in ACM's multi-faceted global resources, you have the opportunity to develop friendships and relationships with colleagues and mentors that can advance your knowledge and skills in unforeseen ways.

ACM welcomes computing professionals and students from all backgrounds, interests, and pursuits. Please take a moment to consider the value of an ACM membership for your career and for your future in the dynamic computing profession.

Sincerely,

Vint Cerf

President
Association for Computing Machinery

Association for
Computing Machinery

Advancing Computing as a Science & Profession

Association for Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: AD13

You can join ACM in several easy ways:

Online	Phone	Fax
http://www.acm.org/join	*+1-800-342-6626 (US & Canada)* *+1-212-626-0500 (Global)*	*+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/about/code-of-ethics

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

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:10.1145/2240236.2240239 http://cacm.acm.org/blogs/blog-cacm

Machine Learning and Algorithms; Agile Development

John Langford poses questions about the direction of research for machine learning and algorithms. Ruben Ortega shares lessons about agile development practices like Scrum.

John Langford
"Research Directions for Machine Learning and Algorithms"
http://cacm.acm.org/blogs/blog-cacm/108385
May 16, 2011

S. Muthu Muthukrishnan invited me to the National Science Foundation's Workshop on Algorithms in the Field with the goal of providing a sense of where near-term research should go. When the time came, though, I instead bargained for a post, which provides a chance for other people to comment.

There are several things I did not fully understand when I went to Yahoo! about five years ago. I would like to repeat them as people in academia may not yet understand them intuitively.

1. Almost all the big-impact algorithms operate in pseudo-linear or better time. Think about caching, hashing, sorting, filtering, etc. and you have a sense of what some of the most heavily used algorithms are. This matters quite a bit to machine learning research, because people often work with superlinear time algorithms and languages. Two very common examples of this are graphical models where inference is often a superlinear operation—think about the n^2 dependence on the number of states in a Hidden Markov Model and Kernelized Support Vector Machines where optimization is typically quadratic or worse. There are two basic reasons for this. The most obvious is that linear time allows you to deal with large datasets. A less obvious but critical point is that a superlinear time algorithm is inherently buggy; it has an unreliable running time that can easily explode if you accidentally give it too much input.

2. Almost anything worth doing requires many people working together. This happens for many reasons. One is the time-critical aspect of development—in many places it really is worthwhile to pay more to have something developed faster. Another is that real projects are simply much bigger than you might otherwise expect. A third is that real organizations have people coming and going, and any project that is by just one person withers when that person leaves. This observation means the development of systems with clean abstractions can be extraordinarily helpful, as it allows people to work independently. This observation also means simple widely applicable tricks (for example, the hashing trick) can be broadly helpful.

A good way to phrase research directions is with questions. Here are a few of my natural questions.

1. How do we efficiently learn in settings where exploration is required? These are settings where the choice of action you take influences the observed reward—ad display and medical testing are two good scenarios. This is deeply critical to many applications because the learning with exploration setting is inherently more natural than the standard supervised learning setting. The tutorial we did detailed much of the state of the art here, but very significant questions remain. How can we do effective offline evaluation of algorithms? How can we be both efficient in sample complexity and computational complexity? Several groups are interested in sampling from a Bayesian posterior to solve these sorts of problems. Where and when can this be proved to work? (There is essentially no analysis.) What is a maximally distributed and incentive-compatible algorithm that remains efficient? The last question is very natural for marketplace

design. How can we best construct reward functions operating on different time scales? What is the relationship between the realizable and agnostic versions of this setting, and how can we construct an algorithm that smoothly interpolates between the two?

2. How can we learn from lots of data? We will be presenting a KDD survey/tutorial about what is been done. Some of the larger-scale learning problems have been addressed effectively using MapReduce. The best example I know is Ozgur Cetin's algorithm at Yahoo! It is preconditioned conjugate gradient with a Newton stepsize using two passes over examples per step. (A nonHadoop version is implemented in Vowpal Wabbit for reference.) But linear predictors are not enough; we would like learning algorithms that can, for example, learn from all the images in the world. Doing this well plausibly requires a new approach and new learning algorithms. A key observation here is that the bandwidth required by the learning algorithm cannot be too great.

3. How can we learn to index efficiently? The standard solution in information retrieval is to evaluate (or approximately evaluate) all objects in a database returning the elements with the largest score according to some learned or constructed scoring function. This is an inherently $O(n)$ operation, which is frustrating when it's plausible that an exponentially faster $O(log(n))$ solution exists. A good solution involves both theory and empirical work here as we need to think about how to think about how to solve the problem, and of course we need to solve it.

4. What is a flexible, inherently efficient language for architecting representations for learning algorithms? Right now, graphical models often get (mis)used for this purpose. It is easy and natural to pose a computationally intractable graphical model, implying many real applications involve approximations. A better solution would be to use a different representation language that was always computationally tractable yet flexible enough to solve real-world problems. One starting point for this is Searn. Another general approach was the topic of the Coarse-to-Fine Learning and Inference Workshop. These are inherently related as coarse-to-fine is a pruned breadth first search. Restated, it is not enough to have a language for specifying your prior structural beliefs; instead we must have a language that results in computationally tractable solutions.

5. The deep learning problem remains interesting. How do you effectively learn complex nonlinearities capable of better performance than a basic linear predictor? An effective solution avoids feature engineering. Right now, this is almost entirely dealt with empirically, but theory could easily have a role to play in phrasing appropriate optimization algorithms, for example.

Good solutions to each of these research directions would result in revolutions in their area, and every one of them would plausibly see wide applicability.

What's missing?

Ruben Ortega "Research in Agile Development Practices"

http://cacm.acm.org/blogs/blog-cacm/109811
June 20, 2011

I am an enthusiastic advocate of agile software development practices like Scrum. Its ability to allow teams to focus on delivering product and communicate status has made it one of the easiest and best software development techniques I have seen in a career that has used ad hoc, Waterfall, and everything in between.

Recent research from New Zealand has furthered the cause by performing a study that involved 58 practitioners in 23 organizations over four years. In reading a Victoria University of Wellington article on "Smarter Software Development" and then looking at Rashina Hoda's thesis "Self-Organizing Agile Teams: A Grounded Theory," there are two interesting takeaways:

1. Self-organizing scrum teams naturally perform a balancing act between:

▸ Freedom and responsibility: The team is responsible for collective decision making, assignment, commitment, and measurement, and must choose to do them.

▸ Cross functionality and specialization: The team is responsible for deciding when to distribute work across team members or have each focus on a certain part of the project.

▸ Continuous learning and iteration pressure: The team is responsible for delivering on its own schedule and the retrospectives to improve each sprint.

The advantage of giving this balancing act to the team is that it take ownership of the solution with the full understanding of all the trade-offs that will need to occur each sprint. By distributing the work to the team, it also makes team members accountable to one another for making sure the goals are achieved.

2. Self-organizing teams have their members assume some well-defined roles spontaneously, informally, and transiently to help make their projects successful:

▸ Mentor: Guides the team in the use of agile methods.

▸ Coordinator: Manages customer expectations and collaboration with the team.

▸ Translator: Translates customer business requirements to technical requirements and back.

▸ Champion: Advocates agile team approach with senior management.

▸ Promoter: Works with customers to explain agile development and how to collaborate best with the team.

▸ Terminator: Removes team members that hinders the team's successful functioning.

These roles are an emergent property that comes from using agile development methods. They are not prescribed explicitly as part of any of the agile development philosophies, but they arise as part of successful use of the methodology.

I am eager to see more research emerge as to where agile software development practices succeed and where they need improvement. There is a large body of evidence that shows it to be a successful strategy, and having the research to support it would help encourage its adoption. C

John Langford is a senior researcher at Microsoft Research New York. **Ruben Ortega** is an engineering director at Google.

ACM Member News

MICHAEL STONEBRAKER ON BIG DATA AND LAUNCHING START-UPS

How big is big data? Humans are creating at least 1.8 zettabytes of it a year, according to IDC's annual Digital Universe Study. That is enough to fill 57.5 billion 32GB iPads.

And we are only beginning to see the kind of technological breakthroughs that will allow us to effectively manage it, says big-data pioneer Michael Stonebraker, adjunct professor at the Massachusetts Institute of Technology.

"Everything of any commercial significance will soon be geopositioned by a sensor," he says. "All auto insurers will place one in cars so they can track safety among their drivers. Libraries will tag all books so they can be quickly found if misplaced on a shelf. This is causing an ultimate tsunami of data. There will be an unbelievable amount of advancements over the next 10 years to address this."

Expect Stonebraker to remain on the forefront. His now-legendary work on relational database systems has led to him launching seven start-ups, beginning with Ingres in the 1970s. He continues to work on methods to improve the organization of data for analytics, as well as adapt to the great increases in velocity through which information is now delivered.

It is the excitement of coming up with an innovation—and then overseeing its development via the start-up process—that keeps Stonebraker engaged.

"I like to see my ideas make the light of day," he says. "If you just publish papers at a university, it's likely that no technology transfer will happen. If you approach a large company with your idea, the chances of it getting picked up aren't high. The best way to see your idea through is a start-up. I specialize in disruptive tech. Start-ups are great for that."

—Dennis McCafferty

Science | DOI:10.1145/2240236.2240241

Jeff Kanipe

Cosmic Simulations

With the help of supercomputers, scientists are now able to create models of large-scale astronomical events.

If you are going to build a synthetic universe in a computer and watch it evolve over billions of years, you are going to need a mighty powerful computer, one that will literally go where no computer has gone before. Last fall, astronomers with the University of California High-Performance AstroComputing Center announced they had successfully completed such a model, which they called the Bolshoi simulation. Bolshoi, which is Russian for "great" or "grand," is an apt word choice. The simulation used six million CPU hours on the Pleiades supercomputer at the U.S. National Aeronautic and Space Administration's (NASA's) Ames Research Center, which, as of June 2012, was rated as the world's 11th most powerful computer on the TOP500 list.

Today, there is hardly a field of science that has not been propelled into new territory by supercomputers. You find them in studies as diverse as genome analysis, climate modeling, and seismic wave propagation. In the last five years, supercomputer technology has advanced so significantly in speed and processing capacity that many researchers no longer refer to these powerful mainframes as supercomputers, but rather HPCs, or high-performance computers. These rarefied machines are designed to process vast amounts of scientific data and run simulations based on that data.

A visualization from the Bolshoi simulation depicting the evolution of gas density in the resimulated 007 cluster.

The Pleiades is a marvel of computer technology. The system architecture consists of 112,896 cores housed in 185 refrigerator-sized racks. It can run at a theoretical peak performance of 1.34 petaflops and has a total memory of 191 terabytes. NASA uses the Pleiades for its most demanding modeling and simulation projects in aeronautics, but its applications obviously do not stop there. To produce the Bolshoi simulation, the Adaptive Refinement Tree (ART) algorithm was run on Pleiades. Rather than compute interactions between pairs of particles, the ART algorithm subdivides space into cubical cells and calculates interactions between cells. This allows for more efficient calculations of gravitational interactions among billions of mass

COURTESY OF MULTIDARK RESIMULATION PROJECT (GUSTAVO YEPES, SPLOTCH)

particles, making it the perfect tool for recreating an unfolding cosmos.

Bolshoi's purpose was to do that and more. It would model not just how the visible universe of stars, gas, and dust evolved, but also how the vast majority of the invisible universe, which is composed of dark matter, evolved. Dark matter is a crucial component of the simulation because, although it cannot yet be directly detected (it can only be inferred from its gravitational effects on normal matter), galaxies are thought to have formed within huge "cocoons" of dark matter, called dark matter halos.

Astronomers actually ran two Bolshoi simulations on Pleiades: the Bolshoi and the BigBolshoi. The Bolshoi computed the evolution of a volume of space measuring about one billion light-years on a side containing more than one million galaxies. The simulation begins about 24 million years after the big bang, which occurred 13.7 billion years ago, and follows the evolution of 8.6 billion dark-matter particles, each with an assigned mass of 200 million times that of the sun, to the present day. Logistically, the simulation required 13,824 cores and a cumulative 13 terabytes of RAM. In all, 600,000 files were saved, filling 100 terabytes of disk space. During the Bolshoi simulation's run, 180 "snapshots" were made showing the evolutionary process at different times. These visualizations will allow astrophysicists to further analyze how dark matter halos, galaxies, and clusters of galaxies coalesced and evolved.

The BigBolshoi simulation was of a lower resolution but covered a volume of four billion light-years, 64 times larger than the Bolshoi model. Its purpose was to predict the properties and distribution of galaxy clusters and superclusters throughout this volume of space.

Any simulation that strives to reflect real processes in nature requires a significant amount of observational input data and a well-founded theory that explains what is observed. In the former case, the Bolshoi simulation is based on precise measurements of the vestigial all-sky afterglow of the big bang, which astronomers call cosmic microwave background radiation. The measurements, made over several years using NASA's Wilkinson Microwave Anisotropy Probe spacecraft, revealed that the background radiation is not perfectly uniform, but exhibits tiny variations, regions that are slightly more or less dense by one part in 100,000. These fluctuations correspond to non-uniformities in the otherwise uniform distribution of matter in the very early universe, and are essentially the seedlings from which emerged all of the galaxies observed in the universe today.

The 180 "snapshots" taken during the Bolshoi simulation will allow astrophysicists to analyze how dark matter halos, galaxies, and clusters of galaxies coalesced and evolved.

Just how the universe went from a nearly smooth initial state to one so full of complex structure has long puzzled cosmologists, but most agree the best explanation is the Lambda Cold Dark Matter theory, or ΛCDM. Once referred to as just the Cold Dark Matter theory, it has since been augmented to included dark energy (Λ), a mysterious counteractive force to gravity that has been invoked to explain the accelerating expansion of the universe. The theory makes specific predictions for how structure in the universe grows hierarchically as smaller objects merge into bigger ones. Because ΛCDM explains much of what is observed, the Bolshoi simulation drew upon this model as its theoretical framework.

The results of both simulations largely confirm cosmologists' assumptions about the formation of large-scale structure, but one discrepancy needs to be addressed.

"Our analysis of the original Bolshoi simulation showed that dark matter halos that could host early galaxies are much less abundant than earlier estimates indicated," says Joel Primack, director of the University of California High-Performance AstroComputing Center and coauthor of the paper announcing the results of the Bolshoi simulation. The previous estimates predicted the number of halos in the early universe should be 10 times greater than is seen in the Bolshoi simulation. The difference, says Primack, is significant. "It remains to be seen whether this is just observational incompleteness or a potentially serious problem for the standard Lambda Cold Dark Matter theory."

Smaller Simulations

Two other recent computer simulations also have broken new astrophysical ground, but at smaller scales. One is the first realistic simulation, at galactic scales, of how the Milky Way was formed. The simulation, named Eris, shows the origin of the Milky Way beginning one million years after the big bang and traces its evolution to present time. It was produced by a research group run by Lucio Mayer, an astrophysicist at the University of Zurich, and Piero Madau, an astronomer at the University of California, Santa Cruz.

Just exactly how spiral galaxies like ours form has been the subject of contentious debate for decades (hence christening the simulation "Eris," the Greek goddess of strife and discord). Previous simulations resulted in galaxies that were either too small or dense, did not have an extended disk of gas and stars, or exhibited too many stars in the central region. Eris, however, achieved the proper balance. Once again, the Pleiades computer was brought to bear, entailing 1.4 million processor hours. Supporting simulations were performed using the Cray XT5 Monte Rosa computer at Zurich's Swiss National Supercomputing Center at the Swiss Federal Institute of Technology. (The Cray XT5 has since been upgraded to a 400-teraflop Cray XE6.) The simulation modeled the formation of a galaxy with 790 billion solar masses comprised of 18.6 million particles, from which dark matter, gas, and stars form. The results are another confirmation of the Cold Dark Matter theory.

"In this theory, galaxies are the outcome of the gravitational growth of tiny

quantum density fluctuations present shortly after the big bang," says Madau. "The ordinary matter that forms stars and planets has fallen into the gravitational wells created by large clumps of dark matter, giving rise to galaxies in the centers of dark matter halos."

Even smaller, on stellar scales, astronomers have employed HPCs to render a clearer picture of why a class of compact, dense objects called neutron stars often move through space at very high velocities, in some cases 1,000 kilometers or more per second. (Most stars have typical space velocities of a few tens of kilometers per second.) A clue may be found in how neutron stars are created. These objects, in which protons and electrons have been gravitationally compressed into neutrons, form from the collapsing cores of massive stars just before they explode as supernovae. Most astronomers have long been convinced the explosion itself somehow gives the neutron star its high-velocity "kick." To explore this possibility, Adam Burrows, an astrophysicist at Princeton University, created a three-dimensional animation of conditions throughout the star during the explosion. He found that the star does not explode symmetrically, but that the explosion rips through the star asymmetrically. The hydrodynamic recoil from such an explosion is more than sufficient to hurl the neutron star off into space.

"This is a straightforward consequence of momentum conservation when things aren't spherical," says Burrows. "A lot of exotic mechanisms have been proposed, but this simplest of origins seems quite natural."

The simulation was run on the Cray XT5 Kraken supercomputer, which is the 21st most-powerful computer in the world, and is housed at Oak Ridge National Laboratory.

None of these studies would have been possible without HPCs. A regular personal computer, for instance, would have required 570 years to make the same calculations to reproduce the Eris simulation. Still, even as computer performance improves, Primack sees new challenges ahead. For example, if the observational and computational datasets expand exponentially while the speed of data transmission expands arithmetically, methodologies for analysis will need to change. "Instead of bringing data to the desktop, we will increasingly have to bring our algorithms to the data," he says.

Another challenge, says Primack, is the energy cost of computation. "The Department of Energy, which currently has the fastest U.S. supercomputers on the TOP500 list, is not willing to have its supercomputer centers use much more than the 10 megawatts that they currently do. It will be a huge challenge to go from 10 petaflops to one exaflop without increasing the energy consumption by more than a small factor. We don't know how to do this yet."

Despite such challenges, a new era of HPC is dawning. Sequoia, an IBM BlueGene/Q system at Lawrence Livermore National Laboratory, has attained 16 petaflops, and four other HPCs in the U.S. are expected to be in the 10-petaflops range soon. IBM's Mira at the Argonne National Laboratory just attained 10 petaflops, and the Blue Waters project is now online at the National Center for Supercomputer Applications in Urbana-Champaign, IL. Titan should become operational at the Oak Ridge National Laboratory later this year, and Stampede at The University of Texas at Austin is expected to be up and running in January 2013. These and other HPCs promise to reveal new and transformative insights into the world and the universe from the smallest scales to the largest. Their simulations will probe levels of complexity we can only imagine, taking us where no one has gone—or could possibly go—before.

Further Reading

Guedes, J., Callegari, S., Madau, P., and Mayer, L.
Forming realistic late-type spirals in a ΛCDM universe: The Eris simulation, *The Astrophysical Journal 742*, 2, Dec. 2011.

Nordhaus, J., Brandt, T. D., Burrows, A., Almgren, A.
The hydrodynamic origin of neutron kick stars, http://arxiv.org/abs/1112.3342.

Prada, F., Klypin, A., Cuesta, A., Betancort-Rijo, J., and Primack, J.
Halo concentrations in the standard ΛCDM cosmology, http://arxiv.org/pdf/1104.5130.pdf.

Rantsiou, E., Burrows, A., Nordhaus, J., and Almgren, A.
Induced rotation in 3D simulations of Ccre collapse supernovae: Implications for pulsar spins, *The Astrophysical Journal 732*, 1, May 2011.

University of California High-Performance AstroComputing Center
Bolshoi videos, http://hipacc.ucsc.edu/Bolshoi/Movies.html.

Jeff Kanipe is an astronomy writer based in Boulder, CO.

Milestones

Computer Science Awards

ACM, the European Association for Theoretical Computer Science (EATCS), and the Austrian government recently honored five leading computer scientists.

EDSGER W. DIJKSTRA PRIZE

ACM and EATCS selected Maurice Herlihy, J. Eliot B. Moss, Nir Shavit, and Dan Touitou to receive the 2012 Edsger W. Dijkstra Prize in Distributed Computing. Herlihy, a professor of computer science at Brown University, and Moss, a professor of computer science at the University of Massachusetts, were honored for their 1993 "Transactional Memory" paper. Shavit, a professor of computer science at Tel Aviv University, and Touitou, chief technology officer at Toga Networks, were honored for their 1995 paper, "Software Transactional Memory." The Edsger W. Dijkstra Prize is awarded to "outstanding papers on the principles of distributed computing, whose significance and impact on the theory or practice of distributed computing have been evident for at least 10 years."

WITTGENSTEIN AWARD

The Austrian Science Fund on behalf of the Austrian Ministry for Science presented the Wittgenstein Award, Austria's leading science prize, to Thomas Henzinger, president of the Institute of Science and Technology. The Wittgenstein Award honors and supports "scientists working in Austria who have accomplished outstanding scientific achievements."

—Jack Rosenberger

Technology | DOI:10.1145/2240236.2240242 Tom Geller

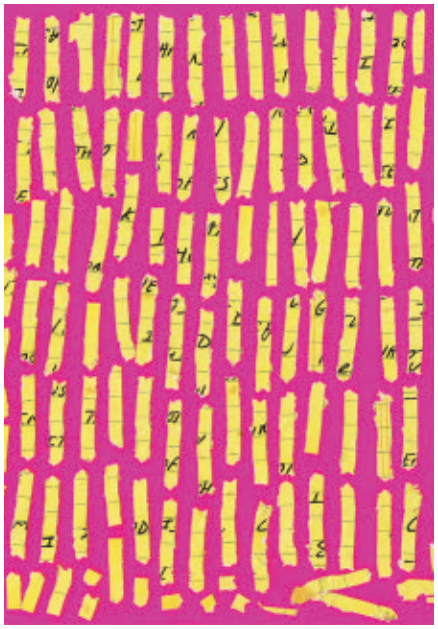
DARPA Shredder Challenge Solved

The eight-person winning team used original computer algorithms to narrow the search space and then relied on human observation to move the pieces into their final positions.

You accidentally shredded an important document. Could you put it back together? Would it be worth the effort? What if it would stop a terrorist plot, or release an innocent person from jail?

Intelligence agencies around the world face questions like these, most famously when the East German secret police abandoned tons of torn pages after the fall of the Berlin Wall in late 1989. Those documents are still being reconstructed, and the creation of a computer program called the e-Puzzler in 2010 increased hopes of finishing the job by 2013. But that program was written to reconstruct those particular documents. It is optimized to handle hand-torn pieces as the East German secret police were forced to manually destroy many of the documents.

To seek "unexpected advances" that could be applied to such situations, the U.S. Defense Advanced Research Projects Agency (DARPA) issued its Shredder Challenge in October 2011, daring the public to unshred five documents and extract their hidden messages. The puzzles were designed to be progressively difficult on multiple axes, ranging from about 200 chads to more than 6,000 each. Text files accompanying the scanned chads provided questions, with points reflecting each question's difficulty. For example, puzzle #3 asked "What is the indicated location?" while the reassembled document showed a set of geographic coordinates and a drawing of Cuba. (Naming the country was worth two points; the city of Cienfuegos was worth an additional six.) Solvers needed to both answer the questions and show how their reconstruction of the document led to that answer. More than 9,000 teams applied for the challenge, but only 69 of them answered one or more questions correctly.



DARPA Shredder Challenge puzzle #1 consisted of more than 200 chads of torn, yellow lined paper with handwritten text.

The winning team was the only one to answer all of the questions in all five puzzles, taking home the $50,000 prize while ending the contest on Dec. 2, 2011, three days ahead of schedule. Like other contest leaders, the eight-person "All Your Shreds Are Belong To U.S." team employed original computer algorithms to narrow the search space, and then relied on human observation to move the pieces into their final positions. In the process the All Your Shreds team discovered unexpected difficulties—and also a peculiarity in the contest that gave it an unexpected edge.

Play to Win

The contest's parameters had a strong effect on how competitors approached the problem. As Don Engel of the two-person, second-place "Schroddon" team says, "If we had taken a completely manual approach, we'd lose because teams can be of any size. If we'd taken only a computer-driven approach, we'd also lose because there are people out there who are better at computer vision." In fact, a team from University of California, San Diego attempted to crowdsource the challenge, setting it up as a sort of game and inviting the public to help them via the Internet. Saboteurs foiled the team's strategy, accusing it of "cheating" and moving shreds to incorrect positions. The team responded by implementing security measures, which also slowed its progress, knocking it from third to sixth place.

The winning All Your Shreds team took advantage of the fact the shredded documents were apparently photocopied before being placed face-up on a solid background and scanned, resulting in a phenomenon known as printer steganography. "High-quality printers and photocopiers, in the United States at least, have a watermark on them so the Secret Service can track phony money," explains team member Otávio Good, founder of Quest Visual. "It's a repeating pattern of yellow dots. If you can find it, you can use that to your advantage. It's essentially like a map of how the pieces go together." However, Good pointed out they were "yellow dots on yellow paper, and the chads were small. It helped us tremendously, but we were pretty much in the lead even before we found the dots. In my opinion, we were still on track to win... it was part of the game."

"The challenge and its component puzzles were designed to resemble the problem facing an intelligence analyst," says Norman Whitaker, deputy director of DARPA's Information Innovation Office. "That analyst is often less interested in putting the pieces together than in getting crucial information off the page in the shortest time possible." In explaining why the challenge could only be a small representation of what is found in the field, he says, "[It's] narrow in scope to make it tractable. It concerned English, hand-

written documents shredded with a commercial shredder. Additional research and development would be required to apply this work to actual confiscated documents." Even then, he defined the problem of document reconstruction as "just one facet of information assurance, an aspect that is often overlooked."

Other contest features played unexpected roles in defining the winning solutions. The Schroddon team found the shredders sometimes sheared the paper in its depth. As team member Marianne Engel notes, "The computer looked at [the edges of such chads] as though they were blank areas, but the human eye could see that the chads were just torn." In addition, the five puzzles varied in the type of shredder used; the number and size of pages; the documents' content; and whether the pages were on lined, unlined, or graph paper. As a result, says Good, "we essentially hard-coded our program to work for each individual puzzle. We didn't create a general 'unscramble the puzzle' codebase."

One method that helped Good's team was the use of a scoring system that improved its algorithms. "As we adjusted our algorithms, we could see whether they were working," says Good. "At that point the efficiency of our recommendation algorithm went up by a factor of 10 within a few hours. So although each puzzle started out difficult, it got easier as we went."

The Human Element

One fact emerged early on in the contest: The puzzles would not be solved by computers alone. Matthias Prandtstetter, a researcher at the Austria Institute of Technology who investigated document reconstruction while at the Vienna University of Technology, acknowledged that much of the scholarly literature on the subject stops where human interaction takes over, thereby failing to address certain real-world issues. For example, "[computer algorithms] might be able to reconstruct a document that's in multiple columns or a table, but not have them in the right order," he says, while a human could finish the job by understanding each portion in context. Prandtstetter suggests language-recognition algorithms could extend those for reconstruction, but believes the pairing of these technologies is in its infancy.

The problem of document reconstruction is an often-overlooked aspect of information assurance, says DARPA's Norman Whitaker.

Specialized knowledge also came into play, for example, as Don Engel of the second-place Schroddon team found he was able to apply his master's work in statistical natural language processing to the task. "One mind-set I took from my studies was that of conditional probability: When you have a few characters, what are the chances that the next one with be a specific character?" says Don Engel. "So as we were assembling shreds, I grepped the dictionary file on my computer to find words with those characters in sequence. Also, for puzzle #5 the question was, 'What are the three translated messages?' That implied that they were either in some foreign language or encoded. I saw a very large number of the letter 'd' and thought, 'That's way more than you'd see in English.' The same was true with certain pairs of letters, such as 'di' and 'ah' and 'it.' As it turned out, the answer to puzzle #5 was in Morse code, written as 'dit dit dah.' "

Part of the All Your Shreds solution involved team members remotely reviewing the computer's piece-pair recommendations via the Internet, then manually making selections and placing pieces. Knowing they would ultimately spend hundreds of hours in this pursuit, its programmers created a custom client to make the interface as fast as possible. "We wanted things to be very user-friendly because the human was a big element," says Good. "When we clicked on a recommendation, we wanted that to happen in real-time—maybe one-tenth of a second on a fast computer. We set up a node.js server to allow team members to place puzzle pieces over the Internet, even while the programmers worked on algorithms. There were about eight bottlenecks in the program, but because we wrote the client in C#, we were able to use C#'s parallel libraries to get past them by parallelizing across all our processors."

Indeed, the subject of document shredding is one that measures human value against human trouble. We shred documents when we believe their value is greater than the reconstruction cost. As long as that cost involves human intervention—a point the Shredder Challenge did not disprove—that equation still holds. As Don Engel says, "If you look at the last 6,000-piece puzzle, that's 6,000 times 6,000 possible pairs, which is just intractable for a human. So we needed a computer to filter for pairs that were plausible. But then, we could use our human brains to determine which ones made sense." C

Further Reading

Justino, E., Oliveira, L.S., and Freitas, C.
Reconstructing shredded documents through feature matching, *Forensic Science International 160*, 2, July 13, 2006.

Prandtstetter, M. and Günther, R.R.
Combining forces to reconstruct strip shredded text documents, *Proceedings of the 5th International Workshop on Hybrid Metaheuristics*, Malaga, Spain, Oct. 8–9, 2008.

Schauer, S., Prandtstetter, M., and Günther, R.R.
A memetic algorithm for reconstructing cross-cut shredded text documents, *Proceedings of the 7th International Workshop on Hybrid Metaheuristics*, Vienna, Austria, Oct. 1–2, 2010.

Skeoch, A.
An investigation into automated shredded document reconstruction using heuristic search algorithms, Ph.D. thesis, University of Bath, 2006.

Ukovich, A., Ramponi, G., Doulaverakis, H., Kompatsiaris, Y., and Strintzis, M.
Shredded document reconstruction using MPEG-7 standard descriptors, *Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology*, Rome, Italy, Dec. 18–24, 2004.

Tom Geller is an Oberlin, OH-based science, technology, and business writer.

Society | DOI:10.1145/2240236.2240243 Samuel Greengard

Advertising Gets Personal

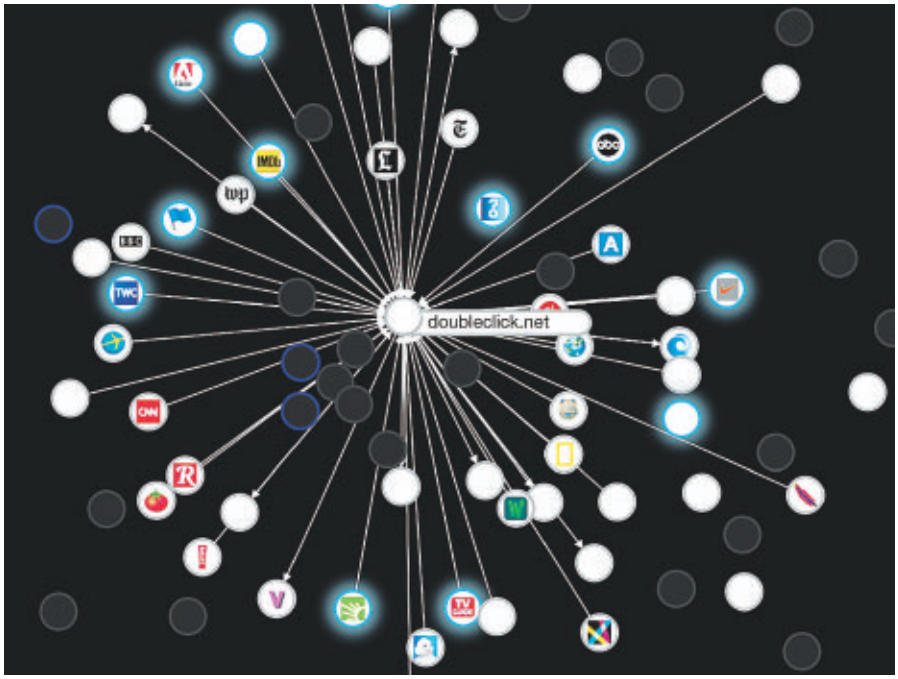
Online behavioral advertising and sophisticated data aggregation have changed the face of advertising and put privacy in the crosshairs.

Not long ago, a man walked into the local Target store in Minneapolis and demanded to speak to the manager. He wanted to know why his then-high school daughter was receiving coupons and promotions for maternity clothing, cribs, and other items that would indicate she was pregnant. "Are you trying to encourage her to get pregnant?" he asked. The store manager examined the stack of coupons and promptly apologized. He said he did not have any idea why the girl had received the coupons. The man then left for home.

This could have been the end of the story. But after talking to his daughter the man discovered she was pregnant. Target had used sophisticated predictive analytics to determine that her previous buying patterns and behavior had indicated a high probability of expecting a baby. In fact, Target and other stores have become so good at gauging customers' buying patterns they now disguise customer-specific promotions by including coupons that are completely irrelevant to the recipient.

Welcome to the new world of advertising. As statisticians, software developers, and advertising experts mine and mix growing volumes of online and offline data and develop increasingly complex algorithms, they are building new and remarkably sophisticated advertising models designed to maximize results—and revenues. "Technology is enabling new—and in some cases hyper-local and personalized—forms of advertising," states John Nicholson, counsel for the law firm Pillsbury Winthrop Shaw Pittman. However, "there's a fine line between what's acceptable and what constitutes an invasion of privacy."

"Online behavioral advertising methods are advancing at an incredibly fast pace," says Lorrie Faith Cranor, an associate professor of computer science, engineering, and public policy at Carnegie Mellon University. "There are clearly advantages to receiving relevant ads, but the Internet, combined with today's data-collection technology, poses serious privacy concerns. Unfortunately, most consumers feel as though they have little control over what happens to their data and how it is used by advertisers."



Collusion, a Firefox add-on, lets a person see all the third-party entities tracking his or her movements across the Web.

By the Numbers

Over the years, advertisers have struggled to better understand the whims of the marketplace and target consumers more effectively. Identifying market niches and customer segments has been a daunting task and there has been no easy way to deliver relevant ads. The result? Most ads target broad demographic segments through television, radio, newspapers, magazines, kiosks, billboards, shopping carts, and other media. In many cases, advertisers simply hope for positive results and learn by trial and error.

However, the last few years have brought about a revolution in data mining particularly as online and conventional methods have allowed advertisers to assemble and reassemble data in new ways. A growing number of retailers, including Target, assign each customer a unique ID number or guest code. It is associated with a credit or debit card, and an individual's purchase history is stored for analysis. This is separate from a loyalty program, and the only way to avoid tracking is to pay with cash and avoid giving a phone number or any other personal data. But the process does not stop there. Increasingly, retailers and others plug in information from third-party sources that track the same individual. This might include the person's Web browsing patterns, credit history, what magazines they read, and even conversations they have had at social media sites.

The result is a fairly comprehensive picture of an individual's buying hab-

SCREENSHOT COURTESY OF MOZILLA.ORG/COLLUSION

its and consumption patterns. This profile—which could include anything from the type of tea or liquor a person likes to consume to medical conditions and sexual orientation—allows marketers to customize ads, but it also offers deep insights into life events and changes. For example, when a woman begins buying vitamin supplements, larger quantities of skin lotion, hand sanitizers, and a larger purse or bag there is an extremely high likelihood she is pregnant. In addition, analytics software has become so sophisticated it is possible to estimate the delivery window within a few weeks.

Of course, using data to predict life events could have far-reaching consequences, particularly if family, friends, or a prospective employer become aware of a sensitive lifestyle or medical issue, such as an affinity for nude beaches or a diagnosis of HIV. Worse, the data may contain errors and present an inaccurate picture that could lead to an employer refusing to hire the person or the loss of a job. As a result, advertisers are attempting to get smarter—some would say sneakier—in the way they deliver ads. Increasingly, they are including coupons and ads that are completely random or irrelevant in order to appear as though they are not spying over a person's shoulder.

Joseph Turian, president of consulting firm MetaOptimize, says that as organizations learn to use analytics and cultivate big data, insights that would have been unimaginable only a few years ago are moving into the mainstream. There are clear advantages for consumers—particularly those looking for discounts and deals—but advertisers need to avoid stepping over the line. "People like the idea of personalized searches and advertising," says Turian. "Many already provide data willingly for discounts through rewards programs. But they want to be in control of their destiny."

Cookies, Tweets, and Dollars

What makes the emerging field of data aggregation and analytics possible is a spate of online data-collection techniques that revolve around IP addresses, third-party cookies, and Web tools that track consumers as they click through Web sites and interact online. Internet service providers, Web sites, and advertising networks sell this data to other companies, including data aggregators. Google, meanwhile, collects data from searches and through keywords in Gmail and YouTube while Facebook has unlimited access to the mother lode of information and messages that appear on its site. Finally, Twitter recently sold its multibillion tweet archive to a U.K. firm that reportedly has more than 1,000 companies lined up for the data.

Analytics software has become so sophisticated it is now possible to estimate a pregnant customer's delivery window within a few weeks.

Today's data-collection system is largely based on an opt-out model that is nearly impossible to understand or manage, many privacy advocates contend. Consumers face the daunting task of trying to decipher lengthy and convoluted privacy policies that in some cases do not match actual practices, Cranor says. What is more, data collection firms often rely on loopholes and devious methods to circumvent cookie-blocking tools built into Web browsers and privacy tools such as Ghostery. In the end, users' attempts to control tracking and personal data often ends up resembling a game of Whac-A-Mole, Nicholson says.

In fact, half of all Internet users recall the ads they view but only 12% correctly remember the disclosure tag-lines attached to ads, Cranor reports. When she studied usage patterns she found that the majority of participants mistakenly believe that ads pop up if they click on disclosure icons and taglines. AdChoices, the tagline most commonly used by online advertisers (it discloses sites' advertising methods and allows consumers to click a button and opt out), was par-

Cloud Computing

New RaaS Pricing Model

The resources behind cloud computing services will soon be sold in increments of seconds, according to researchers from Technion-Israel Institute of Technology.

Providers of cloud computing have moved from renting servers on a monthly basis to renting virtual "server equivalents" for as little as an hour at a time. But even that is inefficient, say the researchers, who presented a paper, "The Resource-as-a-Service (RaaS) Cloud," at the recent USENIX Hotcloud '12 conference in Boston. Providers are moving toward pricing individual resources, such as memory, within a virtual machine, and changing prices in intervals of seconds, based on shifting demand. That trend from infrastructure-as-a-service to resource-as-a-service can save buyers money, earn more for providers, and make efficient use of hardware and energy.

"Clients don't need to buy things they don't need, hosts don't need to sell them things they don't need, and hosts can accommodate more clients on a server," says Orna Agmon Ben-Yehuda, a doctoral student and co-author of the paper.

Clients would use an "economic agent" that makes split-second decisions on how much to spend on which resources, and hosts would allocate resources based on how much a client was willing to pay. Cloud service providers' software could also incorporate economic agents to represent their own business interests.

Coauthor and doctoral student Muli Ben-Yehuda says the trend demands a lot of cloud computing researchers. Software, for instance, will have to adapt to use an ever-shifting set of resources, and workloads will need to be carefully balanced. The challenge, he says, is how to turn computing into a commodity. "How do you make computing something like electricity? It's there whenever you want it, you can have as much as you need, and the price is set by the market."

—Neil Savage

ticularly ineffective at communicating notice and choice. Nearly half of the participants who saw AdChoices believed it was intended to sell advertising space, while a mere 27% believed it was a means to stop tailored ads. "A majority of participants mistakenly believed that opting out would stop all online tracking, not just tailored ads," she notes.

Critics believe the inability to control what software and tracking mechanisms are placed on a person's computer is nothing less than a violation. Many Web sites contain a half-dozen to a dozen or more tracking tools or third-party cookies. It is akin to a company installing video cameras and microphones in a home and recording everything that occurs in the household. "When people find out what is really happening, the typical response is 'Are you kidding!'" says Marcella Wilson, an adjunct professor of computer science at the University of Maryland, Baltimore County.

Privacy Matters

In February, U.S. President Obama unveiled a Consumer Privacy Bill of Rights as part of a comprehensive blueprint to expand privacy protections while continuing to make the Internet a hub of innovation and economic growth. The measure attempts to force companies such as Google, Microsoft, and Yahoo! to stop monitoring when a person clicks a Do Not Track button on their Web browser. Do Not Track is intended to supersede a decade-old voluntary industry initiative called P3P, which has produced tepid results and proved unenforceable. It was designed to offer consumers some control over the type of data collected, how it could be used, and how long it could be stored.

Nicholson, who writes privacy policies for businesses, believes a fundamental problem lies in overly complex and incomprehensible privacy policies, as well as the way data is collected and used in the U.S. compared to Europe and other parts of the world. "The U.S. has treated personal information as more of a sales transaction and said that businesses can do what they want with it," he explains. "Europeans use more of a licensing model that focuses on the person owning their data and a business renting it for a specific use." In fact, the European Data Protection Directive and a newer Data Protection Regulation sets tight controls over how data can be collected, stored, and used. It also includes provisions for notifying consumers and obtaining their consent.

Nicholson says some privacy advocates now support a system modeled after Canada's Privacy by Design initiative, which aims to embed privacy protection into new technology and business processes by default. The underlying goal is for consumers to choose the data they make available to companies. With Privacy by Design, data aggregators would cull only the data they need and have permission to use, keep it for only as long as it is immediately valuable, and then purge the data, he explains.

Others have floated the idea of creating information exchanges that pay consumers for the use of their data. Essentially, an individual would manage his or her profile and decide who can purchase the data, what purposes they can use it for, and for how long. "The reality is that we're currently paying in a currency we don't understand because most people don't recognize the actual value of personal information," Nicholson explains.

Cranor says that, in the end, a balance must exist between today's rapidly advancing data-aggregation methods and increasingly elusive privacy. Although a strict opt-in model would almost certainly prove too unwieldy and annoying for consumers, "the entire process must be more transparent," says Cranor. "People must understand what is happening with their data and what choices they are actually making. Only then can we have a system that works well for everyone." C

Further Reading

Leon, P.G., et al.
What do behavioral advertising disclosures communicate to users? Carnegie Mellon University CyLab, April 2, 2012.

Goldfarb, A. and Tucker, C.
Privacy regulation and online advertising, Social Science Research Network, August 4, 2010.

CNN
Joel Stein talks data mining and personal privacy on CNN American Morning, http://www.youtube.com/watch?v=rYiGT4EuE9w, March 10, 2011.

Tucker, C.E.
The economics of advertising and privacy, *International Journal of Industrial Organization 30*, 3, May 2012.

Turow, J.
***The Daily You: How the New Advertising Industry Is Defining Your Identity and Your Worth*, Yale University Press, New Haven, CT, 2012.**

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

Security

Target Cybercriminals, Urges Report

Governments should spend less money on defensive measures, such as antivirus software and firewalls, and more money on "hunting down cybercriminals and throwing them in jail," according to "Measuring the Cost of Cybercrime," a research paper by Ross Anderson, a professor of security engineering at the University of Cambridge, and a team of six researchers from Germany, the Netherlands, the U.K., and the U.S.

The paper, which the researchers believe is the first systematic report on the costs of cybercrime, was presented at the recent 11th Annual Workshop on the Economics of Information Security.

Developed at the request of the U.K. Ministry of Defense, the report provided estimates of the direct costs, indirect costs, and defense costs of different types of cybercrime in the U.K. and the world. It found that for Internet-based crimes like phishing, spam, online scams, hacking, and denial-of-service attacks, the costs of defense are many times higher than the actual losses. Anderson noted the example of a botnet that was responsible for a third of the world's spam in 2010. The botnet is estimated to have earned its owners about U.S. $2.7 million whereas the worldwide costs of spam prevention probably exceeded U.S. $1 billion.

—Jack Rosenberger

Membership | DOI:10.1145/2240236.2240244 Karen A. Frenkel

Broader Horizons

ACM's Committee for Women in Computing (ACM-W) is widening its reach to involve women in industry as well as academia, including community college faculty and students.

ACM-W, which has traditionally been dedicated to supporting ACM's women members, now has new leadership and a broader mission, with Valerie Barr, chair of the computer science department at Union College, succeeding Elaine Weyuker, an AT&T Fellow at Bell Labs. The pair previously served as co-chairs for six months.

"ACM-W has focused on supporting women in computing, and has had particular focus on and responsibility to women members of ACM," says Barr. "While it is important to make sure that women of achievement are recognized within the women in computing community, we want to be sure that women are supported, celebrated, and advocated for within the general computing community. But now we are ready to also take on responsibilities to ensure all ACM members are aware of the role of women in computing, and to help ACM achieve organizational goals for growth and international reach through our work with and for women."

ACM-W needs to reach the 12% of ACM's 104,000 members who are female and make sure they know about ACM programs, like professional development, and non-ACM programs, like the Computer Research Association-Women's mentoring workshops, and special programs for women at conferences like the Grace Hopper Celebration of Women in Computing.

To attract women who are not ACM members, ACM-W will reach out to faculty and students at community colleges. Women in computing at community colleges are a large and untapped market for ACM, says Barr, who notes that community colleges are a less expensive way for women to return to the work force.

ACM-W is also reaching out to students and faculty by awarding scholarships so they can attend conferences.

"We want to be sure that women are supported, celebrated, and advocated for within the general computing community," says Valerie Barr.

In addition, ACM-W aims to create new programs that promote informal interaction between women at conferences so experienced computer scientists can meet and advise newcomers. Unfortunately, few opportunities exist for women at conferences to talk informally, says Barr. Moreover, young faculty do not make attending non-technical, non-peer reviewed conferences a priority because they do not count toward tenure. Therefore, ACM-W will consider how to support networking opportunities for women who attend technical and professional conferences, possibly by arranging birds-of-a-feather meetings.

In terms of conference scholarships, ACM-W has received $20,000 in funding from both Wipro Technologies and Microsoft Research. The Wipro scholarships will provide each awardee $600 and $1,200 for intra- and intercontinental conferences, respectively, for registration and attendance fees. ACM-W plans to give at least 10 intra- and 10 intercontinental awards each year for the next five years. The Microsoft Research scholarships will enable European women to attend conferences, and recipients will receive the same amounts for intra- and intercontinental conference expenses as the Wipro awardees. ACM-W expects to award about 15 Microsoft Research scholarships this year.

"We see the conference scholarship program very much as a pipeline program for young women who make a credible case that going to a conference will help them make decisions about what they're doing in CS," says Barr. "We're trying to encourage B.A.s to go on to graduate school, M.S.s to get Ph.Ds., and early Ph.D. candidates to focus on a research topic and stick with it."

Barr wants ACM-W's activities to be more prominent because it will increase the visibility of women in computing. Barr also wants the rest of the ACM community to take better notice of the contributions women make to the field. "We've done a good job telling women about women," she says, "but now we need to tell everybody else about women."

Based in Manhattan, **Karen A. Frenkel** is a freelance writer and editor specializing in science and technology.

DOI:10.1145/2240236.2240245

Paul Tjia

Emerging Markets
Inside the Hermit Kingdom: IT and Outsourcing in North Korea

A unique perspective on an evolving technology sector.

The outside world's view of North Korea ranges from the fear of nuclear demagoguery, through tales of economic difficulties, to the fun poking of the film *Team America*. Behind these and many other—almost universally negative—projected images of the country, there is another side. Somewhat unexpectedly, the Democratic People's Republic of Korea—to use its official title—has a sizeable IT sector. Some 10,000 professionals work in the field, and many more have IT degrees. They are already engaged in outsourcing contracts for other countries, and keen to expand further.

North Korea's IT Sector

The origins of the local IT sector can be traced back to the 1980s, with the establishment of various IT research organizations and the creation of IT faculties in higher education institutions such as Kim Il Sung University and Kim Chaek University of Technology (the latter having an international collaborative research program with Syracuse University). Over time, several hundred IT "corporations" have emerged, which fall into three main types, all state owned in whole or part.

There are a number of large specialist IT service providers who work both for local and overseas clients. The biggest of these is the Korea Computer Center (KCC). Established in 1990, it has more than 1,000 employees. Like almost all IT firms, it is headquartered in the capital, Pyongyang, but also has regional branches throughout the country and offices overseas that enable it to work for clients in Europe, China, South Korea, and Japan. Despite its export work, KCC's main focus has been the local market and it develops various products, such as Red Star (the North Korean version of Linux), e-learning products, and translation software. KCC also produces games; their version of Go, a popular Asian chess-like game, has won the Go computer games world championship for several years. Similar, but smaller corporations such as the Pyongyang Informatics Centre and Korea Pioneer Technology are employing hundreds of staff.

Having access to a pool of highly technically skilled labor is a key rationale behind the growth of IT outsourcing in North Korea.

Some IT firms have developed from the internal IT departments of large commercial enterprises, such as Unha Corporation or Korea Roksan General Trading Company. As the country's IT sector has become more dynamic, some of these are being spun off as separate ventures, allowing the IT firms to take on a broader scope of clients and IT service activities. Third, there are a number of joint venture IT firms. These include Nosotek (set up with a German entrepreneur) and Hana Electronics (which involves U.K. investment), plus several joint ventures with Chinese business partners.

An IT company in Pyongyang, North Korea. Pyongyang has become a base for IT offshoring.

Historically, North Korea's IT infrastructure has been behind the curve but in recent years the country has been investing in order to catch up. In a typical IT firm, one can therefore see staff using state-of-the-art technology, mostly imported (for example, hardware from Dell, running the latest versions of Windows or Unix).

Internet access is still heavily regulated but a recently laid optical fiber backbone connecting all cities and counties provides for a high-speed network—"Kwangmyong"—utilizing a fairly sophisticated architecture that forms what can be seen as a "national intranet." A nationwide 3G mobile telecommunications network was completed in 2009. Mobile operator Koryolink (a joint venture with Orascom Telecom from Egypt) has more than one million subscribers.

Of course, the core of any IT sector is its human resources. As noted earlier, some 10,000 professionals already have IT careers, and thousands of North Korean youngsters graduate with IT degrees each year, thus creating a large pool of technically qualified staff. In the corporations I have visited, there are always significant proportions of M.S.- and Ph.D.-qualified software engineers, and a surprising number have participated in training courses abroad: most often in China and India, but also in Europe. These organizations are already well versed in quality assurance methods including ISO9001 and CMMI.

An example of the typical skills profile is shown in the accompanying table, which gives the expertise list for Daeyong Corporation, a relatively small start-up based in Pyongyang that is involved in Web and mobile applications development—including Android and Blackberry—for foreign clients.

IT Outsourcing to North Korea

Having access to a pool of highly technically skilled labor is a key rationale behind the growth of IT outsourcing to North Korea. But such labor is available in many parts of Asia. A unique selling point for North Korea is cost. Tariffs asked can be less than U.S. $10 per hour, enabling clients to employ experienced software engineers for just a few hundred dollars per month. This means North Korea undercuts not just India but also later-emerging software locations like China or the Philippines. The country's highly regulated economy is also an advantage in reducing attrition rates—the type of "job-hopping" that can bedevil Indian contracts is pretty much unknown in North Korea.

Some of the outsourcing to Pyongyang-based firms is quite general. At the fairly low-skill end is basic digitization. For example, Dakor—a Swiss joint venture—is conducting data entry work for European research firms and international organizations like the United Nations and the Red Cross. Work involving more skill includes producing Web sites for U.S. and European customers (though in most cases, the end client is unaware of North Korean involvement since this is subcontracted by an intermediary that deals direct with the client). Examples of projects involving more advanced skills include North Korean firms providing programming inputs for enterprise resource planning systems, business process management systems, and e-business applications. One corporation is building a bank management system—based on the tenets of Islamic banking—for a client located in the Middle East.

PHOTOGRAPH BY PAUL TJIA

IT security, such as facial recognition, is an important export commodity for North Korea.

Developers working on a graphics design project.

North Korean firms produce and export high-quality cartoons and animation work.

But there are also emerging specialisms within North Korea's IT export sector. Perhaps not surprisingly, one of these is IT security. North Korea might have an image—warranted or not—of encouraging cyber attacks, but it has invested a lot in technology and expertise to thwart such attacks on its own systems and, more generally, in security. Fingerprint identification products used for access control (and time attendance) have already been exported, and there are other products developed in areas of car license plate identification, and voice/face recognition.

On the lighter side, film has been one of the main forms of state-supported entertainment in the country since the formal division of the Korean peninsula into two states in the 1950s. From this foundation has developed an export production capacity for high-quality cartoons and animation. The specialized state corporation SEK Studio, established in 1957, has more than 1,600 employees, and works for several European film production studios. Other firms have worked on 2D and 3D animation contracts, and this is starting to expand into areas of related capability such as computer graphics and games exports for Wii, iPhone, BlackBerry and other platforms.

Finally, North Korean IT corporations have developed a set of language skills. English is quite widely used but specialisms have developed in Chinese and Japanese alongside (of course) Korean. North Korea is therefore being used as a base for those who wish to have software products or systems translated into East Asian languages. At present, much of this work is nearshoring; that is, coming from clients based in China, Japan, or South Korea. But the potential is there for a wide range of customers who are targeting East Asian markets, which are considered especially relevant to small- and medium-sized clients.

The Challenges and Future of IT Outsourcing

North Korea's IT sector faces a number of challenges, despite its government's eagerness to promote international collaboration. First, and perhaps largest, is the challenge of perceptions. North Korea confronts a difficult mix

PHOTOGRAPHS BY PAUL TJIA

Example of a North Korean IT corporation skill set.

Major Skills	Ansi C/C++, VC++6, VB6 VC++.NET/CLR, C#, VB.NET, ASP.NET, .NET DotNetNuke, Silverlight, Telerik Sitefinity Java, J2EE, Spring, Struts, Hibernate, Swing, JUNIT, ANT JBoss, Apache Tomcat PHP4/5, Joomla, Wordpress, Magento, Drupal Zend Framework, CodeIgniter, CakePHP, Yii, Smarty MySQL, MSSQL, MS Access, PostgreSQL, Oracle Shell script , JavaScript, VBScript jQuery, extjs, dhtmlx suite HTML4.01, HTML5, XHTML, CSS2.0, CSS3.0 XML, XSLT, Web Services Flash, Flash AS3, Flex
Experienced Techniques	phpNuke, phpBB, Symfony Axis2, Velocity, POI, JSF Microsoft Speech API Google Android Phone, iPhone J2ME, Brew, Perl
Specialized IT Areas	Web Applications, E-Commerce, Blogs, CRM, Social Networking General Windows utilities, Windows/Linux Networking Reverse Engineering Security and Cryptography
Languages	Korean, English, Japanese, Chinese

of ignorance and suspicion. It is not well known yet as an outsourcing destination, particularly outside neighboring countries. All the local IT firms have a Web site, but in most cases it is only visible on the national intranet, not externally. This is particularly difficult when trying to compete for attention with firms in other emerging Asian IT locations like Vietnam or Bangladesh, which often have a strong Web presence.

Suspicion of all things North Korean and a "guilt by association" that fails to differentiate the political from the commercial creates a serious barrier for some clients. Perhaps as a result, Europe—where historical relations with the two Koreas are different—may be a more feasible source of partners than the U.S. Certainly the overhanging geopolitical aspects must be acknowledged: there are political tensions, the Korean War has never officially ended, and there are U.S.-imposed sanctions.

Alongside these factors are more mundane barriers—potential clients in the West lack contact lists or even basic information about North Korea. In practice, the country is quite easy to visit, but it is time consuming to obtain a visa, and Pyongyang has few direct international flights and, as noted, limited Internet connectivity. To help circumvent these difficulties, some North Korean IT firms operate offices abroad, such as in China—there are examples in Beijing, Dandong, and Dalian all staffed by Korean software engineers—through which outsourcing contracts can be undertaken.

Because of the challenges, those who have set up outsourcing arrangements typically do so by working gradually up a "trust and knowledge curve." This begins with a visit by the client to Pyongyang, holding discussions with potential partner organizations until a suitable one is identified. The next step may be a set of staff exchange visits—a "getting to know you" type of familiarization activity that allows Western clients to understand what North Korea has to offer, and allows North Korean managers to understand what the client needs. Then some small pilot projects are contracted. These may well begin with North Korean staff being brought over to the client site to complete some tightly specified and non-mission-critical IT tasks. Only once this onsite experience has been worked through will the clients move to a more offshore mode of working with their North Korean partners.

Does some of this sound familiar? It should. This is where India was approximately 30 years ago—a seeming obscure location for IT work, accompanied by images of animal-drawn carts and corrupt bureaucrats. A few brave U.S. firms stuck a toe in the water with some small onsite "body shopping" projects and built up slowly from there. Today, India's software industry is a multibillion-dollar juggernaut on which Fortune 500 and other firms rely for portions of their IT services.

North Korea is not going to follow the same meteoric IT trajectory as India: the challenges it faces are different and more serious. But nor is it a hypothetical or unthinkable destination. Pyongyang already is a base for IT offshoring, the country does have some unique selling points, and is serious about its aspiration to grow this area. You cannot escape the broader geopolitics when dealing with North Korea but, for those who believe nations are reformed by engagement rather than conflict, building IT relationships could be part of a broader path to more general rapprochement that also helps access a low-cost, high-skill talent pool. C

Further Reading

Carmel, E. and Tjia, P.
Offshoring Information Technology.
Cambridge University Press, 2009.

Hayes, P., Bruce, S., and Mardon, D.
***North Korea's Digital Transformation*, Nautilus Institute, 2011; http://nautilus.org/napsnet/napsnet-policy-forum/north-koreas-digital-transformation-implications-for-north-korea-policy/.**

Korea Computer Center
http://naenara.com.kp/en/kcc/.

Mansourov, A.Y.
***North Korea on the Cusp of Digital Transformation*, Nautilus Institute, 2011; http://www.nautilus.org/wp-content/uploads/2011/12/DPRK_Digital_Transformation.pdf.**

North Korea Tech Web portal
http://www.northkoreatech.org/.

Park, S.-J.
Software and animation in North Korea. In *Europe—North Korea. Between Humanitarianism and Business?*, M. Park, B. Seliger, S.-J. Park, Eds., LIT Verlag, 2010.

Paul Tjia (paul@gpic.nl) is the founder of GPI Consultancy, an independent Dutch consultancy firm in the field of global IT sourcing that organizes study tours to countries such as North Korea and offers support on offshoring strategy, country and partner selection, offshore transition, and cultural training.

DOI:10.1145/2240236.2240246

Fred G. Martin

Education

Will Massive Open Online Courses Change How We Teach?

Sharing recent experiences with an online course.

In the last decade, the Creative Commons philosophy of freely sharing information and the pervasiveness of the Internet have created many new opportunities for teaching and learning. MIT OpenCourseWare spearheaded the sharing of high-quality, university-level courses. While these materials were not originally designed for individuals engaged in self-study, approximately half of OCW's traffic is from these users.[6] Recently the use of learning management systems (LMSs), such as the proprietary Blackboard or open-source Moodle software, has become ubiquitous.

In typical use, LMSs support the structure of conventional courses in an online setting. Lectures and reading material are assigned, homework is scheduled, and discussions are facilitated at regular intervals. As in conventional coursework, classes are usually closed communities, with students registering (and paying) for credit-bearing coursework.

One of the first initiatives to bring together open course philosophy and LMSs was David Cormier's "Massive Open Online Course," or MOOC. In his vision, "Although it may share in some of the conventions of an ordinary course, such as a predefined timeline and weekly topics for consideration, a MOOC generally carries no fees, no prerequisites other than Internet access and interest, no predefined expectations for participation, and no formal accreditation."[9]

This idea—albeit in a more conventional course structure—exploded into the public consciousness with the massive open artificial intelligence (AI) course developed and conducted by Stanford faculty Sebastian Thrun and Peter Norvig last fall. Announced in the summer of 2011, the course received wide publicity, and attracted about 160,000 registered students by its launch in October 2011. Approximately 23,000 students completed the 10-week course.[8] I was one of the 23,000—along with a cohort of 16 students, both graduate and undergraduate, from my home institution (the computer science department at the University of Massachusetts Lowell).

Since the fall AI 2011 course, there has been much activity in this space. Thrun has set up a for-profit company, Udacity, to extend his initial work; Stanford and others are running courses using Coursera; MIT created MITx and is partnering with Harvard on edX; and there are other initiatives.[4] The remainder of this column describes my experiences taking the fall 2011 course alongside my students and facilitating their learning. This is followed by some reflections. It seems likely this new breed of MOOCs will have impact on education at the university level, particularly for technical majors such as computer science.

The Fall 2011 Stanford AI-Class.com

The Stanford course consisted of weekly lectures—two or three 45-minute topics that were broken up into 15 or 20 short videos. Most of the individual videos had embedded questions (multiple-choice or fill-in-the-value). At the end of each mini-lesson, the video image transformed into a Web form where students fill in answers. Already logged in, the class server graded the students immediately. After submitting, students were shown an explanation video.

The lectures themselves were inspired by Khan Academy's casual, teacher-sitting-by-your-side approach. Occasionally, Thrun and Norvig trained the camera at themselves, but the core content was conducted with the camera aimed at a sheet of paper, with Thrun or Norvig talking and writing in real time. I found this format relaxing and engaging; I liked seeing equations written out with verbal narration in sync.

There were weekly problem sets with the same format. These homeworks tracked the lecture material closely. The course included a midterm and a final with the same format as the homework. If you worked through and understood the problems embedded in the lectures, the homework assignments were straightforward. The homework, midterm, and final each had hard deadlines. The server backend kept track of students' scores on the homework assignments, the midterm, and the final, which all counted toward the student's "grade"—a ranking within the active student cohort.

On the Course

Thrun and Norvig were strong teachers. They thought through excellent ways of explaining the ideas and quizzing the in-lecture comprehension checks. They often brought fun props or showed research projects in the video recordings.

Thrun and Norvig were only a week or so ahead of the course delivery, and they paid close attention to students' progress. There was a lot of activity on the Web forums. They recorded several "office hours," where students submitted questions and voted on their favorite ones, and then they picked questions and answered them on camera. In this way, the course was like a typical class—it was not "canned." Thrun's and Norvig's enthusiasm was infectious. Collectively, the real-time nature of the experience made it a lot like a well-taught conventional course.

My Role at UMass Lowell

Students registered for my department's regular AI course, which requires a project. They knew when signing up that I expected them to complete both the Stanford course and a directed project. As mentioned earlier, I had 16 students. We met once weekly for a 75-minute session in a roundtable format. We talked about the Stanford material after each week's assignment was already due. Because of this, I did not have to present the course material in a lecture format. When we met, most of my students had worked through the lectures and the homework. So I did not have to explain things to students for the first time. Instead, we used in-class time for conversations about material that people found confusing or disagreed upon. We had some great discussions over the course of the semester.

A similar approach was developed by Day and Foley in their HCI course at Georgia Tech.[2] They recorded Web lectures, and then used classroom time for hands-on learning activities. Daphne Koller, a colleague of Thrun's at Stanford (and founder of Coursera), has called this "the flipped classroom." She reported higher-than-usual attendance in her Stanford courses taught this way: "We can focus precious classroom time on more interactive problem-solving activities that achieve deeper understanding—and foster creativity."[7]

What Does it Mean?

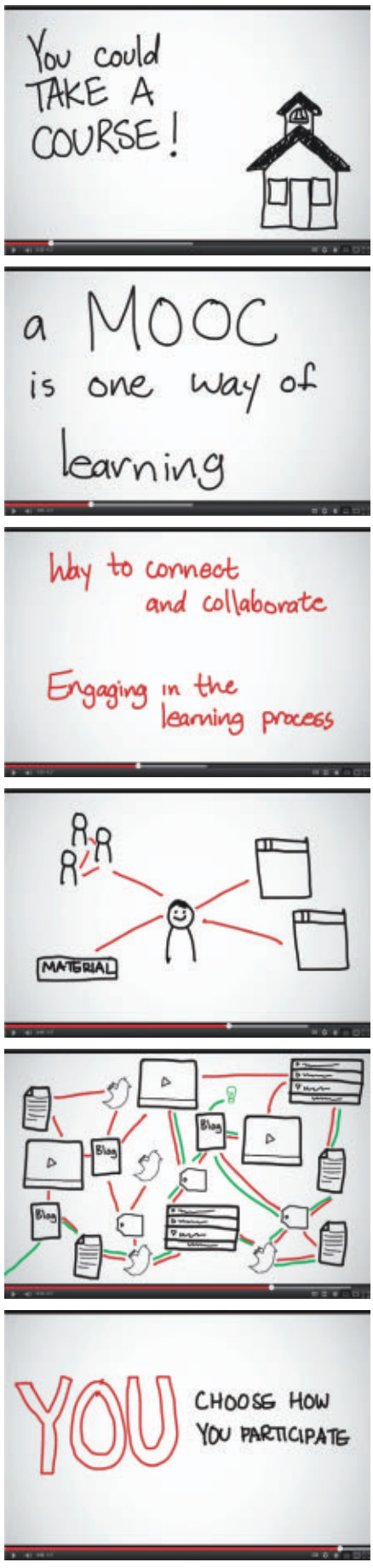
The success of the fall AI course and the bloom of new ones this spring and summer puts real pressure on conventional, lecture-and-test university instruction. Thrun is quoted in several reports as noting that attendance at his face-to-face AI course at Stanford in the fall dropped precipitously. From the 200 registered, after a few weeks, only 30 continued to attend.[8] But this really speaks to the failure to have the in-person time deliver anything different from a lecture. In many ways, the carefully crafted online lectures, peppered with probing questions that are autograded for correctness and then explained further, are indeed an improvement over a conventional lecture.

There are many initiatives to improve the quality of face-to-face time in lectures. When used creatively, clickers can be a valuable modification. But more fundamentally, active learning approaches hold much more promise.

Robert Beichner has developed a classroom approach called "SCALE UP" for active learning in the classroom. His work started in physics education, but years of development and collaboration broadened it to many fields, including the sciences, engineering, and the humanities.[5] In SCALE-UP, faculty engage students in a structured activities and problem-solving during classroom time. Students work in teams of three, and faculty mingle with them, engaging them in discussions. (The SCALE-UP acronym has had several meanings, including "Student-Centered Active Learning Environment for Undergraduate Programs.")

Eric Mazur, also from the physics education community, has developed a related approach that he calls "peer instruction," in which students work in small groups to answer questions posed in lectures. Like Beichner, Mazur is active in disseminating this method.

However, dissemination and adoption are big challenges. These approaches require substantial new development of the problems and ac-



FRAMES FROM DAVID CORMIER'S YOUTUBE VIDEO "WHAT IS A MOOC?"

tivities with which students are to be engaged in the classroom, and teachers must give up their carefully crafted lecture presentations.

Also, teachers need to be protected from low student evaluation scores. Mazur and others have reported that students give lower evaluations in courses with active learning—even when the evidence shows they have learned more.[1,3] Students have grown up with conventional lecture teaching, and just like anyone else, they are resistant to change.

Beyond this, faculty must participate in these active learning approaches as learners, so they understand how to facilitate them as instructors. In my case, in my graduate training I learned how discussion-oriented seminar courses are conducted, so it was natural for me to facilitate the same with my small group of "flipped classroom" AI students.

Conclusion

When Thrun was promoting the fall 2011 online AI course, his Twitter feed included some bold claims: @aiclass: "Advanced students will complete the same homework and exams as Stanford students. So the courses will be equal in rigor."—September 28, 2011

The fall 2011 course for matriculated Stanford students included programming assignments, and the online one did not. This was a clear shortcoming. But the new Udacity courses include programming. Most of my students got a lot out of the fall Stanford course—and our weekly discussion sections made a difference. But the weaker students struggled, and a few strong students were bored. This makes me wonder about the large-scale applicability of the MOOC format. We need to be able to support students who are still learning how to learn, and also challenge our best students. The MOOC concept does not even attempt to address the role of a small, research-oriented project-based course. When we individually mentor each student on his or her own ideas, we are doing something that can never be performed by an autograder.

Part of the excitement around MOOCs is about their potential to change education's cost equation—put a great course online once, and run it unattended many times. But part of the fun of the fall AI course was that Thrun and Norvig were right there with us, and that we were a large cohort of students there with them.

Thrun also asserted: @aiclass: "Amazing we can probably offer a Master's degree of Stanford quality for FREE. HOW COOL IS THAT?"—September 23, 2011

As we know, the modern university is a much larger ecosystem than its collection of courses. Among many other things, students derive great value from being in close contact with their peers, participating in leadership opportunities across campus, and being part of our research labs. It may well be that this new breed of MOOC is a decent replacement for an average, large-sized lecture course. But this is a low bar.

In our drive for efficiency, we must aspire to higher than this. At least, we can use MOOCs to create a successful flipped classroom. We can use our "precious classroom time" for meaningful conversations. As Mazur and Beicher have demonstrated, this can be done even in large lectures by having students work in small groups.

At best, we can really add value, by being teachers. We can individually debug students' thinking, mentor them in project work, and honestly be enthusiastic when they excel. C

References

1. Crouch, C.H. and Mazur, E. Peer instruction: Ten years of experience and results. *Am. J. Phys. 69* (Sept. 2001).
2. Day, J. and Foley, J. Evaluating a Web lecture intervention in a human-computer interaction course. *IEEE Transactions on Education 49*, 4 (Nov. 2006).
3. Fagen, A.P., Crouch, C.H., and Mazur, E. Peer instruction: Results from a range of classrooms. *The Physics Teacher 40* (2002).
4. Fox, A. and Patterson, D. Crossing the software education chasm. *Commun. ACM 55*, 5 (May 2012), 44–49.
5. Gaffney, J.D.H. et al. Scaling up education reform. *Journal of College Science Teaching 37*, 5 (May/June 2008), 48–53.
6. *Giving Knowledge for Free: The Emergence of Open Educational Resources*, Organisation for Economic Co-operation and Development, 2007.
7. Koller, D. Death knell for the lecture: Technology as a passport to personalized education. *New York Times* (Dec. 5, 2011).
8. Lewin, T. Instruction for masses knocks down campus walls. *New York Times* (Mar. 4, 2012).
9. McAuley, A., Stewart, B., Siemens, G., and Cormier, D. *The MOOC Model for Digital Practice.* 2010; http://www.elearnspace.org/Articles/MOOC_Final.pdf.

Fred G. Martin (fredm@cs.uml.edu) is an associate professor of computer science and associate dean in the College of Sciences at the University of Massachusetts Lowell.

DOI:10.1145/2240236.2240247

danah boyd

Privacy and Security
The Politics of "Real Names"

Power, context, and control in networked publics.

ILLUSTRATION BY BRIAN GREENBERG/ANDRIJ BORYS ASSOCIATES

Facebook's terms of service explicitly require its users to provide their "real names and information." Indeed, the norm among many Facebook users is to provide a first and last name that appears to be genuine. Thus, when Google+ launched in the summer of 2011, it tried to emulate Facebook by requiring that new users provide similar credentials. Many early adopters responded by providing commonly used nicknames, pseudonyms, and stage names. Google, determined to ensure compliance, began expelling people who did not abide by the "real names" requirements. They ejected high-profile geeks, including Limor Fried and Blake Ross for failing to use their real names; they threatened to eject Violet Blue, a well-known sex educator and columnist.

The digerati responded with outrage, angry with Google for its totalitarian approach. The "nymwars," as they were called, triggered a passionate debate among bloggers and journalists about the very essence of anonymity and pseudonymity.[2,3,6] Under pressure, Google relented, restoring users accounts and trying to calm the storm without apologizing. Meanwhile, Google's chairman Eric Schmidt publicly explained the "real names" policy is important because Google Plus is intended to be an identity service.[1]

While the furor over "real names" has subsided—and Google now supports pseudonymity—key questions about the role of identity, privacy, and control remain. Why did people respond with outrage over Google while accepting Facebook? Do "real names" policies actually encourage the social dynamics that people assume they engender? Why did people talk about "real names" policies as a privacy issue? This column explores these issues, highlighting the challenges involved in designing sociotechnical systems.

Facebook vs. Google: Norms, Values, and Enforcement

At Harvard, Facebook's launch signaled a safe, intimate alternative to the popular social network sites. People provided their names because they saw the site as an extension of campus life. As the site's popularity grew, new users adopted the norms and practices of early adopters. The Facebook norms were seen as operating in stark contrast to MySpace, where people commonly used pseudonyms to address concerns about safety. Unlike MySpace, Facebook appeared secure and private.

As Facebook spread beyond college campuses, not all new users embraced the "real names" norm. During the course of my research, I found that lateteen adopters were far less likely to use their given name. Yet, although Facebook required compliance, it tended not to actively—or at least, publicly—enforce its policy.

Today, part of Facebook's astronomi-

cal value stems from the quality and quantity of information it has about its users. While it is unlikely that Mark Zuckerberg designed Facebook to be an identity service, that is what it has become. Google's competitive move is explicitly an identity play. Yet, rather than creating a value proposition in which users would naturally share their real names, they made it a requirement.

Larry Lessig argues that four forces regulate social systems: the market, the law, social norms, and technology or architecture.[4] Social norms drove the "real names" culture of Facebook, but Google's approach was purely driven by the market and reinforced by corporate policies and technology. Their failure to create the conditions in which newcomers felt comfortable sharing their names—and their choice to restrict commonly used pseudonyms—resulted in a backlash. Rather than designing an ecosystem in which social norms worked to their favor, their choice to punish dissidents undermined any goodwill that early adopters had toward the service.

Implicit Assumptions about "Real Names"

Although some companies implement "real names" policies for business reasons, many designers believe such policies are necessary to encourage healthy interactions in online communities. Implicit is the notion that in "real life," people have to use their "real names" so why shouldn't they be required to do so online? Yet, how people use names in unmediated interactions is by no means similar to what happens online.[5]

When someone walks into a cafe, they do reveal certain aspects of themselves while obfuscating other aspects of their identity. Through their bodies, they disclose information about their gender, age, and race. Through fashion and body language, people convey information about their sexuality, socioeconomic status, religion, ethnicity, and tastes. This information is not always precise and, throughout history, people have gone a long way to obscure what is revealed. While people often possess documents in their wallets that convey their names, this is not how most people initiate interactions.

The practice of sharing one's name is embedded in rituals of relationship building. People do not share their names with every person they encounter. Rather, names are offered as an introductory gesture in specific situations to signal politeness and openness.

Privacy is not about restricting information; it is about revealing appropriate information in a given context.

While the revelation of a person's name may link them to their family or signal information about their socioeconomic position, most names in a Western context provide little additional information beyond what is already conveyed through the presence of the individual. As such, they simply serve as an identifier for people to use when addressing one another. Online, the stakes are different.

Online, there are no bodies. By default, people are identified through IP addresses. Thus, it is common to lead with a textual identifier. In the days of Usenet and IRC, that identifier was typically a nickname or a handle, a username, or an email address. With Facebook and Google Plus, people are expected to use their names.

The power of search also shifts the dynamics. Although it is possible for wizards in Hogwarts to scream the equivalent of "grep" into the ether and uncover others' location, background information, and relationships, this is not something mere mortals can do in everyday life. Until the Internet arrived. Today, information about people can be easily accessed with just a few keystrokes. Through search, the curious can gain access to a plethora of information, often taken out of context. Without the Internet, inquiring about someone takes effort and provokes questions. Asking around often requires addressing a common response, "Why do you want to know?" Yet, search engines empower the curious to obtain—and misinterpret—information without any social consequences. That shifts how people relate online.

Privacy and Names

Accountability is commonly raised as one of the reasons behind which people should provide identifiable information in online settings. When people prefer not to share their names, they are assumed to have something to hide. Many people claim people are better behaved and more "honest" when their identifying information is available. While there is no data that convincingly supports or refutes this, it is important to note that both Facebook and face-to-face settings continue to be rife with meanness and cruelty.

Even if we collectively value accountability, accountability is more than an avenue for punishment; accountability is about creating the social context in which people can negotiate the social conditions of appropriate behavior. Most social norms are regulated through incentive mechanisms, not punishment. Punishment—and, thus, the need to identify someone outside of the mediated context—is really a last-resort mechanism. The levers for accountability change by social context, but accountability is best when it is rooted in the exchange.

There are people who abuse other's trust, violate social norms, or purposefully obscure themselves in order to engage in misdeeds. This is not just a problem online. But most people who engage in lightweight obfuscation are not trying to deceive. Instead, they are trying to achieve privacy in public environments.

Wanting privacy is not akin to wanting to be a hermit. Just because someone wants to share information does not mean they want to give up on privacy. When people seek privacy, they are looking to have some form of control over a social situation. To achieve that control, people must have agency and they must have enough information to properly assess the social context. Privacy is not about restricting information; it is about revealing appropriate information in a given context.

People feel as though their privacy has been violated when their agency has

been undermined or when information about a particular social context has been obscured in ways that subvert people's ability to make an informed decision about what to reveal. This is why people feel so disempowered by technological moves where they feel as though they cannot properly manage the social situation.

In unmediated contexts, choosing when or how to reveal one's name allows people to meaningfully control a social situation. For example, when a barista asks a customer for her name, it is common for the customer to provide only her first name. There are also customers who provide a nickname or a fake name when asked for such information, particularly if their name is obscure, hard to pronounce, or overly identifiable. The customs involved in sharing one's name differ around the world and across different social contexts. In some settings, it is common to only provide one's last name (for example, "Mr. Smith"). In other settings, people identify themselves solely in relationship to another person (for example, "Bobby's father"). People interpret a social situation and share their name based on how comfortable they are and what they think is appropriate.

When people are expected to lead with their names, their power to control a social situation is undermined. Power shifts. The observer, armed with a search engine and identifiable information, has greater control over the social situation than the person presenting information about themselves. The loss of control is precisely why such situations feel so public. Yet, ironically, the sites that promise privacy and control are often those that demand users to reveal their names.

Who Is in Control?

Battles over identity, anonymity, pseudonymity, and "real names" are not over. New systems are regularly developed and users continue to struggle with how to navigate information disclosure. What is at stake is not simply a matter of technology; identity in online spaces is a complex interplay of design, business, and social issues. There is also no way to simply graft what people are doing online onto what they might do offline; networked technology shifts how people engage with one another. Thus, it is important to move away from offline metaphors in order to address the complexity of people's mediated interactions.

The "real names" debate goes beyond identification technologies and economic interests. Regardless of the business implications, the issue of whether or not to mandate "real names" is fundamentally one of power and control. To what degree do designers want to hold power over their users versus empower them to develop social norms? To what degree do companies want to maintain control over their systems versus enable users to have control over their self-presentation and actions?

These are complex socio-technical questions with no clear technical or policy solution. Furthermore, even though design plays a significant role in shaping how people engage with new technologies, it is the interplay between a system and its users that determines how it will play out in the wild. Design decisions can inform social practices, but they cannot determine them. As with all complex systems, control is not in the hands of any individual actor—designer, user, engineer, or policymaker—but rather the product of the socio-technical ecosystem. Those who lack control within this ecosystem resist attempts by others to assert control. Thus, finding a stabilized solution requires engaging with the ecosystem as a whole. C

References

1. Banks, E. Eric Schmidt: If you don't want to use your real name, don't use Google+. *Mashable* (Aug. 28, 2011); http://mashable.com/2011/08/28/google-plus-identity-service/.
2. Dash, Anil. If your Website's full of a------, it's your fault. *A Blog About Making Culture* (July 20, 2011); http://bitly.com/JAyktS.
3. Fake, C. Anonymity and pseudonyms in social software. *Caterina.net* (July 26, 2011); http://caterina.net/wp-archives/88.
4. Lessig, L. *Code: And Other Laws of Cyberspace*. Basic Books, New York, 1999.
5. Madrigal, A. Why Facebook and Google's concept of 'real names' is revolutionary. *Atlantic Monthly*, (Aug. 5, 2011); http://www.theatlantic.com/technology/archive/2011/08/why-facebook-and-googles-concept-of-real-names-is-revolutionary/243171/.
6. Skud. Preliminary results of my survey of suspended Google+ accounts. *InfoTropism* (July 25, 2011); http://infotrope.net/2011/07/25/preliminary-results-of-my-survey-of-suspended-google-accounts/.

danah boyd (danah@danah.org) is a senior researcher at Microsoft Research, a research assistant professor in Media, Culture, and Communication at New York University, a visiting researcher at Harvard Law School, a fellow at Harvard's Berkman Center, and an adjunct associate professor at the University of New South Wales.

Calendar of Events

August 17–19
International Conference on Security of Internet of Things,
Amritapuri, India,
Sponsored: SIGSAC,
Contact: Rangan Venkat,
Email: venkat@amrita.edu

August 21–24
Information Interaction in Context: 2012,
Nijmegen, Netherlands,
Contact: Kraaij Wessel,
Email: wessel.kraaij@tno.nl

August 26–29
Advances in Social Networks Analysis and Mining 2012,
Istanbul, Turkey,
Sponsored: SIGMOD,
Contact: Can Fazli,
Email: canf@cs.bilkent.edu.tr

August 27–29
12th International Conference on Quality Software,
Xi'an China,
Contact: Zhou Xingshe,
Email: zhouxs@nwpu.edu.cn

August 28–31
Asia Pacific Conference on Computer Human Interaction,
Matue-City, Shimane Japan,
Sponsored: SIGCHI,
Contact: Kentaro Go,
Email: go@yamanashi.ac.jp

September 3–6
8th International ICST Conference on Security and Privacy in Communication Networks,
Padua, Italy,
Contact: Mauro Conti,
Email: conti@di.uniroma1.it

September 4–6
24th International Teletraffic Congress,
Krakow, Poland,
Contact: Papir Zdazisaaw,
Email: papir@kt.agh.edu.pl

September 4–7
ACM Symposium on Document Engineering,
Paris, France,
Sponsored: SIGWEB,
Contact: Cyril Concolato,
Email: cyril.concolato@enst.fr

September 6–8
The International Workshop on Internationalisation of Products and Systems,
Bangalore, Kanatak India,
Contact: Apala Lahiri Chavan,
Email: apala@humanfactors.com

DOI:10.1145/2240236.2240248

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

Kode Vicious
A System Is Not a Product

Stopping to smell the code before wasting time reentering configuration data.

Every once in a while, I come across a piece of good code and like to take a moment to recognize this fact, if only to keep my blood pressure low before my yearly medical checkup. The first such piece of code to catch my eye was clocksource.h in Linux. Linux interfaces with hardware clocks, such as the crystal on a motherboard, through a set of structures that are put together like a set of Russian dolls.

At the very center is the cyclecounter, a very simple abstraction that returns the current counter from an underlying piece of hardware. A cyclecounter knows nothing about the current time, time zone, or anything else; it knows only what the register in a piece of hardware is when asked about it. The cyclecounter has two pieces of state that help translate cycles into nanoseconds but nothing else. The next doll out is the timecounter. A timecounter contains a cyclecounter and raises the level of abstraction to the level of monotonically increasing time, measured in nanoseconds. On top of these structures are others that eventually give the system enough abstraction to know what the time of day is, and so forth.

So, what is so great about this code? Well, two things: first, it is well structured, in that it is built from small components that can cooperate without a layering violation; second, it is written and documented in a style that is clear and clean enough that I was able on first reading to understand how it worked. The comments and structure for the cyclecounter shown in the figure here give you a flavor of what makes me so happy to read this code.

An example of good code.

```
/**
 * struct cyclecounter - hardware abstraction for a free running counter
 *    Provides completely state-free accessors to the underlying hardware.
 *    Depending on which hardware it reads, the cycle counter may wrap
 *    around quickly. Locking rules (if necessary) have to be defined
 *    by the implementor and user of specific instances of this API.
 *
 * @read:         returns the current cycle value
 * @mask:          bitmask for two's complement
 *           subtraction of non 64 bit counters,
 *           see CLOCKSOURCE_MASK() helper macro
 * @mult:          cycle to nanosecond multiplier
 * @shift:        cycle to nanosecond divisor (power of two)
 */
struct cyclecounter {
    cycle_t (*read)(const struct cyclecounter *cc);
    cycle_t mask;
    u32 mult;
    u32 shift;
};
```

I think you can see why I like this code, but just in case you can't, let me be more specific. The code is well laid out, nicely indented, and has variables that are short, yet readable. There are no Bouncy Caps or_very_long_names_that_read_like_sentences. The comment is long enough to describe not just what the structure is, but how it is used, and it even mentions what has to be done if multiple threads need to access one of these structures simultaneously. If only all code were this well documented! You can read more of this code online at http://lxr.free-electrons.com/source/include/linux/clocksource.h.

My other example of good code requires more explanation, so I am going to reserve it for a future column. After all, I don't want to waste the calming effect all in the same issue.

Dear KV,

After spending the last year buying and deploying a set of monitoring systems in my company's production network, we hit our first bug in the manufacturer's system software. After we reported the bug, the manufacturer asked us to upgrade the systems to the latest release. It turns out the upgrade process requires us to reset the devices to their factory defaults, losing all our configuration information on each system and requiring a person to reenter all of the configuration data after the upgrade.

At first, we thought this might be something required only for this particular upgrade, which crosses a major revision, but we have since learned we must reenter our configuration data each and every time we upgrade the software on these systems. I suppose we should have asked this question before we bought the systems, but it just never occurred to us that anyone would sell a box purporting to act as an appliance that could not be easily upgraded in the field. One of the other members of my group suggested we just return the boxes and ask for our money back, but, depressingly enough, these are the best systems we have found for network monitoring.

Down on Upgrades

Dear Down,

It seems that what you thought was a product—that is, a set of components thoughtfully put together by people who care about their customers—was actually just a collection of parts that under normal circumstances worked well enough to get by. Unfortunately, the number of companies that think about what will happen after version 1.0 of their systems is quite small.

I have been fortunate—or perhaps I should say the sales critters I have dealt with in the past 10 or so years have been fortunate—not to come across too many systems that act in the way you describe. The idea that the manufacturer would require a user or systems administrator to reenter already-saved data after an upgrade is stupid, ludicrous, and a bunch of other words that my editors just are not going to allow in this publication.

Even in the worst-designed products—and I have used enough of those—there is usually some bit of perl that takes the old configuration data and turns it into something that is mostly valid for the latest revision.

As a matter of fact, many years ago I worked on a networking switch project and the first thing the systems team (which was responsible for getting a reasonable operating system and applications onto the box) did was to come up with a way to field-upgrade the system. Anyone who has ever configured a switch or router knows you do not just toss out the configuration on an upgrade.

The sad part is that doing this correctly just is not that difficult. Most embedded systems now use stripped-down Unix systems such as Linux or the BSDs, all of which store their configurations in well-known files. Granted, this is not the nicest way to store configuration data, because it tends to be a bit scattered, but it is not that difficult to write a script that can handle differences between the versions and reconcile them. On FreeBSD there is etcupdate, and Linux has etc-update and dispatch-conf. In a properly designed system the configuration would likely be stored in a simple database or an XML file, both of which are field-upgradable with fairly simple scripts.

These sorts of issues are what differentiate an *appliance* that can be deployed and maintained with little human interference from a *system* that has to be diddled constantly to keep it happy. It is a sad fact that many programmers and engineers do not think much of appliances and are more likely to think their users should "man up" and spend their time making up for the original designer's lack of forethought.

I still remember one of the first all-digital stereo systems I ever saw. It was designed around a Sun workstation and cost on the order of $15,000. It was obvious from the moment you looked at the controls that little or no thought had been put into what people really wanted out of a sound system. What most people want out of a stereo is good-quality sound with a minimum of button pushing to get what they want. What the system I saw presented was a lot of button pushing for about the same quality of sound I could get out of an amplifier and a CD player. If the user interface was horrific, it was nothing compared with the system performance. The box crashed three times before I finally walked out of the store. The one thing I got from that experience was an understanding that systems and products are not the same thing.

A *system* is a collection of components put together to do a job. A *product* is a system that has been designed and built to make the work of carrying out the job smooth and natural to the user. I can certainly cobble together software to rip, store, and play my CDs on a computer; that is a system, whereas an iPhone is a product. When I upgrade my phone, I do not reenter my data. If I did, the product would have died at the first revision. More likely, Steve Jobs would have taken someone's head off if he had been told the upgrade path required reentering data. Given how complex modern appliances are—and let's face it, your TV probably has an Ethernet jack—it is clear that people have thought about and solved this problem.

The real issue is with the people who design these devices. Somehow the fact an appliance is going to be hanging out with a bunch of computers—for example, in a colocation—makes it acceptable for the implementers to make their box look more like an open source desktop, which will be diddled by an experienced IT person or the users themselves. It is a practice that really needs to stop, if only because I don't want everyone to wind up bald, like me. My hair didn't fall out, I pulled it out!

KV

Related articles on queue.acm.org

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

A Conversation with Chris DiBona
http://queue.acm.org/detail.cfm?id=945130

Closed Source Fights Back
Greg Lehey
http://queue.acm.org/detail.cfm?id=945126

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

DOI:10.1145/2240236.2240249

Chris Forman, Avi Goldfarb, and Shane Greenstein

Economic and Business Dimensions

The Internet Is Everywhere, but the Payoff Is Not

Examining the uneven patterns of Internet economics.

AMID THE RAPID diffusion of the Internet in the 1990s, many media commentators, investment analysts, and policymakers hoped the Internet would erase geographic and socioeconomic boundaries and improve the comparative economic performance in less prosperous regions.

The optimism had some merit. The economic tide was rising during this time, with everyone's fortunes rising. The best-sellers *The Death of Distance*[1] and *The World Is Flat*[4] were just two of the books that espoused that rosy view. The "productivity paradox"—the question of whether ubiquitous use of computers would ever produce economic growth—disappeared among professional economists. Between 1995 and 2000 the economic tide was rising, as wages increased by 20% on average across the U.S.

Note the words "on average." What effect did the growth of the commercial Internet have on the economic performance of different locations in the U.S. economy? Our analysis published in the *American Economic Review* suggests the Internet is widespread, but its economic payoffs are not even.[3]

Misplaced Optimism

We conducted a multiyear study to document how businesses adopted, deployed, and improved the Internet from 1995 to 2000, a period of rapid initial investment by business. In prior research,[2] we learned to look carefully at the specific type of investment. We distinguished between rapidly diffusing email and Web browsing and more advanced applications. Such advanced Internet applications were enabling productivity increases by lowering costs of communication between suppliers and customers over long distances, and these applications required skilled labor to implement and operate. Diffusion of these applications shaped productivity for many enterprises.

We were nevertheless struck by the persistence of an earlier perception: the Internet was almost everywhere but the pattern was uneven. Most large business establishments were using email and browsing by the late 1990s. However, the more we looked, the more we could see that advanced commercial uses of the Internet such as inventory management systems and online database sharing were found more often in larger cities than in smaller ones. They were also found more in data-heavy industries such as finance and wholesale distribution than they were in manufacturing, mining, and social services. At one level this makes sense—advanced Internet applications are more likely to be found in electronic parts supply and sophisticated financial operations than in landscaping firms and nursing homes. Nevertheless, there was a disparity: more prosperous cities were home to sophisticated companies that knew how to best take advantage of the new technology, and those companies (and cities) received more benefits from Internet use than other cities. There might well be payoffs from the Internet, but they were not evenly shared.

The Payoff Puzzle

Our more recent study combined information on business Internet usage with U.S. national-level wage data. We studied from the period 1995 to 2000 using several large datasets, including the Quarterly Census of Employment and Wages that gives county-level information on average weekly wages and employment, and the Harte Hanks Market Intelligence Computer Intelligence Technology Database of survey information about how firms use the Internet. In total, we included relevant data from over 85,000 private establishments with more than 100 employees each.

We found that business adoption of Internet technologies correlated with wage and employment growth in only 163 of the over 3,000 counties in the U.S. All of these counties had populations above 150,000 and were in the top quarter of income and edu-

cational achievement before 1995. Between 1995 and 2000, they showed a 28% average increase in wages, compared with a 20% increase in other counties, where the Internet appears to not have had much impact on economic performance (see the accompanying figure).

In short, the Internet appears to have exacerbated regional wage inequality, explaining over half the difference in wage growth between the 6% of counties that were already well-off and all other counties.

Why did the Internet make such big waves in these few areas? Three explanations are possible:

- Big cities have needed communications infrastructure and are home to firms capable of making capital investments that allow the Internet to enhance what such sophisticated firms are already doing.
- A phenomenon called "skill-biased technical change" might be under way, in which new technologies raise the wages only of skilled workers, while unskilled workers cannot use the new technology and do not receive the necessary training to benefit from its deployment through wage gains.
- Cities have denser labor markets, and better communication between supply and demand. Programmers with skills in the language of mainframes, such as COBOL, and the language of the Web, such as HMTL and Perl lived in (or moved to) big cities, where more firms would be looking for such workers. Similarly, a range of other inputs that would make Internet adoption more valuable is more available in cities.

The Internet may allow firms in rural Iowa to reach new customers on Wall Street but it also allows Wall Street banks to reach investors in rural Iowa.

Each explanation is plausible, and probably explains a piece of the story. However, existing data does not allow us to distinguish between the explanations, and so the payoff puzzle remains.

Policy

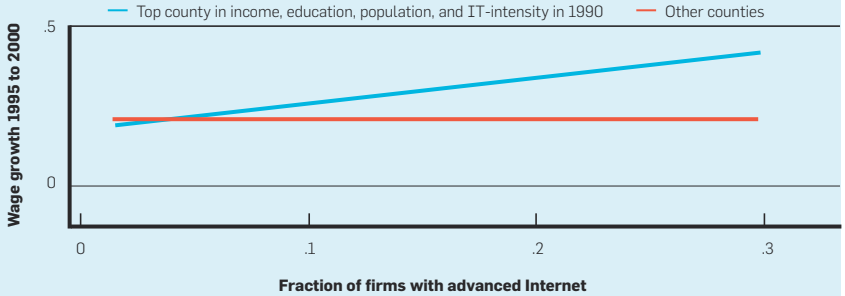
The Internet has been widely diffused and used: in this, our results echo the popular optimism. Also, the benefits are not limited to the locations that dominate supply of equipment and software, such as Boston, Seattle, New York, and Silicon Valley.

However, in contradiction to popular optimism, our data does not show any evidence of improvement in the *comparative* economic performance of isolated locations or less dense locations. The Internet may allow firms in rural Iowa to reach new customers on Wall Street but it also allows Wall Street banks to reach investors in rural Iowa. Our study shows that benefits from Internet adoption such as increased wages are, on balance, more likely to show up in New York City than in rural Iowa. The predicted "leveling" effects of the Internet have not materialized, and it is doubtful that short-run investment in Internet infrastructure (such as extending access to broadband) will generate immediate economic benefits for areas that have otherwise low capital investment, inadequate labor skills, and market conditions.

Conclusion

These results have important implications for public policy. Many lawmakers have argued for government to subsidize the expansion of the Internet into poor and isolated regions. For example, the 2009 economic stimulus package allocated over $7 billion for broadband expansion in underserved areas, under the assumption that creating such infrastructure will raise wages and income. Our results suggest such Internet infrastructure investments by themselves are unlikely to raise wages in poor and isolated regions, at least in the short run. Such investments might be justified to strengthen education, increase civic engagement, or promote health and safety. And, over 20 to 30 years expanding Internet access might lead to more widespread economic gains. However, there is little evidence to suggest a short-term payoff in the form of increased local wages.

Advanced Internet investment and wage growth by county type.



References

1. Cairncross, F. *The Death of Distance*. Harvard University Press, Cambridge, MA, 1997.
2. Forman, C., Goldfarb, A., and Greenstein, S. How did location affect adoption of the commercial Internet? Global village vs. urban leadership. *Journal of Urban Economics 58*, 3 (Mar. 2005), 389–420.
3. Forman, C., Goldfarb, A., and Greenstein, S. The Internet and local wages: A puzzle. *American Economic Review 102*, 1 (Jan. 2012), 556–575.
4. Friedman, T.L. *The World is Flat: A Brief History of the Twenty-First Century*. Farrar, Straus, and Giroux, New York, 2005.

Chris Forman (chris.forman@mgt.gatech.edu) is the Robert and Stevie Schmidt associate professor of IT management in the College of Management at Georgia Institute of Technology, Atlanta, GA.

Avi Goldfarb (agoldfarb@rotman.utoronto.ca) is an associate professor of marketing at the Joseph L. Rotman School of Management at the University of Toronto.

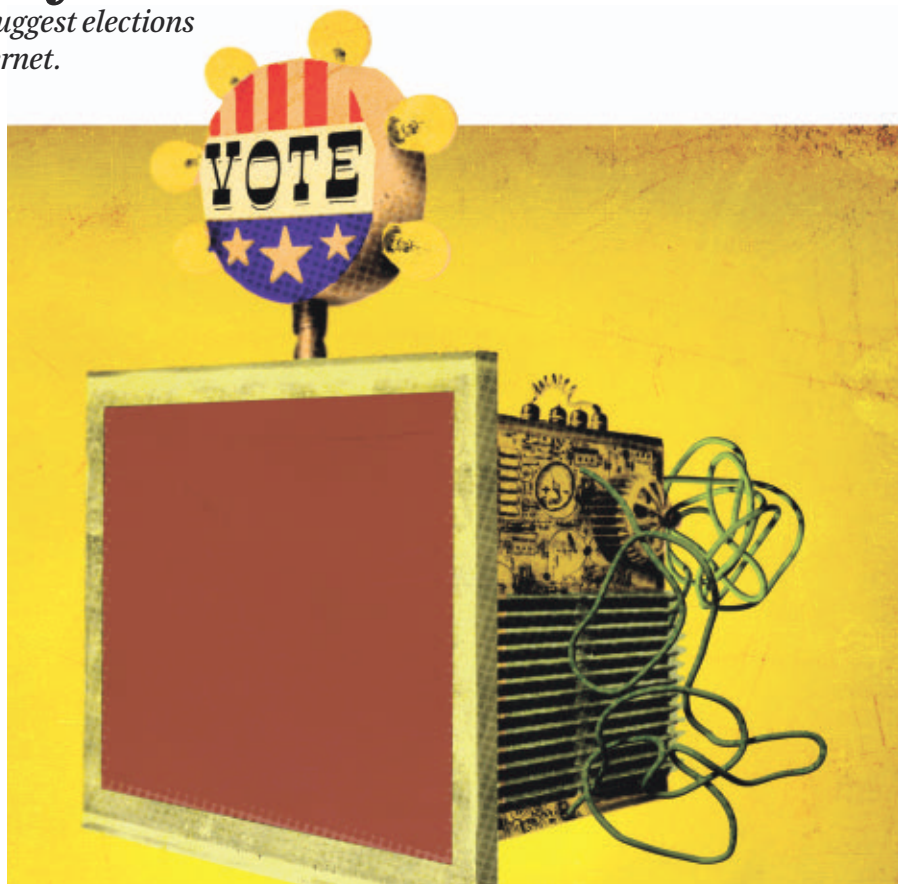
Shane Greenstein (greenstein@kellogg.northwestern.edu) is the Kellogg Chair of Information Technology and Professor in the Kellogg School of Management at Northwestern University, Evanston, IL.

DOI:10.1145/2240236.2240251

Kai A. Olsen and Hans Fredrik Nordhaug

Viewpoint
Internet Elections: Unsafe in Any Home?

Experiences with electronic voting suggest elections should not be conducted via the Internet.



Several countries have experimented with Internet elections. While voters in Estonia could use the Internet in their 2007 national election for parliament,[3,7] other countries have offered this option for local elections only. The argument in favor of allowing voting over the Internet is that this may increase voter participation, which has dropped in many countries in recent years. Another key argument is that Internet elections can increase access for people with disabilities. Furthermore, some people feel that Internet elections are an inevitable case of allowing democracy to enter the Internet world along with shopping, banking, and many other applications.

One argument against voting on the Internet is that it cannot guarantee the anonymity that a voting booth can provide. This can lead to coercion and the buying and selling of votes. Even more serious is the risk that someone may tamper with the voting or the results. This risk is aggravated by the fact that voting could be performed from a computer that has malware installed.

In order to overcome such obstacles the Ministry of Local Government in Norway launched a $40 million project ("e-valg 2011") in 2009 to design an electronic voting system to be used in the 2011 local elections. The system is based on experience from other countries, and is comparable to the Estonian system. Experts from academia, research institutions, and industry were engaged in the development of a secure system, and to check the proposed solutions. To prevent pressure for voting against one's wish, the system allowed repeated voting. The Internet option was closed one day prior to the election, voters always had the option to cast their vote at a polling station, and the voter's last vote would always override any previous votes. To counter malware and other risks, an advanced cryptographic system was employed.[2] An important part of this is a coding system designed to prevent vote tampering.

This system uses codes to identify each party participating in the election.[4] These codes are presented on the back of a "voter's card," which is mailed to everyone who has the right to vote. After casting the ballot on the Internet using a home or office computer, voters receive a confirmation text message. This message includes a code the voter can compare to the card in order to check if the vote was regis-

ILLUSTRATION BY RICHARD BORGE

tered for the correct party. Since these codes only exist on the paper card and are particular to each voter, malware would have to break into the central server in order to generate the codes. Tests also showed that many voters do check that the code is correct. Thus, the ministry determined that it would require a "large-scale conspiration and unreasonable amounts of money" to break the system.

Malware Exploiting Voters

As we will describe, however, there are many ways to violate such a system. Malware can discard votes or change votes at the same time as sending users the correct confirmation code. In order to demonstrate these possibilities, we designed a malware system of our own. Although we have assumed the server part of the e-voting system is secure, our malware exploits the most vulnerable part of the voting system: the voter. After the voter has chosen a certain party, the official voting system presents a page with the name of the party and the user is asked to confirm by clicking a send button. Our version is similar, but it also asks for the (secret) party code: "Please confirm your selection by typing in the code for this party. You will find the code on the back of your voter's card." In a test simulating voting on paper, we tested this on 158 college students, including 25 IT students. None of the students found any faults with our system. In addition, over 400 high school students tested an online version. All of these students typed in the party code when required by the system. This was despite the fact that all participants, both college and high school students, were shown an animation made by the ministry that explains the e-voting system and stresses the correct use of the party code as a final manual check. However, in all other situations PIN codes are supposed to be typed into some computer system. Even the e-voting system has a part where a PIN received by a text message is to be typed in during the identification process. Thus, while our malware uses the code as users anticipate, it is the official system that applies the code in an unfamiliar manner.

Once it has the code, the malware can easily send the correct confirmation to the voter and then discard or change the vote. In the latter case, the voter will receive another text message, this time from the ministry, with a different code (for the party that the malware has chosen). With malware also present on the smartphone, this second text message can be discarded. Or, more simply, the user can be warned that additional messages may be forthcoming due to some communication problems, and to "please ignore these." A ministry document on security objectives stresses the risk of such malware: "The insecurity of browsers and operating systems on the client platform will invariably make it possible to subversively install malicious software."[1]

We have demonstrated that voters are actually victims of the user interface.

However, those with malicious intent may opt for more straightforward solutions. Introducing a fake URL is one possibility. Many users will find the link through other Web pages, such as community pages. These sites have been hacked before and can be hacked again. A false Web site will also receive identification data from the voter, and will, at least in the Norwegian system, be able to change the phone number for the confirmation message. Even more simply, a villain could send an email message to voters before the election with a "vote here" URL. By targeting the recipients, such as groups the villain feels vote unsatisfactorily, it is extremely simple to make an e-voting system that mimics the original system, asks for the party code, sends a confirmation message to the voter, and then discards the vote. It took us just a few hours to create such a system.

While someone would certainly notice a large-scale attack, a small-scale attack, perhaps in one community only, could go unnoticed. Even if there is disclosure, what should be done? Should voters be asked to vote again or should the election be considered invalid? Creating such chaos may even be the main intent of the villain. Even in such a case, there may not be an easy "undo." As well as changing your vote, malware may also know how you voted. This information, which most people consider private, could be used for blackmail or embarrassment.

In theory, the Norwegian e-voting system is safe. According to the developers, the risks involved can be expressed mathematically. However, this is based on the condition that voters do what they are supposed to do. We have demonstrated that voters are actually victims of the user interface. We argue that voting systems are particularly vulnerable. A conscientious voter who participates in all elections will in many countries vote once every two years. This is not frequent enough to get any practice or routine with an e-voting system. In any case, there may also be modifications in the interface from one election to another. So, while methods such as those presented here may, in principle, be used to obtain secret codes from Internet bank users, in practice, most of us would become suspicious if the system broke its expected pattern; for example, if it asked for additional codes. Furthermore, if someone breaks into your bank account, you would at least notice money has disappeared. A changed or discarded vote, however, may never be discovered.

Estonia requires citizens to insert their nationwide ID card into a card reader connected to the home computer to vote on the Internet, and then offer PIN codes as a further proof of identification. This may increase security on the home computer, but the risk of malware or a false election site is still the same.

Postal voting is another option offered in many countries, and some states have this as the only alternative.[5] It offers the same possibility of voting from one's home as the Internet. Theoretically it can be as vulnerable as Internet voting, but it would take more resources to mount an attack. Also, while the Internet offers anonymity to the villains, this may not be the case

Trustworthy design of an electoral system is critical for democracy.

for those who try to interfere with a postal voting system.

Postal voting, as any system that allows a voter to cast a vote outside a voting booth, still has the disadvantages that voters can be coerced or paid to vote in a certain way. The possibility of repeated voting could reduce this problem. By going to the polling place after giving the Internet or postal vote, one has the opportunity to vote again. However, a patriarch of a closely controlled family could easily restrict his daughter's movements on the final day of the election, just as he could control their Internet voting. For those buying votes it is just a small calculated risk that the seller of a vote will turn up on Election Day.

Repeated voting on the Internet may not offer any solution. Votes can still be bought, not by requiring how people vote, but by taking control over their ID codes. This allows the buyer to vote on their behalf. However, the Estonian solution with an ID card would make it more difficult to hand this over to others. This is especially the case when the card is also used for other purposes.

In Isaac Asimov's science fiction story *Franchise* (1955), the all-encompassing supercomputer Multivac chose Norman Muller as the "Voter of the Year." In this electronic democracy, a single person was selected to represent all voters. Based on the answers to a set of questions to Norman, Multivac determined the results of the election. Norman was proud that the U.S. citizens had, through him, "exercised once again their free, untrammeled franchise." This is not exactly Internet voting, but the two systems do have something in common: it is impossible for non-experts to verify they work correctly. The old system with paper ballots may be inefficient, but it does allow any voter to understand how it works. This is the case also for postal voting. Trust in such a system is more direct than with any e-voting application.

Conclusion

In an October 2011 *Communications* Inside Risks column, Carsten Schürman argued for modernizing the Danish democratic process.[6] He stressed the importance of listening to the voices of scientists and other specialists when designing new systems, but, as we have seen, this did not work in Norway. While he praised the European e-voting initiatives, he was skeptical regarding Internet voting—"for which there are still more open problems than solved ones." Perhaps voting is one task that should not be moved to the Internet? Trustworthy design of an electoral system is critical for democracy; this is a place where no risks, neither practical nor theoretical, can be tolerated. The advantage of running a computer system that is to be used sparingly is also dubious and, as we have seen, creates additional risks since users have no routine. It is also reasonable to believe that electoral participation does not depend on the voting system alone. Perhaps it has something to do with politics? C

References

1. *e-Vote 2011 Security Objectives.* Ministry of Local Government; http://www.regjeringen.no/upload/KRD/Kampanjer/valgportal/e-valg/tekniskdok/Security_Objectives_v2.pdf.
2. Gjøsteen, K. Analysis of an Internet voting protocol, 2011; http://eprint.iacr.org/2010/380.
3. Meagher, S. When personal computers are transformed into ballot boxes: How Internet elections in Estonia comply with the United Nations international covenant on civil and political rights. *American University International Law Review 23*, 2 (Feb. 2009), 349–386.
4. Nestås, L.H. and Hole, K.J. Building and maintaining trust in Internet voting. *IEEE Computer 45*, 5 (May 2012).
5. Qvortrup, M. First past the postman: Voting by mail in comparative perspective. *Political Quarterly 76*, 3 (Fall 2005), 414–419.
6. Schürmann, C. Modernizing the Danish democratic process. *Commun. ACM 54*, 10 (Oct. 2011), 27–29.
7. Trechsel, A. et al. *Internet Voting in Estonia. A Comparative Analysis of Four Elections since 2005.* Report for the Council of Europe, European University Institute, Robert Schuman Centre for Advanced Studies, 2010.

Kai A. Olsen (kai.olsen@himolde.no) is a professor of informatics at Molde University College and at the University of Bergen, Norway.

Hans Fredrik Nordhaug (hans.f.nordhaug@himolde.no) is an associate professor and study counselor in the department of informatics at Molde University College, Norway.

DOI:10.1145/2240236.2240250

Neil McBride

Viewpoint

The Ethics of Software Engineering Should be an Ethics for the Client

Viewing software engineering as a communicative art in which client engagement is essential.

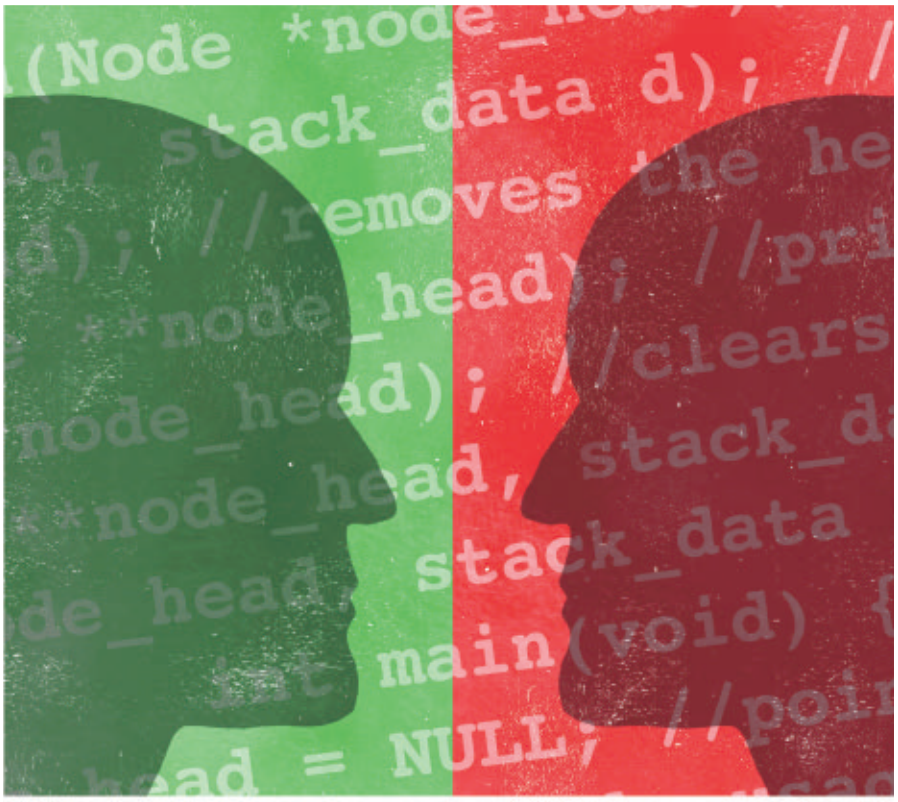


ILLUSTRATION BY GLUEKIT

Advances in parallel computing mean Internet traffic can be monitored in real time and illicit activity automatically detected. Single-chip autonomous motes with wireless communications and environmental sensors can be scattered in a natural or medical environment. Routine sequencing of individual's genetic code will soon be commercially viable and will require new software applications to support it. How are software engineers to understand the social and ethical challenges of such advances?

In the last 15 years, the environment and practice of software engineering has changed radically. Commercial applications are mostly Web-based, often globally accessible and subject to emergent and unexpected effects arising from their use on a global scale. Not only are the products different, but the practice is different. Agile methods dominate. Rapid systems delivery and continuous contact with the client is expected. Software engineers have to engage with increasingly complex social and ethical environments.

And yet, currently, the only support we can offer the software engineer is a code of ethics, which is 15 years old, parochial in its scope, and often obvious in its content. It recommends behavior such as having good documentation, rejecting bribery, and refraining from the use of illegal software. Its orientation is primarily negative and deals with injunctions that most practitioners would subscribe to anyhow.[1]

The developing nature of software engineering requires not a revision of an ailing code but a revolution in ethical thinking that acknowledges the purpose and practice of software engineering. Computer systems are designed and implemented to support human purposeful activity. Whether the software is concerned with student enrollment, customer relationship management, or hospital administration, its success lies in the extent to which it enables others to engage in activities directed toward a goal. The software engineer is a service provider,

supporting the processes and activities that result in human flourishing.

Hence the moral behavior underpinning the practice of software engineering requires, firstly, an ethics of service, focusing on the relationship with the client, the client's employees and customers. Secondly, the software engineer must be concerned about the ethics of the client. The closer relationship of IT with the client and the user demands a more active involvement with the ethical environment within which the software operates.

A service-oriented, client-oriented software engineering ethics requires outward-looking engagement with complex social and moral climates. How is this to be achieved when the comfort and certainty of codes cannot be relied upon? What is the alternative and how can software engineers become reflective practitioners?

Service Ethics

A service-oriented ethics will focus on relationships and the complex network of social interactions that is the social context of the software. How does the outsourcer relate to the client? Who are the ultimate users of the software? How do those users relate to others involved in the purposeful activity that is supported by the software? A customer relationship management system, for example, serves the purposeful activity of the managers, service staff, and customers of a business. Ethical issues may arise from the complex social environment, the culture, the business practice, and the customer expectations. This social complexity will defeat any code of conduct. We cannot engage in ethics at arms length, nor conduct relationships by remembering reams of rules.

A service-oriented ethics is a matter of character not codes. We develop character through practice that enables us to reflect on complex development environments and sensitively identify and tackle most ethical issues that arise. Character is expressed in virtues such as courage, patience, honesty, and empathy. For example, patience is developed through practice in a situation where communication with the client is slow and time is needed to understand what the client wants. Courage is developed through standing up to unreasonable demands, and raising uncomfortable issues about how software will affect users. The virtue of honesty must be practiced so that we do not overpromise and are transparent about the limitations of the systems we build. The characteristic of humility will help the software engineer to listen sensitively to the client but not assume the client knows better.

But whether we, as software engineers, are prepared to engage in character development rather than rule adherence will depend on our own motivation. We must reflect on our own goals. What is driving us? What meaning do we apply to the work we do? Is our work just driven by a focus on the technology we use? Are we just concerned with earning money?

If we are driven by external goods in terms of financial rewards or technical advancement, there may be little motivation to engage with ethical concerns. But a shift toward internal rewards may encourage us to reflect on our goals in our professional life. An emphasis on internal rewards will come from looking outward and considering our connection with community—local and global—and finding fulfillment in our contribution to human flourishing.

Domain Ethics

At the heart of a service ethics is the development of empathy—the virtue of putting ourselves in other's shoes, of learning about their environment and concerns. This requires immersion in the client's world, and seeing ourselves not as technicians, but as servants of human purposeful activity developing an ethics-in-use. To do this we must understand the moral climate of the client: the ethical issues involved in the practice of medicine, of nursing, of computer gaming, of accountancy, of energy provision. We cannot assume our own professional codes of conduct are adequate or even relevant to the areas we are serving.

The software needs to support the moral environment within which the client works, not the moral climate of the software engineer. The value and benefits of the software are obtained when it is used in the client's environment. That is not to say we do not question the ethical framework of the client, or offer suggestions for improvement. But how can we comment on something we have not engaged with or understood? Although some of the ethical situations faced by our clients will be foreign to us, this does not mean we should not pursue an understanding of our client's predicaments that will enable us to deliver more ethically sensitive and appropriate software.

Having understood the issues in the domain of practice, does the software engineer learn another code of conduct or retreat to his own better-rehearsed software engineering code

How can software engineers become reflective practitioners?

of conduct? Attempts to apply the client's domain's code of ethics may require an immersion in a culture and practice that is impracticable for the software engineering.

However, both software engineers and their clients develop characteristic virtues through practice that influence and inform their decision making when faced with ethical choices. Fundamental virtues such as patience, kindness, compassion, and empathy are common to all areas of human practice. That is not to say that new virtues, relevant to the client's practice, will not be recognized and applied as we learn about the client.

So the development of our own moral characteristics through the learning and practice of virtues will grow closer connections with the client and their activity. And combined with a knowledge of the ethical difficulties encountered by the client will help us to sympathize with the client and at the least avoid writing crass moral assumptions into our software that are culturally foreign and morally jarring.

The Way Forward

So how are we to do this? The danger is we replace a catalogue of rules with a catalogue of virtues. An understanding of the idea of virtues, the importance of character and the relation to actions is important. Studying the virtue of courage will help us identify when courage is needed, when we need to overcome fear and selfish concerns to stand up for a client, to point out an ethical malfunction, or to whistle-blow. It will also enable us to understand the boundaries, to tell the difference between considered courage and rash foolishness. A skills-based approach such as that advocated by Huff et al.[2,3] is helpful, but we do not want to descend into debates about hierarchies of virtues or taxonomic exercises. We must look inward, outward, and all around.

Inward reflection on who we are, what our drives us, what we consider important, will help us become aware of the social and ethical assumptions we make and take for granted. What we consider good and bad behavior will affect how we respond to social situations and ethical dilemmas.

Outward engagement with the world in which the software acts will help the software engineer develop empathy and a feeling for the ethical climate in which the software runs. Familiarity with the technical issues associated with the field of application is extended to the ethical issues. Outward engagement will involve searching for examples of good practice, involvement with mentors and others who model virtues, and creating narratives about the service we engage in and the purposeful activity we support.

We must understand the nature of the domains we engage with and the facts concerning the ethical problems associated with them.

All-around vision will involve learning to connect with issues that matter to people. The software engineer who spends all his time buried in Python manuals and playing Call of Duty 4 is hardly likely to develop wisdom and ethical understanding. Software engineers should be encouraged to connect with the wider world. Reading the literature of the great religions, modern and ancient authors, engaging in political debate, reading about advances in science and global problems will help develop all-around vision.

A set of static rules should be replaced by a dynamic process. Starting by learning about the ethical problems and difficulties the client faces, we move on to consider what character traits, what virtues would be important in this domain. We can then look for examples and stories of expressions of those virtues in that domain or others. An emphasis on narratives will lead us to ask what would a virtuous person do?

We can then reflect on the implications for software engineering and improve the design of the product and the delivery of the service accordingly. Such a process enables us to practice, learn from that practice, and grow in wisdom.

Conclusion

The software engineer who considers himself only a technician, who rejects social and ethical engagement, claiming "it's a matter for lawyers and doctors" is a destructive force and a burden on the client. We should consider software engineering as a communicative art where engagement with the client is at its heart and an understanding of the tasks, goals, and moral concerns of the client leads to the development of software that is appropriate to the client's needs.

But we cannot expect a written code to be the prime motivator of professional moral character any more than a written constitution guarantees good government. Indeed the written code of ethics may lead to complacency, ignorance, and the neglecting of moral reflection. Codes of ethics can provide a smokescreen for deserting personal responsibility.

Hence we must understand the nature of the domains we engage with and the facts concerning ethical problems associated with them. And we must develop virtues that will help us to use our understanding of the application domain in making ethical decisions. The nature of the domains we engage in will change, but the human characteristics we need such as empathy, patience, honesty, compassion will not change.

Rules can be learned and forgotten quickly, becoming a virtuous software engineer takes time and practice. Wisdom develops over a lifetime. C

References

1. Harris, C.E. The good engineer: Giving virtue its due in engineering ethics. *Science and Engineering Ethics 14* (2008), 155–164.
2. Huff, C.W., Barnard, L., and Frey, W. Good computing: A pedagogically focused model of virtue in the practice of computing (part 1). *Journal of Information, Communication & Ethics in Society 6*, 3 (2008), 246–278.
3. Huff, C.W., Barnard, L., and Frey, W. Good computing: A pedagogically focused model of virtue in the practice of computing (part 2). *Journal of Information, Communication & Ethics in Society 6*, 4 (2008), 284–316.

Neil McBride (nkm@dmu.ac.uk) is a Reader in Information Technology Management in the Centre for Computing and Social Responsibility at De Montfort University, Leicester, U.K.

DOI:10.1145/2240236.2240254

Article development led by acmqueue
queue.acm.org

An open standard that enables software-defined networking.

BY THOMAS A. LIMONCELLI

OpenFlow: A Radical New Idea in Networking

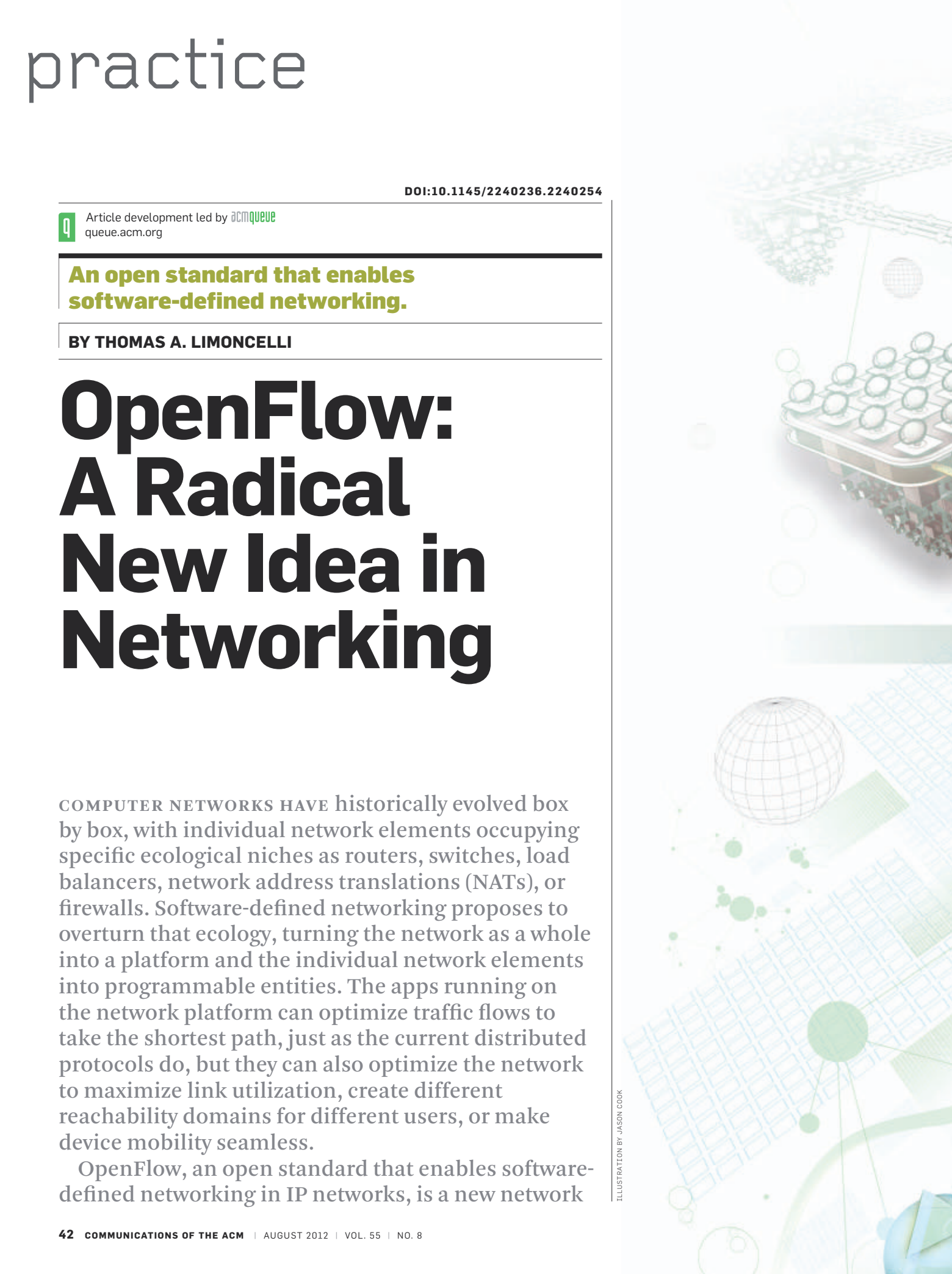
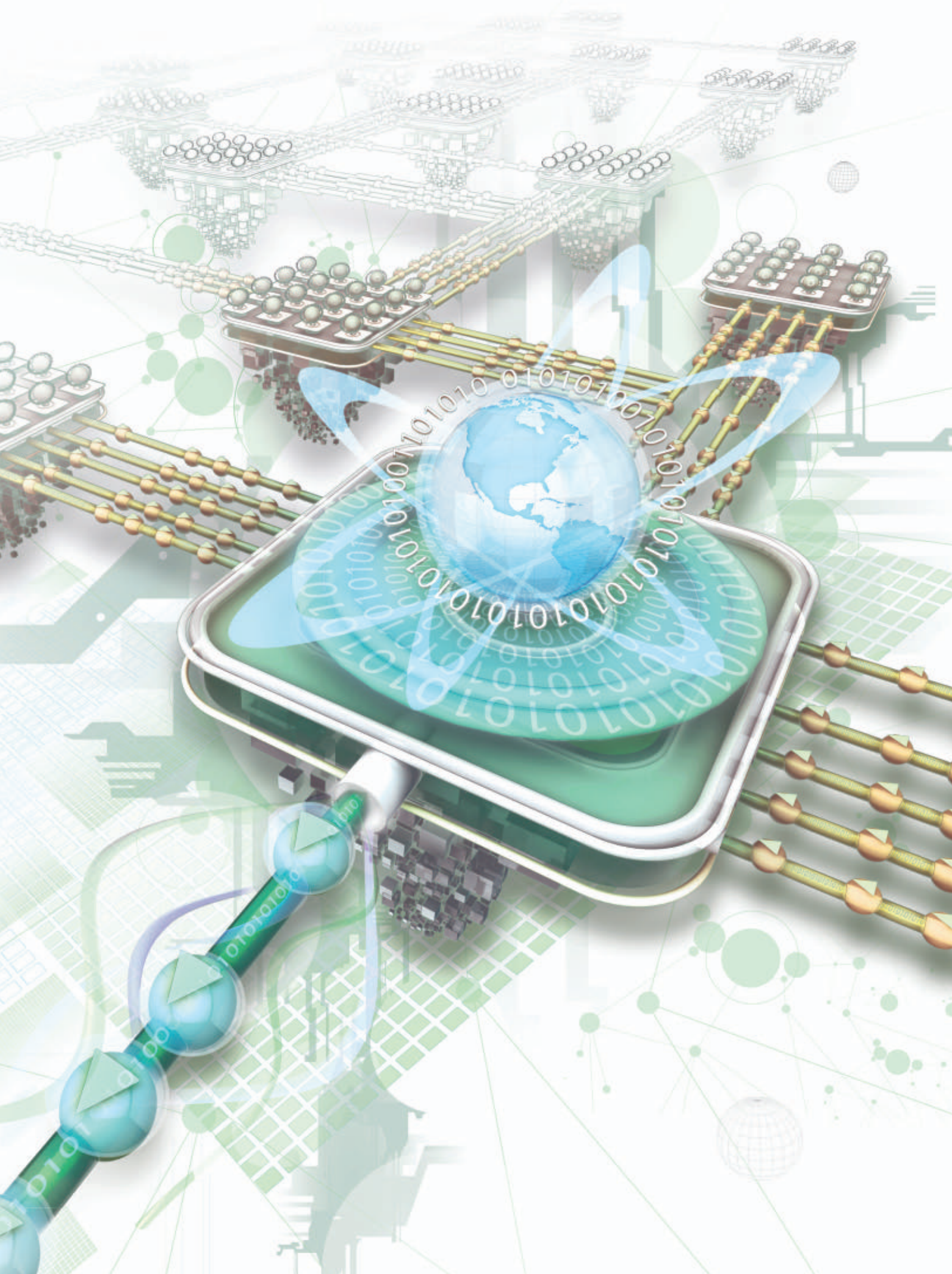


ILLUSTRATION BY JASON COOK

COMPUTER NETWORKS HAVE historically evolved box by box, with individual network elements occupying specific ecological niches as routers, switches, load balancers, network address translations (NATs), or firewalls. Software-defined networking proposes to overturn that ecology, turning the network as a whole into a platform and the individual network elements into programmable entities. The apps running on the network platform can optimize traffic flows to take the shortest path, just as the current distributed protocols do, but they can also optimize the network to maximize link utilization, create different reachability domains for different users, or make device mobility seamless.

OpenFlow, an open standard that enables software-defined networking in IP networks, is a new network



technology that will enable many new applications and new ways of managing networks.

'Real' Examples

Here are three real, though somewhat fictionalized, applications:

Example 1: Bandwidth Management. A typical wide area network has 30%–60% utilization; it must "reserve" bandwidth for "burst" times. Using OpenFlow, however, a system was developed in which internal application systems (consumers) that need bulk data transfer could use the spare bandwidth. Typical uses include daily replication of datasets, database backups, and the bulk transmission of logs. Consumers register the source, destination, and quantity of data to be transferred with a central service. The service does various calculations and sends the results to the routers so they know how to forward this bulk data when links are otherwise unused. Communication between the applications and the central service is bidirectional: applications inform the service of their needs, the service replies when the bandwidth is available, and the application informs the service when it is done. Meanwhile, the routers feed real-time usage information to the central service.

As a result, the network has 90%–95% utilization. This is unheard of in the industry. The CIO is excited, but the CFO is the one who is really impressed. The competition is paying almost 50% more in capital expenditure (CAPEX) and operating expenditure (OPEX) for the same network capacity.

This example is based on what Google is doing today, as described by Urs Hoelzle, a Google Fellow and senior vice president of technical infrastructure, in his keynote address at the 2012 Open Networking Summit.[1]

Example 2: Tenantized Networking. A large virtual-machine hosting company has a network management problem. Its service is "tenantized," meaning each customer is isolated from the others like tenants in a building. As a virtual machine migrates from one physical host to another, the virtual LAN (VLAN) segments must be reconfigured. Inside the physical machine is a software-emulated network that connects the virtual machines. There must be careful coordination between the physical VLANs and the software-emulated world. The problem is that connections can be misconfigured by human error, and the boundaries between tenants are... tenuous.

OpenFlow allows new features to be added to the VM management system so that it communicates with the network infrastructure as virtual and physical machines are added and changed. This application ensures each tenant is isolated from the others no matter how the VMs migrate or where the physical machines are located. This separation is programmed end-to-end and verifiable.

This example is based on presentations such as the one delivered at the 2012 Open Networking Summit by Geng Lin, CTO, Networking Business, Dell Inc.[3]

Example 3: Game Server. Some college students set up a Quake server running on a spare laptop as part of a closed competition. Latency to this server is important not only to gameplay, but also to fairness. Because these are poor graduate students, the server runs on a spare laptop. During the competition someone picks up the laptop and unplugs it from the wired Ethernet. It joins the Wi-Fi as expected. The person then walks with the laptop all the way to the cafeteria and returns an hour later. During this path the laptop changes IP address four times, and bandwidth varies from 100Mbps on wired Ethernet, to 11Mbps on an 802.11b connection in the computer science building, to 54Mbps on the 802.11g connection in the cafeteria. Nevertheless, the game competition continues without interruption.

Using OpenFlow, the graduate students wrote an application so that no matter what subnetwork the laptop is moved to, the IP address will not change. Also, in the event of network congestion, traffic to and from the game server will receive higher priority: the routers will try to drop other traffic before the game-related traffic, thus ensuring smooth gameplay for everyone. The software on the laptop is unmodified; all the "smarts" are in the network. The competition is a big success.

This example is fictional but loosely based on the recipients of the Best Demo award at SIGCOMM (Special Interest Group on Data Communications) 2008.[4] More-serious examples of wireless uses of OpenFlow were detailed by Sachin Katti, assistant professor of electrical engineering and computer science at Stanford University, in a presentation at the 2012 Open Networking Summit.[2]

A Network Routing System That Lets You Write Apps

OpenFlow is either the most innovative new idea in networking, or it is a radical step backward—a devolution. It is a new idea because it changes the fundamental way we think about network-routing architecture and paves the way for new opportunities and applications. It is a devolution because it is a radical simplification—the kind that results in previously unheard of new possibilities.

A comparison can be made to the history of smartphones. Before the iPhone, there was no app store where nearly anyone could publish an app. If you had a phone that could run applications at all, then each app was individually negotiated with each carrier. Vetting was intense. What if an app were to send the wrong bit out the antenna and take down the entire phone system? (Why do we have a phone system where one wrong bit could take it down?) The process was expensive, slow, and highly political.

Some fictional (but not *that* fictional) examples:

- "We're sorry but your tic-tac-toe game doesn't fit the 'c'est la mode' that we feel our company represents."
- "Yes, we would love to have your calorie tracker on our phone, but we require a few critical changes. Most importantly the colors must match our corporate logo."
- "We regret to inform you that we are not interested in having your game run on our smartphone. First, we can't imagine why people would want to play a game on their phones. Our phones are for serious people. Second, it would drain the battery if people used their phones for anything other than phone calls. Third, we find the idea of tossing birds at pigs to be rather distasteful. Our customers

would never be interested."

Network equipment vendors do not currently permit apps. An app that fulfills a niche market is a distraction from the vendor's roadmap. It could ruin the product's "*c'est la mode*." Routers are too critical. Router protocols are difficult to design, implementing them requires highly specialized programming skills, and verification is expensive. These excuses sound all-to familiar.

But What If It Became Easy?

To understand why writing routing protocols is so difficult requires understanding how routing works in today's networks. Networks are made of endpoints (your PC and the server it is talking to) and the intermediate devices that connect them. Between your PC and www.acm.org there may be 20 to 100 routers (technically, *routers and switches* but for the sake of simplicity all packet-passing devices are called *routers* in this article). Each router has many network-interface jacks, or *ports*. A packet arrives at one port, the router examines it to determine where it is trying to go, and sends it out the port that will bring it one *hop* closer to its destination.

When your PC sends a packet, it does not know how to get to the destination. It simply marks the packet with the desired destination, gives it to the next hop, and trusts that this router, and all the following routers, will eventually get the packet to the destination. The fact that this happens trillions of times a second around the world and nearly every packet reaches its destination is awe-inspiring.

No "Internet map" exists for planning the route a packet must take to reach its destination. Your PC does not know ahead of time the entire path to the destination. Neither does the first router, nor the next. Every hop trusts that the next hop will be one step closer and eventually the packet will reach the destination.

How does each router know what the next hop should be? Every router works independently to figure it out for itself. The routers cooperate in an algorithm that works like this: each one periodically tells its neighbors which networks it connects to, and each neighbor accumulates this information and uses it to deduce the design of the entire network. If a router could think out loud, this is what you might hear: "My neighbor on port 37 told me she is 10 hops away from network 172.11.11.0, but the neighbor on port 20 told me he is four hops away. Aha! I'll make a note that if I receive a packet for network 172.11.11.0, I should send it out port 20. Four hops is better than 10!"

How does each router know what the next hop should be? Every router works independently to figure it out for itself.

Although they share topological information, routers do the route calculations independently. Even if the network topology means two nearby routers will calculate similar results, they do not share the results of the overlapping calculations. Since every CPU cycle uses a certain amount of power, this duplication of effort is not energy efficient.

Fortunately, routers need to be concerned with only their particular organization's network, not the entire Internet. Generally, a router stores the complete *route table* for only its part of the Internet—the company, university, or ISP it is part of (its autonomous domain). Packets destined for outside that domain are sent to *gateway routers*, which interconnect with other organizations. Another algorithm determines how the first set of routers knows where the gateway routers are. The combination of these two algorithms requires less RAM and CPU horsepower than if every router had to know the entire Internet. If it were not for this optimization, then every router would need enough RAM and CPU horsepower to store an inventory of the entire Internet. The cost would be staggering.

The routing algorithms are complicated by the fact that routers are given clues and must infer what they need to know. For example, the conclusion that "port 20 is the best place to send packets destined for 172.11.11.0" is inferred from clues given by other routers. A router has no way of sending a question to another router. Imagine if automobiles had no windows but you could talk to any driver within 25 feet. Rather than knowing what is ahead, you would have to infer a worldview based on what everyone else is saying. If everyone were using the same vocabulary and the same system of cooperative logical reason-

ing, then every car could get to where it is going without a collision. That's the Internet: no maps; close your eyes and drive.

Every router is trying to infer an entire worldview based on what it hears. Driving these complicated algorithms requires a lot of CPU horsepower. Each router is an expensive mainframe doing the same calculation as everyone else just to get the slightly different result it needs. Larger networks require more computation. As an enterprise's network grows, every router needs to be upgraded to handle the additional computation. The number and types of ports on the router haven't changed, but the engine no longer has enough capacity to run its algorithms. Sometimes this means adding RAM, but often it means swapping in a new and more expensive CPU. It is a good business model for network vendors: as you buy more routers, you need to buy upgrades for the routers you have already purchased. These are not standard PCs that benefit from the constant Dell-vs.-HP-vs.-everyone-else price war; these are specialized, expensive CPUs that customers cannot get anywhere else.

The Radical Simplification of OpenFlow

The basic idea of OpenFlow is that things would be better if routers were dumb and downloaded their routing information from a big "route compiler in the sky," or RCITS. The CPU horsepower needed by a router would be a function of the speed and quantity of its ports. As the network grew, the RCITS would need more horsepower but not the routers. The RCITS would not have to be vendor specific, and vendors would compete to make the best RCITS software. You could change software vendors if a better one came along. The computer running the RCITS software could be off-the-shelf commodity hardware that takes advantage of Moore's Law, price wars, and all that good stuff.

Obviously, it is not that simple. You have to consider other issues such as redundancy, which requires multiple RCITS and a failover mechanism. An individual router needs to be smart enough to know how to route data between it and the RCITS, which may be several hops away. Also, the communication channel between entities needs to be secure. Lastly, we need a much better name than RCITS.

The basic idea of OpenFlow is that things would be better if routers were dumb and downloaded their routing information from a big "route compiler in the sky."

The OpenFlow standard addresses these issues. It uses the term *controller* or *controller platform* (yawn) rather than RCITS. The controller takes in configuration, network, and other information and outputs a different blob of data for each router, which interprets the blob as a decision table: packets are selected by port, MAC address, IP address, and other means. Once the packets are selected, the table indicates what to do with them: drop, forward, or mutate then forward. (This is a gross oversimplification; the full details are in the specification, technical white papers, and presentations at http://www.OpenFlow.org/wp/learnmore.)

Traditional networks can choose a route based on something other than the shortest path—perhaps because it is cheaper, has better latency, or has spare capacity. Even with systems designed for such *traffic engineering*, such as Multiprotocol Label Switching Traffic Engineering (MPLS TE), it is difficult to manage effectively with any real granularity. OpenFlow, in contrast, enables you to program the network for fundamentally different optimizations on a per-flow basis. That means latency-sensitive traffic can take the fastest path, while bulk flows can take the cheapest. Rather than being based on particular endpoints, OpenFlow is granular down to the type of traffic coming from each endpoint.

OpenFlow does not stop at traffic engineering, however, because it is capable of doing more than populating a forwarding table. Because an OpenFlow-capable device can also rewrite the packets, it can act as a NAT or AFTR (address family transition router); and because it can drop packets, it can act as a firewall. It can also implement ECMP (equal-cost multi-path) or other load-balancing algorithms. They do not have to have the same role for every flow going through them; they can load-balance some, firewall others, and rewrite a few. Individual network elements are thus significantly more flexible in an OpenFlow environment, even as they

shed some responsibilities to the centralized controller.

OpenFlow is designed to be used entirely within a single organization. All the routers in an OpenFlow domain act as one entity. The controller has godlike power over all the routers it controls. You would not let someone outside your sphere of trust have such access. An ISP may use OpenFlow to control its routers, but it would not extend that control to the networks of customers. An enterprise might use OpenFlow to manage the network inside a large data center and have a different OpenFlow domain to manage its WAN. ISPs may use OpenFlow to run their patch of the Internet, but it is not intended to take over the Internet. It is not a replacement for inter-ISP systems such as Border Gateway Protocol (BGP).

The switch to OpenFlow is not expected to happen suddenly and, as with all new technologies, may not be adopted at all.

Centralizing route planning has many benefits:

▸ **Takes Advantage of Moore's Law.** Not only are general-purpose computers cheaper and faster, but there is more variety. In the Google presentation at the 2012 Open Networking Summit, Urs Höelzle said Google does its route computations using the Google compute infrastructure.

▸ **Offers Deeper Integration.** End-to-end communication can occur directly from the applications all the way to the controller. Imagine if every Web-based service in your enterprise could forward bandwidth requirements to the controller, which could then respond with whether or not the request could be satisfied. This would be a radical change over the "send and pray" architectures in use today.

▸ **Turns Network Hardware into a Commodity.** The CPU and RAM horsepower required by the device is a function of the speeds and number of ports on the device as shipped. Therefore, it can be calculated during design, eliminating the need to factor in slack. Also, designing and manufacturing a device with a fixed configuration (not upgradable) is less expensive.

▸ **Makes Algorithms Simpler.** Rather than making decisions based on inference and relying on cooperating algorithms, more-direct algorithms can be used. A dictatorship is the most efficient form of government. Suppose the VoIP phones of the campus EMS team should always get the bandwidth they need. It is easier to direct each router on campus to give EMS phones priority than it is to develop an algorithm whereby each router infers which devices are EMS, verifies a trust model, confirms that trust, and allocates the bandwidth—and hopes that the other routers are doing the same thing.

▸ **Enables Apps.** The controller can have APIs that can be used by applications. This democratizes router features. Anyone (with proper authorization and access) can create network features. No open source ecosystem exists for applications that run within the Cisco IOS operating system. Creating one will be much easier in the OpenFlow world. Apps will not control individual routers (this is already possible) but the entire network as a single entity.

▸ **Allows Global Optimization and Planning.** Current routing protocols require each router to optimize independently, often resulting in a routing plan that is optimal locally but not globally. The OpenFlow controller can plan based on a global, mostly omniscient, understanding of the network.

▸ **Provides Centralized Control.** Not that today's routers do not have any APIs, but the applications would need to communicate with the API of every router to get anything done and I do not know any network engineer who is open to that. With OpenFlow, apps can talk to the controller where authentication, authorization, and vetting can happen in one place rather than on every router.

Conclusion

In the past smartphone vendors carefully controlled which applications ran on their phones and had perfectly valid reasons for doing so: quality of apps, preventing instability in the phone network, and protection of their revenue streams. We can all see now that the paradigm shift of permitting the mass creation of apps did not result in a "wild west" of chaos and lost revenue. Instead, it inspired entirely new categories of applications and new revenue streams that exceed the value of the ones they replaced. Who would have imagined Angry Birds or apps that monitor our sleep patterns to calculate the optimal time to wake us up? Apps are crazy, weird, and disruptive—and I cannot imagine a world without them.

OpenFlow has the potential to be similarly disruptive. The democratization of network-based apps is a crazy idea and will result in weird applications, but someday we will talk about the old days and it will be difficult to imagine what it was once like. C

Related articles on queue.acm.org

Revisitng Network I/O APIs: The netmap Framework
Luigi Rizzo
http://queue.acm.org/detail.cfm?id=2103536

SoC: Software, Hardware, Nightmare, Bliss
George Neville-Neil, Telle Whitney
http://queue.acm.org/detail.cfm?id=644265

TCP Offload to the Rescue
Andy Currid
http://queue.acm.org/detail.cfm?id=1005069

References

1. Hoelzle, U. Keynote speech at the Open Networking Summit (2012); http://www.youtube.com/watch?v=VLHJUfgxEO4.
2. Katti, S. OpenRadio: Virtualizing cellular wireless infrastructure. Presented at the Open Networking Summit (2012); http://opennetsummit.org/talks/ONS2012/katti-wed-openradio.pdf.
3. Lin, G. Industry perspectives of SDN: Technical challenges and business use cases. Presentation at the 2012 Open Networking Summit; http://opennetsummit.org/talks/ONS2012/lin-tue-usecases.pdf (slides 6-7).
4. SIGCOMM Demo; http://www.openflow.org/wp/2008/10/video-of-sigcomm-demo/.

Thomas A. Limoncelli is an author, speaker, and system administrator. His best-known books include *Time Management for System Administrators* (O'Reilly, 2005) and *The Practice of System and Network Administration*, 2nd edition (Addison-Wesley, 2007). He works at Google in New York City on the Ganeti project (http://code.google.com/p/ganeti). See his blog at http://EverythingSysadmin.com.

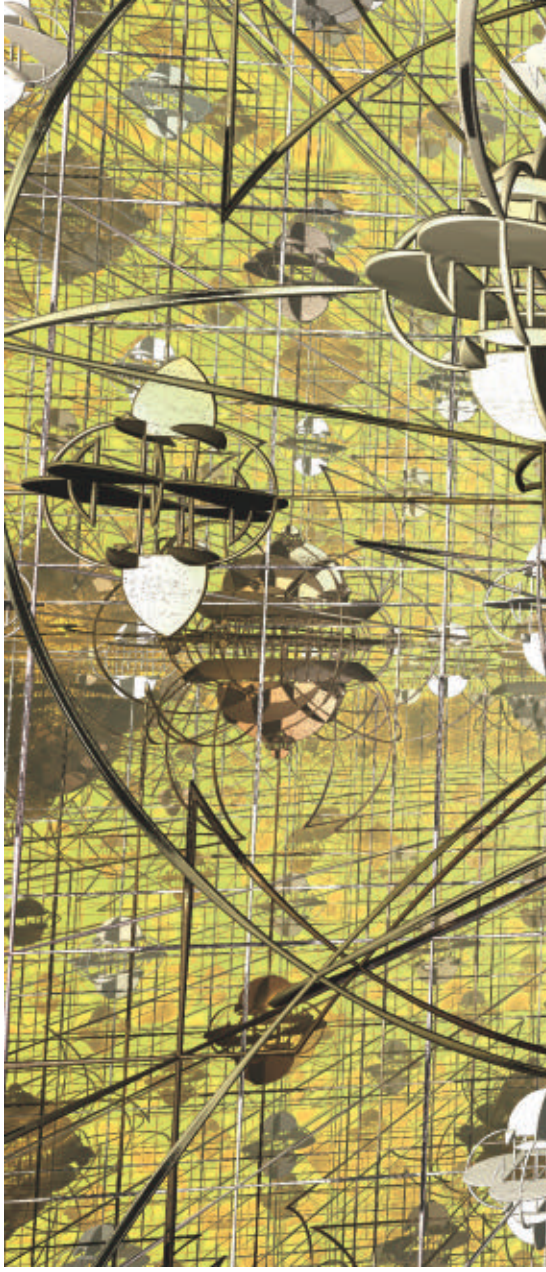
DOI:10.1145/2240236.2240252

Article development led by acmqueue
queue.acm.org

Increasing parallelism demands new paradigms.

BY RAFAEL VANONI POLANCZYK

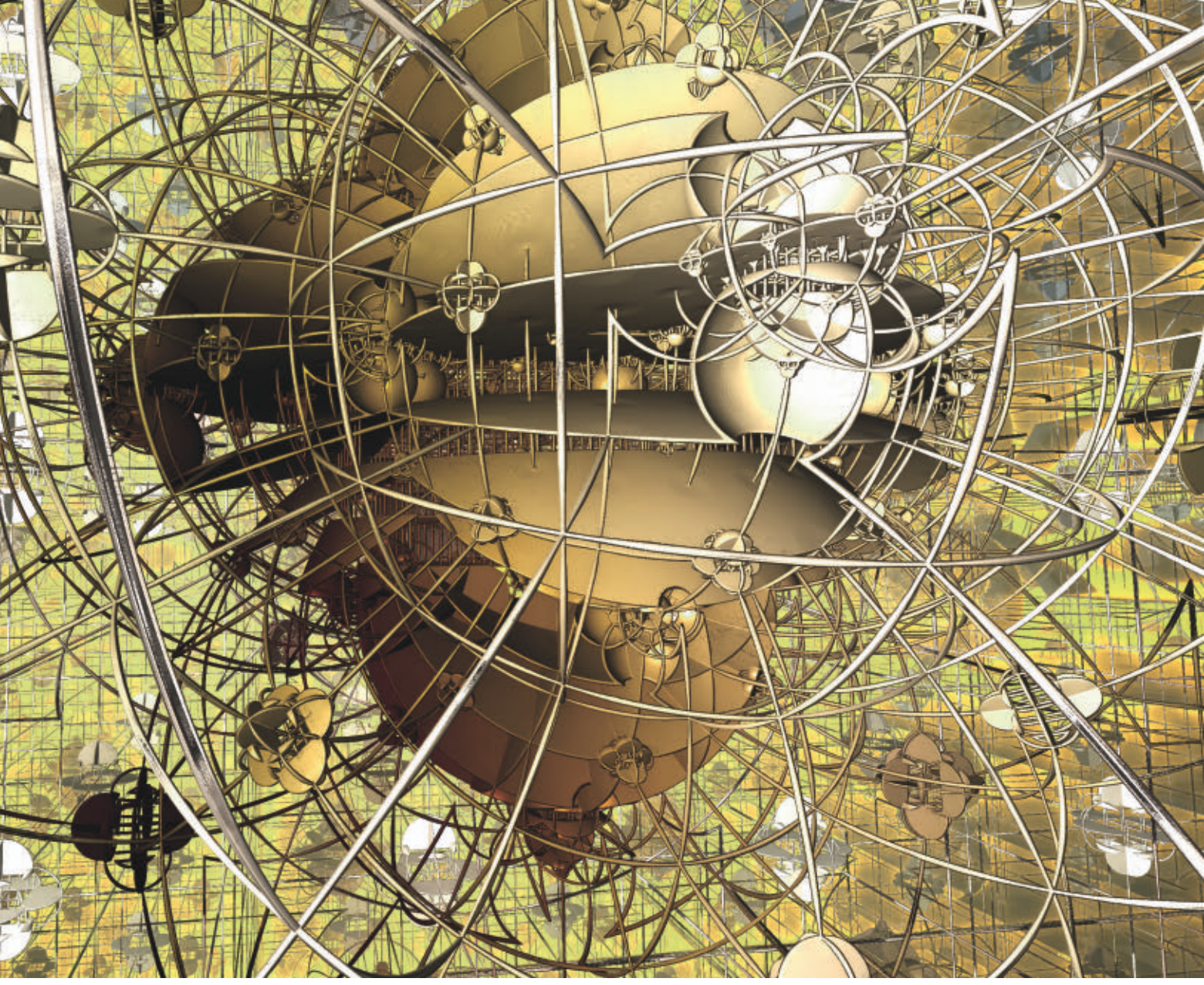
Extending the Semantics of Scheduling Priorities



APPLICATION PERFORMANCE IS directly affected by the hardware resources required, the degree to which such resources are available, and how the operating system addresses the requirements with regard to the other processes in the system. Ideally, an application would have access to all the resources it could use and be allowed to complete its work without competing with any other activity in the system. In a world of highly shared hardware resources and general-purpose, timeshare-based operating systems, however, no guarantees can be made as to how well resourced an application will be.

What can be done to improve both the way in which applications are developed and how the underlying layers of the software stack operate, in order to gain better overall utilization of shared hardware resources? Extending some of the semantics of scheduling priorities to include priority over shared resources could allow the performance-critical components of applications to execute with less-contended access to the resources they require.

The past decade has seen the emergence of the multicore processor and its subsequent rapid commoditization. Software developers, whether at the application level or in the broader design of systems, simply cannot ignore this dramatic change in the landscape. While not all problems require a parallel implementation as a solution,[1] this opportunity must be considered more seriously than ever before. Both academia and the industry have followed this trend by educating and advocating concurrent software development, while also seeking new



techniques to exploit hardware parallelism. Unfortunately, many details about developing concurrent applications on modern processors have mostly slipped through the cracks, only to be found in white papers, blogs, and similar watering holes of the performance community.

What developers are finding more often than not is that sizing parallel applications is not as straightforward as it once seemed. Until quite recently the one or two processors available within a single processor chip did not cause more contention for shared resources than perhaps two software threads fighting over shared cache space. Nowadays, not only are there several levels of sharing between logical CPUs (shared execution pipeline, floating-point units, different cache levels, among others), but also these many more CPUs make that sharing a much more complicated problem.

If this growing multiprocessing scale and its associated microarchitectural complexity were not enough, modern processors are also dynamically adapting their processing capacity based on the current utilization in an attempt to provide applications with the resourcing they need. Intel's Turbo Boost feature increases the processor's operating frequency as fewer cores are active and thermal conditions allow. The SPARC T4 processor, in contrast, dynamically allocates core resources to its active hardware threads, incrementally benefiting the few active threads by having more inactive ones. Both features are in essence enabling heterogeneous applications by improving single-threaded performance.

This new landscape poses new questions for you as a developer and system administrator: How many threads should your workload create? What resources do they require? Are all threads equally important? How should you size shared data structures? What should you tell the operating system (or more generically, the underlying layer) about your application?

Provisioning Threads in Multithreaded Applications

Although the simple classic recipe of one software thread for each logical CPU may still be valid in some cases, it can no longer be applied indiscriminately.[2] Parallel applications must know which portions of their program may require resources that are not widely available in the system. With that knowledge and some under-

ILLUSTRATION BY MIKAEL HVIDTFELDT CHRISTENSEN

standing of the possible deployment platforms, applications may be able to size themselves by matching their hardware requirements to what the underlying layer has to offer. Failure to do this properly leads to either contention over shared resources—as too many threads compete for them—or the underutilization of resources that the system otherwise had available.

For homogeneous multithreaded applications—those in which all threads perform similar tasks (and therefore have similar requirements)—you could simply partition the available resources into *n* slices according to how much of each resource a single thread will require. A scientific application that makes heavy use of floating point might create one thread per available floating-point unit (FPU) in the system (or two or three, if they can all take turns while executing their floating-point sections). For heterogeneous workloads, on the other hand, it may be advantageous to set aside more resources for specific threads. For example, in a producer/consumer architecture with a single producer and various consumers, giving the producer as many resources as it can take advantage of would likely be beneficial, trying to keep the consumers as busy as possible. This dependency relationship between producer and consumers is the primary point of contention in the application, making the producer its most critical component.

You may also want to exploit the knowledge of sharing relationships to take advantage of the dynamic resourcing features in recent processors. In other words, you can manually create the conditions that allow them to come into play. In this case the goal is not simply to prevent performance degradation by reducing the number of threads competing for some necessary component; you want the application to take advantage of all the performance you can get from the processor. In the previous producer/consumer example, the throughput of the application would likely increase if you placed the producer on a dedicated core, granting it exclusive access to all the hardware resources within that component.

Consequences of Virtualization

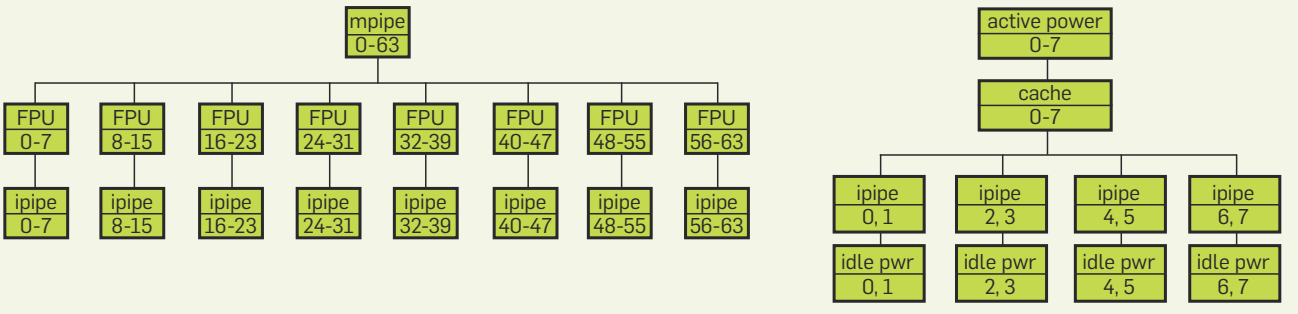
Virtualization mechanisms are a confounding factor, as they will often hide the details of the underlying architecture and the current utilization levels of its various components (such as obscuring direct access to the physical performance counters on the processor). This may prevent an application from determining the available physical resources, and therefore make it unable to size itself correctly by distributing its requirements across the available resources. It may also prevent the application from monitoring the consequences of its own behavior and adapting to changes in system utilization. Thankfully, these limitations can be circumvented if the application uses its own mechanisms to evaluate performance and capacity. For example, the application can run a micro-benchmark during startup, and/or periodically as it runs, to evaluate how the system is performing according to its own metrics. It can then use that information to adapt to the current conditions.

In the end, even a correctly sized parallel application still relies on the operating system—or on the underlying runtime environment—for mechanisms to provision threads, as well as for appropriate scheduling decisions for correct thread placement. Unfortunately, operating systems have traditionally offered only very simple mechanisms to provision specific threads. For example, the use of processor sets in Solaris to provide an entire core (and its otherwise shared components) to certain threads in an application is a reasonably well-known tuning method used by field engineers and specialized customers. Process binding is also used when manually placing processes and threads to ensure a desired behavior. These mechanisms are too static, however. They require manual intervention and are usually too expensive for this purpose. A preferable solution would be to provision threads more accurately with the resources they require as they become runnable, without user intervention, leveraging the knowledge that developers have of their applications and reducing the amount of work (or interference) required by the operating system or the system administrator.

Priority Over Shared Resources

The current implementation and semantics of scheduling priorities date from a period when single-processor systems were the norm. Resources were very limited and had to be correctly divided among threads in the system by allowing them to run for determined periods of time according to their priority. The recent emergence of

Example topologies for SPARC T4 and Intel Xeon processors.



systems with a large number of processors has fundamentally changed the scenario. Given the large number of resources available, threads no longer compete just for processor time, but also for shared hardware resources.

This scheduling model fails to recognize the sharing aspects of today's processors, allowing for some performance anomalies that are sometimes difficult to address. Consider, for example, a high-priority thread competing over a specific resource with a set of "hungry" lower-priority threads. In this case, it would be desirable to extend the implementation of priorities to include priority over shared resources. The operating system could then choose to move the lower-priority threads away from where the higher-priority one is running or to find a more appropriate place for it to execute with less-contended resources.

This extension presents a new method through which developers and system administrators can specify which components of an application should be more or less provisioned. It is a dynamic, unobtrusive mechanism that provides the necessary information for the operating system to provision threads more effectively, reducing contention over shared resources and taking advantage of the new hardware features discussed previously. Furthermore, the new behavior is likely to benefit users who already identify threads in their applications with different levels of importance (an important aspect of this work, for practical reasons).

Additionally, several other aspects of priorities play to our advantage. Since the proposed "spatial" semantics will determine how many resources will be assigned to threads, it is critical that this mechanism is restricted to users with the appropriate privileges—already a standard aspect of priorities in all Unix operating systems. Priorities can also be applied at different levels: at the process, thread, or function level, allowing optimizations at a very fine granularity.

The traditional implementation of load balancing does not perform well in heterogeneous scenarios unless the scheduler is capable of identifying the different requirements of each thread.

Load Balancing and Priorities

Load balancing is perhaps one of the most classic concepts in scheduling. Modulo implementation details, the basic idea is to equalize work across execution units in an attempt to have an even distribution of utilization across the system. This basic assumption is correct, but the traditional implementation of load balancing does not perform well in heterogeneous scenarios unless the scheduler is capable of identifying the different requirements of each thread and the importance of each thread within the application.

A few years ago the Solaris scheduler was extended to implement load balancing across shared hardware components in an effort to reduce resource contention. We had discovered that simply spreading the load across all logical CPUs was not enough: it was also necessary to load-balance across groups of processors that share performance-relevant components. To implement this policy, Solaris established the Processor Group abstraction. It identifies and represents shared resources in a hierarchical fashion, with groups that represent the most-shared components (pipe to memory, for example) at the top and groups that represent the least-shared ones at the bottom (such as execution pipeline). The accompanying figure illustrates the processor group topology for two different processors: the SPARC T4 and Intel Xeon processors, with each hardware component and the CPUs they contain.

Each processor group maintains a measure of its capacity and utilization, defined as the number of CPUs and running threads in a particular group. This information is then incorporated by the scheduler and used when deciding where to place a software thread, favoring groups where the utilization:capacity ratio would allow it to make the most progress.

Performance-Critical Threads

The processor group abstraction and the associated load-balancing mechanism for multicore, multithreaded processors successfully reduced contention at each level of the topology by spreading the load equally among the various components in the system. That alone, however, did not account for the different characteristics and resource requirements of each thread in a heterogeneous application or workload.

To address this issue Solaris recently extended its load-balancing mechanism so that a thread's notion of utiliza-

tion (or required resources) is proportional to its scheduling priority. This allows the scheduler to load-balance lower-priority threads away from where high-priority threads are running, automatically reducing contention for resources. With some simple heuristics, you can safely assume if a high-priority thread has enough CPU utilization to take advantage of the existing hardware resources, then it should be granted as much access to them as possible.

Automatically identifying which threads or portions of an application should be assigned a higher priority is not a simple task. There is no single characteristic that could allow us to make such assumptions for a wide variety of workloads—one could point out several cases where threads of widely different resource requirements are considered critical in the context of their applications. Most critical threads or components, however, are at the top of a dependency relationship in heterogeneous environments. From the startup components of applications to producer/consumer scenarios, any component upon which other parts of the application depend can be considered critical, and they should be assigned a suitably high priority. Such dependency relationships, however, are not easily observable without some previous knowledge of the architecture of the application.

In Solaris 11, once a performance-critical component or thread is identified, the developer or system administrator has only to place it in the fixed-priority scheduling class at priority 60 or at any real-time priority. The scheduler will then artificially inflate its load according to the underlying platform, attempting to improve its performance by allowing it to execute with more exclusive access to hardware resources. It is important to note that this optimization was implemented to take advantage of the available resources in the system. In other words, if all of the system's logical CPUs are required, no single thread will be forced to wait for the benefit of a single high-priority thread.

This implementation also optimizes differently according to the underlying hardware architecture. On sun4v systems, the scheduler will attempt to provision a performance-critical thread with all of the CPUs sharing an execution pipeline, and a quarter of the CPUs sharing a physical chip on x86 systems. These policies are optimized for both known sources of contention and dynamically resourcing features in their respective platforms. A simple example of the desired behavior would be to have an entire core devoted to a single high-priority thread on a SPARC T4 system, while all the other lower-priority threads share the remaining resources (again, as long as enough idle resources are available in the system).

From the startup components of applications to producer/consumer scenarios, any component upon which other parts of the application depend can be considered critical, and they should be assigned a suitably high priority.

Conclusion

The contemporary landscape of increasing parallelism requires new paradigms. These will affect developers and system administrators at a number of levels in developing new applications and systems. Some are occupied with the considerations of mechanisms at the level of the hardware, virtualization, and operating system. Application developers must have suitable means to designate critical elements of their applications and to interact with the underlying system software to ensure those elements are given the special resourcing that they require.

Acknowledgments

I am indebted to Eric Saxe, Jonathan Chew, and Steve Sistare who, among a broader number of colleagues in the Solaris Kernel Group, were particularly helpful in the development of the ideas presented in this article.

Related articles on queue.acm.org

Real-World Concurrency
Bryan Cantrill anf Jeff Bonwick
http://queue.acm.org/detail.cfm?id=1454462

Performance Anti-Patterns
Bart Smaalders
http://queue.acm.org/detail.cfm?id=1117403

References

1. Cantrill, B. and Bonwick, J. Real-world concurrency. *ACM Queue 6*, 5 (2008).
2. Smaalders, B. Performance anti-patterns. *ACM Queue 4*, 1 (2006).

Rafael Vanoni Polanczyk is a software developer in the Solaris Kernel Group at Oracle, where he spends most of his time working on the scheduler/dispatcher subsystem. Rafael lives in San Francisco and is originally from Porto Alegre in southern Brazil, where he received a B.Sc. in computer science from UFRGS (Universidade Federal do Rio Grande do Sul).

DOI:10.1145/2240236.2240253

Article development led by acmqueue
queue.acm.org

A first step toward programming 21st-century applications.

BY MANUEL SERRANO AND GÉRARD BERRY

Multitier Programming in Hop

THE WEB IS becoming the richest platform on which to create computer applications. Its power comes from three elements: modern Web browsers enable highly sophisticated graphical user interfaces (GUIs) with 3D, multimedia, fancy typesetting, among others; calling existing services through Web APIs makes it possible

to develop sophisticated applications from independently available components; and open-data availability allows applications to access to a wide set of information that were unreachable or that simply did not exist before. The combination of these three elements has already given birth to revolutionary applications such as Google Maps, radio podcasts, and social networks.

The next step is likely to be incorporating the physical environment into the Web. Recent electronic devices are equipped with various sensors (GPS, cameras, microphones, metal detectors, speech commands, thermometers, motion detection, and so on) and communication means (IP stack, telephony, SMS, Bluetooth), which enable applications to interact with the real world. Web browsers integrate these features one after the other, making the Web runtime environment richer every day. The future is appealing, but one difficulty remains: current programming methods and languages are not ideally suited for implementing rich Web applications. This is not surprising as most were invented in the 20th century, before the Web became what it is now.

Traditional programming languages have trouble dealing with the asymmetric client-server architecture of Web applications. Ensuring the semantic coherence of distributed client-server execution is challenging, and traditional languages have no transparent support for physical distribution. Thus, programmers need to master a complex gymnastics for handling distributed applications, most often using different languages for clients and servers. JavaScript is the dominant Web language but was

conceived as a browser-only client language. Servers are usually programmed with quite different languages such as Java, PHP, and Ruby. Recent experiments such as Node.js propose using JavaScript on the server, which makes the development more coherent; however, harmonious composition of independent components is still not ensured.

In 2006, three different projects—namely, Google Web Toolkit (GWT), Links from the University of Edinburgh,[1] and Hop from INRIA (http://www.inria.fr)[5]—offered alternative methods for programming Web applications. They all proposed that a Web application should be programmed as a single code for the server and client, written in a single unified language. This principle is known as multitier programming.

Links is an experimental language in which the server holds no state, and functions can be symmetrically called from both sides, allowing them to be declared on either the server or the client. These features are definitely interesting for exploring new programming ideas, but they are difficult to implement efficiently, making the platform difficult to use for realistic applications.

GWT is more pragmatic. It maps traditional Java programming into the Web. A GWT program looks like a traditional Java/Swing program compiled to Java bytecode for the server side and to JavaScript for the client side. Java cannot be considered the unique language of GWT, however. Calling external APIs relies on JavaScript inclusion in Java extensions. GUIs are based on static components declared in external HTML files and on dynamic parts generated by the client-side execution. Thus, at least Java, JavaScript, and HTML are directly involved.

The Hop language takes another path relying on a different idea: incorporating all the required Web-related features into a single language with a single homogeneous development and execution platform, thus uniformly covering all the aspects of a Web application: client side, server side, communication, and access to third-party resources. Hop embodies and generalizes both HTML and JavaScript functionalities in a Scheme-based[3] platform that also provides the user with a fully general algorithmic language. Web services and APIs can be used as easily as standard library functions, whether on the server side or client side.

Figure 1. A file browser Web widget.

```
(define-tag <BROWSER> ((id (gensym))           ;; HTML identifier
                       (class #f)              ;; HTML class
                       (path::bstring "/tmp")) ;; default initial path
   (define (<dir-entry> path)
      ;; function to build HTML display for directories
      (<DIV> :data-hss-tag "dir-entry"
         :onclick ~(with-hop ($(service () (<directory> path)))
                     (lambda (h)
                        (innerHTML-set! $id h)))
         name))
   (define (<file-entry> path)
      ;; function to build HTML display for files
      (<DIV> :data-hss-tag "file-entry"
         (basename path)))
   (define (<directory> path)
      ;; main display function
      (cons
         ;; HTML representation of the parent directory
         (<dir-entry> (make-file-name path ".."))
         ;; HTML representation of files
         (map (lambda (np)
                 (if (directory? np)
                     (<dir-entry> np)
                     (<file-entry> np)))
            (sort string<? (directory->path-list path)))))
   ;; main HTML element
   (<DIV> :data-hss-tag "browser"
      :id id :class class
      (<directory> path)))
```

Figure 2. Hop compiled into HTML+JavaScript

```
<div data-hss-tag='dir-entry'>
  onclick='HOP.with_hop( "/hop/anonymous314",
     function( h ) {
       var _el3 = document.getElementById( "g1592" );
       _el3.innerHTML = HOP.eval( h );
     } )'
 /tmp
</div>
```

Multitier Programming and HTML Abstraction

This article presents the Hop approach to Web application programming, starting with the basic example of a Web file viewer. In the first version, files and directories are listed on a bare page. Directories are clickable to enable dynamic browsing, and plain file names are displayed. This simple problem illustrates the most central aspect of realistic Web applications—namely, the tight collaboration and synchronization of servers and clients. Here, the server owns the files while the browser visualizes them. When the user clicks on a directory, the client requests new information from the server.

Upon receipt, the client updates the page accordingly. The file viewer is easy to implement if each request delivers a new complete page. By adding the requirement that the file viewer should be a widget embedded inside a complex Web page, the problem becomes slightly more difficult, since updates now concern only incremental subparts of the documents. This demands a new kind of collaboration between the server and the client.

Implementing such a Web client-server file viewer with a single Hop code is almost as straightforward as implementing it for a single computer, because HTML is built in and execution partitioning between servers and clients is automatic. Figure 1 shows a complete Hop solution to this problem.

Hop identifiers can contain special characters such as "<" and "?". In Hop, `directory?`, `string=?`, and `<SPAN>` are legal identifiers. In the example, `<BROWSER>` is the name of a variable bound to a function that creates an HTML `div` element.

HTML objects are created by Hop functions with the same names (`<DIV>`...), (`<SPAN>` ...), among others; no HTML closure is needed because the code deals with parenthetical functional expressions instead of texts. The full power of Scheme is used to build `DIV`s and `SPAN`s. The `<dir-entry>` and `<file-entry>` auxiliary functions are used to build fragments of the final HTML document. The `map` operation in the main `<directory>` function is used to build the global HTML page out of these HTML fragments, with help from the Hop `sort` function to sort the printed output alphabetically. Figure 2 shows how values produced by the `<div-entry>` function are compiled into HTML and JavaScript.

The `~` and `$` constructs appearing at line 7 of Figure 1 are the multitier programming operators. Let us explain how they work. Expressions prefixed by `~` are client-side expressions evaluated by the browser; unannotated expressions are evaluated on the server. Compiling is done on the server. A client code is seen as a value, computed or elaborated by the server and automatically shipped to the client on demand. The `$` construct is used to inject a server value within the client code at elaboration time. For example, `~(alert $(hostname))` prints the server name on the client screen. In the example of Figure 1, line 7 provokes the injection of an automatically generated service named `anonymous314` in the HTML excerpt of Figure 2. The `~`-prefixed expression at line 7 specifies the action to be executed by the client when the user clicks on a directory. The action invokes the anonymous service, called by the client and run on the server.

Runtime communication between servers and clients involves services that extend the notion of function. A *service* is the binding of a function to an URL. The URL is used to invoke the function remotely, using the special form (`with-hop (<service> <arguments>) <callback>`). The arguments are marshaled for network transmission and the remote procedure call is initiated; on remote-call completion, the callback is invoked on the caller side with the unmarshaled remote-call result. Extra options control how errors and timeouts are handled (these are not presented in this article).

The example here contains one anonymous service defined in the `<dir-entry>` auxiliary function. This service is invoked each time a user clicks on a directory. It calls the `<directory>` function that traverses the server file system. Because services are statically scoped just as functions, the `<directory>` identifier in the service code refers to the `<directory>` server function defined later in the same mutually recursive define clause. The `path` parameter is bound by the definition of the `<dir-entry>` function and used in the service call from within the client code. Lexical binding is the same for the server and client code.

Once the `<BROWSER>` function is defined, it can be freely used as a new HTML tag in any regular HTML element within Hop. For example, the code shown in Figure 3 instantiates two file browsers in a table.

Making variations on `<BROWSER>` that impact both the server and the client is very easy. Figure 4 shows a one-line modification of the `<file-entry>` function body that displays the file name and size.

Hop relies on a mixed dynamic/static type discipline. When a variable or a function is annotated with a type, the compiler uses this type to statically check type correctness and improve the generated code. To detect compile-time errors, Hop does not compete with statically typed languages such as ML or Haskell but trades static guarantees for expressiveness: for example, it supports programming constructs that are out of reach of statically typed languages, such as CLOS-like object-oriented programming style. Also note that the children of HTML nodes can be of any type (a list, string, number, another node, and others) without requiring type annotations, as specified by the W3C recommendation of XML Schema for HTML. In the example, only the path variable is type-annotated.

A Hop widget may declare its own CSS (Cascading Style Sheets) rules to abstract away its implementation details. The declaration in Figure 5 creates a new CSS type associated with the `<BROWSER>` tag. Figure 6 shows how one can use the browser CSS type to add extra icons before directory and leaf nodes.

Figure 3. A Web file browser.

```
(<HTML>
   (<TABLE>
      (<TR>
         (<TD> (<BROWSER> :class "usr" :path "/usr"))
         (<TD> (<BROWSER> :class "local" :path "/usr/local")))))
```

Figure 4. Modified file browser for file size.

```
(define (<file-entry> path)
   (<DIV> :data-hss-tag "file-entry"
      (basename path)
      (<SPAN> (round (file-size path) 1024)))) ;; <-- added line for size
```

In Hop, HTML objects on the server are members of a full-fledged data structure that implements the abstract syntax tree of the client HTML document; this contrasts with most Web environments in which they are reduced to bare strings of characters. A Hop server computation elaborates a DOM (document object model) representation of HTML similar to the one used by the browser, compiles it on the fly to actual HTML, and ships it on browser demand.

This multistage process eliminates several drawbacks of traditional Web frameworks. First, the Hop runtime environment guarantees several properties of the generated HTML such as syntactic correctness—for example, reporting HTML tag misspellings as unbound-variable errors. Second, a single document can be automatically compiled into different HTML versions on demand. For example, the same document can be optionally compiled into a mix of HTML4 and Flash for old browsers, into HTML5 for more skilled browsers, or even into XHTML+RDFa for semantics annotations.

Most current client-side Web libraries abstract over HTML by providing a set of JavaScript functions that accept regular HTML nodes as parameters; typically, `DIV` and `SPAN` are used as new widget containers. This approach has several drawbacks. First, HTML extensions do not look like regular tags. The implementation of a GUI thus requires assembling different formalisms. Second, extension initializations are complex to schedule. In general, the application must resort to window `onload` JavaScript events to ensure that the API constructors are not called before the DOM tree is fully built. Third, to configure the graphical rendering of the extension, implementation details must be unveiled to let programs designate the HTML elements to be configured. This jeopardizes maintainability.

Figure 5. Extending CSS with a new type and new attributes.

```
$(define-hss-type browser "div[data-hss-tag=browser]"
   (define-hss-property (directory-image v)
      (format "div[data-hss-tag=dir-entry] {
        background-image: ~l;
        background-position: left;
        background-repeat: no-repeat;
        padding-left: 2em;
      }" v))
   (define-hss-property (file-image v)
      (format "div[data-hss-tag=file-entry] {
        background-image: ~l;
        background-position: left;
        background-repeat: no-repeat;
        padding-left: 2em;
      }" v))
```

Figure 6. CSS user rules.

```
browser {
  directory-image: url( http://nowhere.org/dir.png );
  leaf-image: url( http://nowhere.org/leaf.png );
}

browser.blue {
  directory-image: url( http://nowhere.org/blue/dir.png );
  color: blue;
  font-weight: bold;
}
```

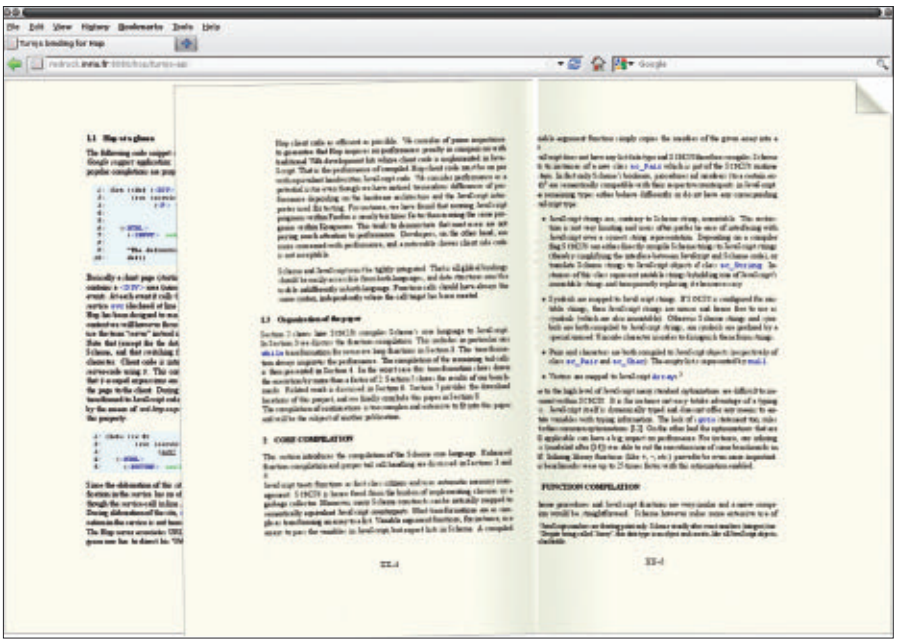


Figure 7. A PDF viewer in a Web browser.

Code Reuse

The example file-viewer application presented here involves the main ingredients of a Web application—a distributed architecture, HTML abstraction, and CSS configurations—but it is still too simple to be realistic. To develop richer Web applications with comparatively little effort, it is now common practice to rely on the wide set of publicly available JavaScript APIs. All Web frameworks offer one way or another of using them, often through backdoors that let the programmer insert alien JavaScript calls into the natively supported language.

Hop follows a different approach by integrating these APIs into its language. Calling a JavaScript function or creating a JavaScript object from Hop is as easy as manipulating a standard Hop entity. Furthermore, once compiled, client-side Hop-generated JavaScript can be distinguished from handwritten JavaScript only by the name

mangling used to map Hop identifiers to JavaScript identifiers. Everything else is identical: Hop data, variables, and functions are directly mapped to their JavaScript counterparts.

The direct integration of JavaScript APIs within Hop makes Web-platform development by component combination easy. To illustrate API integration, we show how to write a PDF viewer that displays the contents of files with the nice visual effect of turning the pages of an actual book. Figure 7 shows a snapshot of this application, and Figure 8 displays the actual source code. It is based on jQuery, a popular JavaScript library; a jQuery plug-in called turn.js (http://www.turnjs.com), which implements the page-turning effect; and the JavaScript PDF previewer implemented by the Mozilla Foundation (https://github.com/andreasgal/pdf.js). It is assumed that the PDF previewer is packaged in the same way as the <BROWSER> first presented. In the code, the method turn initializes the turn.js plug-in. It prepares an HTML element for use as a book container. This JavaScript method is directly used as a regular Hop function.

The turn.js API automatically invokes JavaScript listeners before and after page turns. This feature can generate page content on demand on the server. Figure 9 presents a complete example where the client requests page contents from the server right before turning a page. This example uses the turn.js method bind to bind an anonymous Hop function to user events, in this case associating a user function with page-turn events. This shows that Hop functions can be invoked directly by JavaScript as regular JavaScript functions.

No platform other than the Web has ever let anyone write sophisticated GUIs so simply, by combining APIs and codes provided by different parties.

Open Data

Governments and public organizations have recently understood that the data they generate is a valuable asset. New companies such as Data Publica were created with the sole purpose of collecting, organizing, and redistributing open data.

Traditionally, open data was exogenous to the Web. For example, the

Figure 8. A PDF viewer invoking two JavaScript APIs.

```
(<HTML>
   (<HEAD>
      ;; Third-party JavaScript libraries
      :jscript "http://code.jquery.com/jquery-1.7.1.min.js"
      :jscript "http://www.turnjs.com/turn.min.js"
      ;; Hop bindings for external JavaScript
      :include "pdfjs-api-1.0.0.hz")
   (<DIV>
      ~(add-event-listener! window "load"
          (lambda ()
             ;; initialize the turnjs plugin when the client document is ready
             (let ((bk (jQuery "#book"))) ;; direct use of a jQuery function
                (bk.turn))))              ;; direct use of a turnjs method
      (<DIV> :id "book"
         (map (lambda (i)
                 ;; create one HTML element per PDF page
                 (<PDF> :src (make-file-path REPOSITORY "queue12.pdf") :page i))
            (iota 10 1)))))
```

Figure 9. A dynamically generated book using a JavaScript API.

```
(define-service (filebook count)

   (define cache
      ;; a local cache to avoid computing several times the same page
      (make-vector count #f))

   (define getpage
      ;; the service that returns the pages content
      (service (i)
         (unless (vector-ref cache i)
            ;; not in the cache, fill the page with a random quotation
            (vector-set! cache (system->string "fortune")))
         ;; returns the cache entry
         (<BLOCKQUOTE> (vector-ref cache i))))

   (<HTML>
      (<HEAD>
         :jscript "http://code.jquery.com/jquery-1.7.1.min.js"
         :jscript "http://www.turnjs.com/turn.min.js"
         :css "filebook.hss")

      ~(define (turnpage i)
          ;; function called for each visualized page
          (with-hop ($getpage i)
             (lambda (h)
                ;; insert result of the getpage call in the i-th div
                (innerHTML-set! (format "page-~a" i) h))))

      ~(add-event-listener! window "load"
          ;; initialize the turn plugin when the client document is ready
          (lambda (_)
             (let ((el (jQuery "#dir")))
                (el.turn)
                (el.bind "turning"
                   (lambda (_ page)
                      ;; function invoked by the turnjs plugin when
                      ;; the user clicks on a next/prev page button
                      (turnpage page)
                      (turnpage (+ 1 page)))))))

      (<DIV> :id "dir"
         ;; create one empty div per document page; these divs are
         ;; filled by the client-side turnpage function
         (map (lambda (i) (<DIV> :id (format "page-~a" i) ""))
            (iota count)))))
```

French government agency INSEE (National Institute of Statistics and Economic Studies; http://www.insee.fr), created in the middle of the 20th century, is in charge of conducting all sorts of statistical analyses of the population and economy. INSEE has become an important open data provider.

Surprisingly, the data might be endogenous, too. Google's flu trends analysis is a remarkable example of this kind. Google researchers have observed that the number and locales of search requests about flu is strongly related to the propagation of the illness.[2] Google makes this data available country by country, making it easy for anyone to implement applications using these statistics.

Here, we demonstrate how to create a video showing the propagation of the flu over time. This is mostly a tool-combination problem since all the hard work can be delegated to external parties. Thus, the work shown here consists only of collecting, combining, and reusing all the tools and data at our disposal. The actual Hop implementation, shown in Figure 10, counts no more than 15 lines of code.

Figure 10. Visualizing the flu propagation worldwide.

```
(module flutrend
   (import (gchart "gchart-api-1.0.*.hz")))

(<HTML>

   ~(define sources
       $(map (lambda (data) (google-chart :type 'map :data data))
            (with-input-from-file "http://www.google.org/flutrends/data.txt"
               (lambda (p)
                  ;; skip legal Google notice
                  (read-lines p 11)
                  ;; parse the actual flu trends values
                  (read-csv-records p)))))

   (<IMG> :src (google-chart :type 'map :area 'europe)
      :onload ~(when (pair? sources)
                  (after 300
                     (lambda ()
                        (set! this.src (car sources))
                        (set! sources (cdr sources)))))))
```

Figure 11. Reacting to SMS.

```
(module androidemo
   (library mail phone hopdroid))

(define android (instantiate::androidphone))
(define tts (instantiate::androidtts (phone android)))

(define-service (phone/sms)

   (define speak #t)

   (add-event-listener! android "sms-received"
      (lambda (e::event)
         ;; function invoked on the server (i.e., the phone)
         ;; upon sms-reception
         (let ((val e.value))
            ;; broadcast to all clients the reception of the sms
            (hop-event-broadcast! "sms" val)
            ;; speak out the sms content on the phone
            (when speak (tts-speak tts (cadr val))))))

   ;; create two empty HTML elements that will be filled
   ;; upon SMS reception
   (let ((sms-num (<TD>))
         (sms-body (<TD>)))
      (<HTML>
         ~(add-event-listener! server "sms"
             ;; function invoked on the client (i.e., a web browser)
             ;; upon reception of the server "sms" event
             (lambda (e)
                ;; insert the calling number in the HTML page
                (innerHTML-set! $sms-num (car e.value))
                ;; insert the SMS content in the HTML page
                (innerHTML-set! $sms-body (cadr e.value))))
         (<TABLE> (<TR> sms-num sms-body))
         (<BUTTON> :onclick ~(with-hop ($(service () (set! speak (not speak)))))
            "mute/speak"))))
```

The application code relies on a binding that makes the Google Chart API directly accessible from Hop. This Web API creates all kinds of charts and geographical maps, generating images according to the arguments specified on appropriate URLs. This API is combined with the extraction of flu statistics. First, image URLs are generated out of statistical values parsed using a Hop spreadsheet-elements library. Then images generated on the fly by Google Chart are displayed one after the other, every 300 milliseconds.

This example shows that new and deep knowledge might emerge when the ability to collect, compare, and analyze vast sets of open data is easily accessible. Hop is specifically geared toward this objective, with composition and reuse as the two central features.

Bigger than the Web

The human Web can be made even bigger by binding sensors from and actuators to the physical environment, such as those provided by smartphone sensors, multimedia equipment, home or automotive automation equipment, and the countless other electronic devices that surround us. However, only the most popular of these sensors will be natively supported by the next Web browsers. For example, the incoming versions of the Firefox and Chrome navigators have announced support for video and audio capture; accessing other remote equipment through browsers or programmed applications will still require third-party support. The Hop philosophy is to migrate the remote interface facilities to the server, give the user full Hop programming power on the associated data, and deliver results to clients in the standard way when using browser-based interfaces.

For example, Figure 11 shows how to implement a Hop application that integrates features provided by a mo-

bile phone inside a Web application. The contents of incoming SMSs are displayed in a browser window, letting users choose whether or not an SMS should be spoken aloud.

This application uses three separated tiers: a Web browser, a Web server, and a phone server. The phone server is responsible for receiving the SMSs and speaking them on demand. The Web server plays the role of orchestrator; when the phone notifies it that a new SMS has been received, it alerts the interested clients and, depending on the user configuration, speaks out the short message.

The application takes advantage of another useful feature of Hop: allowing a server to be a new source of program-generated Web events for clients or other servers. In the phone example, the server starts establishing a bridge with the phone (using the `android` variable), then waits for the notification of an SMS and accesses the phone text-to-speech facility (using the `tts` variable).

The server code registers a function invoked each time a new SMS is received to forward a software Web event to interested Web clients.

Implementation

Hop is an open source project whose source code can be downloaded from the Web site http://hop.inria.fr. All the actual source code shown in this article can also be downloaded from this Web site. The development kit contains the compilers, interactive documentation, various tools for creating and installing applications, and the runtime environment, which is a dedicated Web server that runs on all Un*x platforms—Linux, Mac OS X, and Android. It is operational on mainstream modern architectures such as x86/32, x86/64, PowerPC, and the ARM family.

The Hop runtime environment requires only 3MB–4MB of RAM. This small memory footprint makes it suitable for smartphones or even smaller machines, such as the new ARM-equipped computers (Phidget SBC2, Raspberry Pi, among others) that offer only a few megabytes to applications. As far as speed is concerned, the Hop Web server delivers dynamic Web pages significantly faster than traditional servers such as Apache or Lighttpd. The performance gain is a result of the Hop-in-Hop implementation, which enables the server to respond to requests involving dynamic computations without costly execution context switching.[4]

A New Playground

Hop is the conjunction of a multitier programming language and a runtime environment for the Web. Its main goals are to unify the linguistic features needed for Web applications, to automate the physical distribution of code in a fully transparent way, and to help programmers in reusing and combining external resources. All of these are keys to program simplicity and conciseness.

This article has presented several complete Web application examples written in Hop. The first example illustrates how to raise the HTML abstraction level by using an algorithmic programming language approach. The second example shows how to reuse third-party code. The third example shows that combining Web technologies with open data opens the path for new application ideas. The last example presents a simple diffuse application on the Web. These examples have all been chosen for their simplicity; the same technique would be equally effective when dealing with much bigger applications in domains such as home automation or multimedia.

None of the applications presented here exceeds 30 lines of Hop source code. This is not accidental; it is the consequence of a perception shift where the Web is no longer regarded as a mere platform for sharing documents but as a revolutionary runtime environment. Note, however, that Hop is a realistic programming language, which implies a lot of additional bells and whistles. This article has ignored most of them, concentrating on multitier programming and the connection to official Web technologies.

Finally, note that the Hop core language is a strict functional programming language, with runtime type checking. This design choice comes mostly from a personal bias of the first author. Arguably, it fits well with the current programming trends of the Web, but other flavors should be possible, yielding other programming languages. We hope to see such new languages appearing, because the multiplicity of approaches will foster creativity and possibly open a new era in language and tool design. C

Related articles on queue.acm.org

There's Still Some Life Left in Ada
Alexander Wolfe
http://queue.acm.org/detail.cfm?id=1035608

Extensible Programming for the 21st Century
Gregory V. Wilson
http://queue.acm.org/detail.cfm?id=1039534

Purpose-Built Languages
Mike Shapiro
http://queue.acm.org/detail.cfm?id=1508217

References

1. Cooper, E., Lindley, S., Wadler, P. nd Yallop, J. Links: Web programming without tiers. Presented at the 5th International Symposium on Formal Methods for Components and Objects (2006).
2. Ginsberg, J., Mohebbi, M., Patel, R., Brammer, L., Smolinski, M. and Brilliant, L. Detecting influenza epidemics using search engine query data. *Nature 457* (Feb. 19, 2009), 1012–1014.
3. Kelsey, R., Clinger, W. and Rees, J. The revised (5) report on the algorithmic language scheme. *Higher-Order and Symbolic Computation 11*, 1 (1998); http://www.sop.inria.fr/indes/fp/Bigloo/doc/r5rs.html/
4. Serrano, M. Hop, a fast server for the diffuse Web. In *Proceedings of the 11th International Conference on Coordination Models and Languages* (Lisbon, Portugal, 2009).
5. Serrano, M., Gallesio, E. and Loitsch, F. Hop, a language for programming the Web 2.0. In *Proceedings of the First Dynamic Languages Symposium* (Portland, OR, 2006).

Manuel Serrano is a senior scientist at INRIA, leading the INDES (Informatique Diffuse et Sécurisée) team in Sophia-Antipolis. After completing his Ph.D. at the Pierre and Marie Curie University (UPMC) on the compilation of functional languages, he moved to Nice and created the Bigloo development environment for Scheme. He joined INRIA in 2001 and has focused on development environments for the diffuse web since 2005.

Gérard Berry, director of research at INRIA, works on programming languages, their mathematical semantics, and program verification technologies. His focus has been on the lambda-calculus and its models, reactive and real-time programming and verification, and high-level digital circuits specification, synthesis, and verification. Prior to joining INRIA he was chief scientist of Esterel Technologies and was the main author of the Esterel language, which has been used in academia and industry for applications ranging from complex circuit synthesis to airplane control.

DOI:10.1145/2240236.2240255

How to have the best of location-based services while avoiding the growing threat to personal privacy.

BY STEPHEN B. WICKER

The Loss of Location Privacy in the Cellular Age

"OUR VIEW OF reality is conditioned by our position in space and time—not by our personalities as we like to think. Thus every interpretation of reality is based upon a unique position. Two paces east or west and the whole picture is changed."

—Lawrence Durrell, *Balthazaar*[10]

"...to be human is to be 'in place'."

—Tim Cresswell, *Place: A Short Introduction*[7]

On April 20, 2011, U.K. researchers Alasdair Allan and Peter Warden caused a media frenzy by announcing their discovery of an iPhone file—consolidated.db[a]—that contained time-stamped user-location data.[4] A FAQ published by Apple[3] and congressional testimony by Apple's vice president for software technology[26] subsequently revealed that at least some of the initial concerns were groundless. Assuming Apple's anonymity-preservation techniques are adequate, Apple does not compile location traces for individual users, instead enlisting those users as data collectors in a worldwide exercise in crowdsourcing. Apple is creating a highly precise map of cell sites and access points in an effort to improve the speed and accuracy of its user-location estimates, thus providing more-refined location-based services. However, despite Apple's quick and thorough response, long-term issues remain.

This article explores the evolution of location-based services (LBS), culminating in Apple's and Google's use of crowdsourced data to create a system for obtaining location fixes potentially faster and more accurate than the global positioning system (GPS). This article also develops an intuitive sense of the potentially revelatory power of fine-grain location data, then addresses the question of potential harm. The most obvious concern is the stalker, while others involve manipulation and threats to autonomy. Also provided is a brief review of the philosophy of place, focusing on the ability of location-based advertising (LBA) to disrupt individuals' relationships with their surroundings. It then turns to the potential for anonymous LBS, with the aim of saving the benefit while avoiding potential harm. Finally,

» key insights

- **The precision of cellular-location estimates means service providers are able to obtain location estimates with address-level precision, creating a serious privacy problem, as the estimates can be highly revealing of user behavior, preferences, and beliefs.**
- **Supposedly anonymous location traces can be de-anonymized through correlation with publicly available databases.**
- **Privacy-aware design makes it possible to retain the full benefit of LBS while preventing accumulation of address-level location traces for a given individual and reducing the potential for de-anonymization.**

a The file had already been identified in a 2010 text on iOS forensics by Sean Morrissey[20] but was largely ignored at the time.

ILLUSTRATION BY BRIAN GREENBERG/ANDRIJ BORYS ASSOCIATES

it asks: How much location data must a marketer acquire before a correlation attack can de-anonymize the data?, answering through the Shannon-theoretic concept of "unicity" distance and recommending ground rules for development of truly anonymous LBS.

Technology of Place

Cellular telephony has always been a surveillance technology. As discussed by the author,[27] cellular networks are designed to track a phone's location so incoming calls are routed to the most appropriate cell tower, usually the one closest to the user. As most users are aware, recent generations of cellphones are capable of much more fine-grain location resolution. The first step toward adding this capability came with E911, the Federal Communications Commission's 1996 effort to enhance location resolution for cellular 911 calls.[b] E911 established a requirement that cellular service providers send location information to the Public Safety Answering Point when subscribers make 911 calls with their cellphones. The intuition underlying E911 was clear: It would be desirable for emergency services to be able to locate a victim without searching the entire coverage area of a cell site. However, the technological and sociological impact has far outstripped this intuition over the past 16+ years.

One of the more immediate consequences of E911 is that many cellular handsets now have some form of GPS capability, whether standalone or network-assisted.[29] With it, service providers increasingly recognize that a much broader (and more lucrative) range of location-based services could be provided. However, it should be understood that GPS was not designed with cellphones in mind.[c] GPS

b Notice of Proposed Rulemaking, Docket 94-102, adopted as an official report and order, June 1996;[12] the order and all its subsequent incarnations are referred to as E911 in this article.

c It was designed with guided missiles and bombers in mind.[16]

was intended for outdoor use; the weak signals transmitted from the 24 space vehicles (SVs) that constitute the GPS space segment are difficult to detect indoors and blocked by tall buildings.[16] GPS is also designed to work with autonomous receivers; GPS signals are modulated to provide the receiving unit with the locations and orbits of the SVs, information needed to compute the receiver's location.[d] The locations and orbits are provided on the same carrier used for (civilian) distance estimation. In order to avoid interference, the data rate for these transmissions is slow—only 50bps—so a receiver takes up to 12.5 minutes to obtain all the information it needs to perform a location fix. Networks often assist cellphones by providing this information over much-faster cellular links,[9] but cellphone manufacturers are apparently looking to other means for quick, accurate location fixes for their subscribers.

This brings us to the April 2011 kerfuffle over Apple's and Google's use of cellphones to identify Wi-Fi and cell-tower locations. In testimony before the U.S. Congress's Judiciary Committee's Subcommittee on Privacy, Technology and the Law, Guy Tribble, Apple's vice president for software technology, confirmed what analysts of the consolidated.db file had already determined: Apple iPhones record the MAC address and signal strength[e] for detected access points, then time-stamp and geo-tag that data. The geo-tag consists of a GPS/cell-tower-derived location estimate of the iPhone that has detected the access point. For detected cell sites, the cell-tower ID and signal strength are combined with the detecting iPhone's location estimate.

Tribble provided little technical detail but did suggest that by obtaining such data from a large number of iPhones (crowdsourcing), highly accurate estimates of the location of sites and access points could be determined. With a map of these locations, precise location estimates can be generated for phones that report receiving signals from the cell sites and access points.

A simple analysis makes the point. Consider a data set of n records for a single access point, with each record consisting of the location of a different receiving unit and the strength with which that unit receives the signal from the access point. The location of the access point can be computed by determining the weighted centroid of the measurements.[5] Following creation of a map of the locations of cell sites and access points, a position fix for a cellphone can be computed through trilateration using received signal-strength measurements.

Trilateration is similar to what is performed by GPS receivers, with the added benefit that the distances are much shorter and the access points and cell towers are not moving. Overall, one would expect the resulting location estimates to be at least as good as a GPS fix in urban and residential areas and could be of sufficiently fine granularity as to be able to resolve an individual address.

The presence of consolidated.db in iPhones (a database of time-stamped GPS fixes for the cellphone) gives the appearance that Apple is tracking iPhone users, but Tribble said the "data is extracted from the database, encrypted, and transmitted—anonymously—to Apple over a Wi-Fi connection every 12 hours (or later if the device does not have Wi-Fi access at that time)."

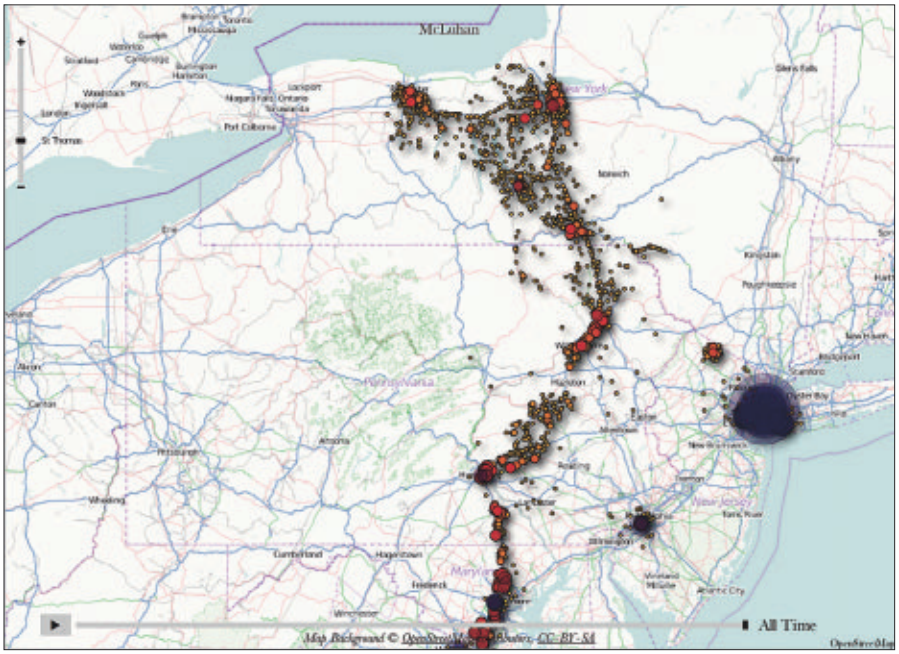
The extent the data is anonymous is questionable without further detail. The author generated the figure here using the consolidated.db database on his iPhone and the iPhone Tracker application developed by Pete Warden.[f] His well-traveled path from Ithaca, NY, to Washington, D.C. (National Science Foundation and Defense Advanced Research Projects Agency) and onward to his parent's house in Virginia Beach, VA, is apparent for all to see. It would take little effort to associate this trace with the author. As the Netflix example covered later suggests, there is more to anonymization than stripping a location trace of its associated phone number and user-account ID.

Personality of Place

The iPhone location trace says a lot about the author, including his predilection for visiting Washington, D.C., New York, and his parents. What would a more fine-grain set of tracking data, like that potentially being

d Detailed SV orbital information is called "ephemeris"; each SV transmits its own ephemeris, along with an almanac providing less-detailed information for all active SVs.

e Signal strength is converted into a "horizontal accuracy number"; Apple does not collect the user-assigned name for the network.



A cellphone's travels; data from consolidated.db in the author's iPhone.

f In an FAQ at http://petewarden.github.com/iPhoneTracker/\#5, Warden noted that the data is actually more accurate than the maps generated by the tool; Warden inserted the intentional dithering to reduce the privacy risk created by the tool while still making apparent the problem with consolidated.db.

made available to LBS by emerging smartphone-location technology, have to say about an individual? Consider the following information, which can be derived through the correlation of fine-grain location data with publicly available information:

Location of your home. What kind of neighborhood do you live in? What is your address? Mortgage balances and tax levies are often available once an address is known. Your socioeconomic status can be deduced;

Location of your friends' homes. What sort of homes do they have? Do you ever spend the night? How often?;

Location of any building you frequent with a religious affiliation. Or do you never frequent such buildings? In either case, beliefs can be deduced;

Locations of the stores you frequent. Your shopping patterns reflect your preferences and in some cases your beliefs or vices;

Locations of doctors and hospitals you visit. Do you visit frequently? How long do you stay? The fact that you have a serious illness is readily determined, as are, in some cases, even the type of illness through your visits to specialty clinics. Such information is of interest to insurance providers and the marketers of pharmaceuticals, among others; and

Locations of your entertainment venues. Do you attend the local symphony? Do your tastes run to grunge rock? Do you frequent bars? What type? One can draw multiple conclusions from the frequency of visits and types of venue.

One could go on with this list. The fact is fine-grain location information can be used to determine a great deal about an individual's beliefs, preferences, and behavior. Databases containing such information pose a threat to individual security and privacy, as they can be a focus for hackers with criminal intent. On a less-malevolent note, such data is immensely valuable to direct-marketing firms. Entire businesses are built around the compilation of lists of such information acquired through other means.

Is this a problem? Isn't LBS data collection simply additional, perhaps redundant, data collection that feeds the ordinary and tedious process of direct marketing? The following sections show this is not the case. LBS supports location-based advertising (LBA), which has the potential for exerting substantially more power over individual behavior than previous modes of advertising.

The continued accumulation of location data may reach a point where a marketer can uniquely match an anonymous location trace to a named record in a separate database.

The work of advertising. As of early 2012, InfoUSA maintained a list of 210 million U.S. consumers for sorting into various categories, including area code, ZIP code, home value/home ownership, housing type, mortgage, personal finance, hobbies and interests, children/grandparents/veterans, ethnicity, religion, and voter information.[g] As seen from the earlier thought experiment, much of it can now be derived by correlating location data with address databases. But data collection through location-based services takes the process of collection to a new level of invasiveness while adding an additional control variable to the process of advertising.

To begin, LBS data collection is able to substantially refine the personal information available from other sources; for example, one may claim to practice yoga, but marketers may now know how frequently one takes classes, pointing to a specific level of enthusiasm previously known only to individuals and their fellow yogis and yoginis.[h] LBS also enables an approach to consumer targeting that goes well beyond previous marketing strategies by collecting information about beliefs, preferences, and behavior while one performs the illustrative practice. A mailing list may indicate one is generally a lover of Italian food, while an LBS may have the additional knowledge that you are currently in an Italian restaurant. To understand why this is important, first consider the "work" that advertising has us do on its behalf.

In her 1978 book on the psychology of advertising, Judith Williamson described advertising as shifting meaning from one semantic network to another.[28] As the book was originally written in the 1970s, Williamson focused on print advertising, with an occasional reference to broadcast television. In

g http://www.infousa.com/

h You may substitute political parties, sporting events, dog shows, or any other personal interest to imagine a more personally relevant example.

a canonical example she pointed to an advertisement that seems simple, a photograph of the iconic French actress Catherine Deneuve juxtaposed with a bottle of perfume (Chanel No. 5), encouraging us to bind to the perfume the association of class and beauty people of a certain age might associate with Catherine Deneuve. Meaning has been shifted from one semantic network (the realm of actresses) to another (a brand of perfume).

LBA has the potential to perform a similar sleight-of-mind, causing us to exchange the meaning we associate with a place for one suggested by an ad. Moreover, this location-based semantic shift is taking place through ads delivered to a device that can track the individual. This raises two new privacy issues: The first is that LBA has the potential to be a feedback system with dynamic control. The advertiser can present an ad when one is near a target location, then track that person to determine whether the ad has had the desired response. In the language of Gilles Deleuze,[8] the advertiser can observe the response to the information stream presented to the individual, then "modulate," or refine, that stream over time, driving the individual to a desired state of behavior; in this case, movement to and consumption at the target location. Primitive examples of modulation fueled by click-tracking can be seen by an aware observer of the Web. If one fails to produce the desired response to a pop-up window, other windows offer alternatives on behalf of the advertiser. Second, unlike click-tracking, LBA exploits consumers' physical location, attempting to manipulate their relationship to their physical surroundings. The following highlights the potential for a more insidious form of manipulation at an entirely new level of psychological conditioning.

Philosophy of place. Many people view geography as the study of locations and facts; for example, "Jackson is the capital of Mississippi" is the stuff of geography, as is the shape and size of the Arabian Peninsula. However, in the 1970s, humanistic geographers began to move the field toward a consideration of "place" as more than a space or location, beyond latitude, longitude, and spatial extent.[7] In an oft-quoted definition, the geographer and political philosopher John Agnew defined place as consisting of three things[1]:

Location. "Where," as defined by, say, latitude and longitude;

Locale, or the shape of the space. Shape may include defining boundaries (such as walls, fences, and prominent geographical features like rivers and trees); and

Sense. One's personal and emotional connections established through location and locale.

Place is thus a location to which one ascribes meaning. The process by which meaning attaches to place, and the importance of this process to the individual and to society, have become a prime focus for humanistic geographers. One aspect of it builds on the work of the phenomenologists. Phenomenology, generally associated with the German philosopher Franz Brentano and the Austrian philosopher Edmund Husserl, studies the structures of consciousness. Phenomenology proceeds by first bracketing-out our assumptions of an outside world, then focusing on our experience of the world through our perception. Phenomenologists study consciousness by focusing on human perception of phenomena, hence the name.[i]

Brentano is credited with one of the key results of the phenomenologist approach. In his 1874 book *Psychology from an Empirical Standpoint*,[j] he said one of the main differences between mental and physical phenomena is the former has intentionality; that is, it is about, or directed at, an object, or one cannot be conscious without being conscious of something. In the latter part of the 20th century, humanistic geographers took this philosophy a step further; in his 1976 book *Place and Placelessness*, Edward Relph asserted that consciousness could only be about something in its place, making place "profound centers of human existence."[23]

Another thread in the philosophy of

The more that can be done within the handset and kept within the handset, the greater the preservation of anonymity.

i For a quick look at the field see http://plato.stanford.edu/entries/phenomenology/ and for more detail Sokolowski, R. *Introduction to Phenomenology*, Cambridge University Press, Cambridge, U.K., 1999.

j *Psychologie vom empirischen Standpunkt*; http://www.archive.org/details/psychologievome00brengoog

place originates with 20th century German philosopher Martin Heidegger who described human existence in terms of *dasein*, a German word that can be translated as "human existence," or perhaps more helpfully as "being there." The important thing for us here is to understand that *dasein* is always in the world.[k] As humans we enter a preexisting world of things and other people and develop our sense of self by (and only by) interacting with them. According to Heidegger, an inauthentic existence is one in which the individual fails to distinguish him or herself from the surrounding crowd and its priorities.

Humanistic geographers have taken up the concept of *dasein*, using it to explore the role of place in human existence. In his 2007 book *Place and Experience: A Philosophical Topography*, Jeff Malpas invoked *dasein* and related concepts of spaciality and agency to show that place is primary to the construction of meaning and society.[l,19]

Using these concepts, this article now aims to characterize the potential impact of LBA, the objective of which is to alter the ever-present, ongoing human process of interaction with the immediate surroundings. LBA attempts to shift intentionality, diverting consciousness from an experience of the immediate surroundings to the consumption of advertised goods. In Heideggerian terms, LBA interferes directly with the individual's project of crafting an authentic existence.

Consider the following situation, developed in two stages: A family is seated at their dining room table enjoying dinner together, but there is an exception—the father, a relentless worker, is reading texts and email messages instead of joining the conversation. One could say he is no longer present. He has left the place. Or to turn it around, as far as the father is concerned, the dinner table is no longer a "place" with familial meaning but merely a location for eating. Now, to complete the example, assume that someone who wants to communicate with the father from afar knows when he is at the table and chooses that time to send texts. The texter now has the ability to disrupt the father's relationship with the family dinner, a relationship often filled with a strong, even defining, sense of meaning.

The dinner table is a natural example for the author,[m] but one might consider a walk through one's hometown, visiting an old high school, or attending a play. LBA has the potential to detract from the experience of these familiar and meaning-filled environs. One's surroundings may thus lose their "placeness" through LBA, including their meaning, and become merely a path to be traversed. As places become locations, meaning is lost to the individual. That is, we lose some of ourselves, as well as one of the critical processes through which we become a self.

Location Anonymity

Having established the importance of location privacy, is it necessary to forgo the benefits of LBS and LBA? Fortunately the answer is no, but it needs to be clear to data collectors that it is not sufficient to simply scrub names and phone numbers from location traces. As AOL[15] and Netflix[21] have learned, supposedly anonymous datasets are often susceptible to correlation attacks in which datasets are associated with individuals through comparison of the datasets to previously collected data. Netflix is particularly instructive; in 2006 it issued a public challenge to develop a better movie-recommendation system.[22] As part of the challenge, it released training data consisting "of more than 100 million ratings from over 480,000 randomly chosen, anonymous customers on nearly 18 thousand movie titles." Within weeks, computer scientists Arvind Narayanan and Vitaly Shmatikov had showed the data was not as anonymous as Netflix might have thought. Narayanan and Shmatikov devised an elegant algorithm that correlated the NetFlix data with other publicly available data and thus identified a number of users in the Netflix training data.[21] Along the way, they developed rules of thumb for such correlation attacks, noting such attacks work well when they emphasize rare attributes and that the winning match should have a much higher score than the second-place match. The first can be understood intuitively; a marketer would learn more from the knowledge that someone has purchased the author's most recent text on error-control coding than from finding that someone has purchased a Harry Potter book. The second rule is equally intuitive, as it is intended to avoid false positives.

Here, these rules are useful for developing a Shannon-theoretic model for correlation attacks on supposedly anonymized location traces. In his 1949 paper "Communication Theory of Secrecy Systems,"[24] Claude Shannon defined unicity distance as the minimum amount of ciphertext needed before uncertainty about a piece of plaintext could be reduced to zero. The translation to the de-anonymization of location traces is clear; the continued accumulation of location data may reach a point where a marketer can uniquely match an anonymous location trace to a named record in a separate database.

The goal in this article is not a specific number as a cutoff for data accumulation or an all-encompassing framework into which all de-anonymizing attacks have a place. Rather, it develops an example model and evaluates its dynamics—how the structure of the model changes as the amount of location data increases—in order to craft design rules for anonymous LBS.

A Shannon-theoretic approach to location anonymity. Let a marketing database S consist of a collection of binary preference vectors $\{\mathbf{X}_i\}$ of length n, where the index i indicates a specific user. The individual vectors have the form

$$\mathbf{x}_i = (x_{i,0}, x_{i,1}, x_{i,2}, \ldots, x_{i,n-1});\ x_{i,j} \in \{0,1,e\}$$

Each coordinate $\mathbf{x}_{i,j}$ is a binary indicator representing the user i's preference with regard to some specific item, belief, or behavior; for example, $\mathbf{x}_{i,0}$ might indicate whether the user likes cats (yes or no), and $x_{i,1}$ might

k This concept has had profound influence on the field of artificial intelligence; for example, Philip Agre explicitly applied Heideggerian thought in moving the practice of computational psychology away from cognition and toward action in the world.[2]

l For a more-focused exploration of place in the thought of Martin Heidegger see Malpas, J. *Heidegger's Topology: Being, Place, World*. MIT Press, Cambridge, MA, 2008.

m He would never be allowed to behave like the father in the example.

indicate feelings about dogs. Some preferences (such as the identity of the user's favorite rugby team) might cover several coordinates, depending on the number of teams that can be represented in the database. If a given preference for a particular user is unknown, the associated coordinate is given the value "*e*" for erasure.[n] The marketer's knowledge concerning a user's beliefs, preferences, and behavior are thus coded into binary vectors of a fixed length (*n*) with a consistent semantic attribution to each coordinate or block of coordinates.

Now let $\mathbf{L}_m$ be a trace of length *m*, a sequence of *m* location fixes generated by a single subscriber

$$\mathbf{L}_m = (l_0, l_1, l_2, \ldots l_{m-1})$$

As discussed earlier, marketers can associate locations with beliefs and preferences, but the amount of information derived clearly varies depending on the type of location in the trace. Now consider a preference mapping *F* that maps location traces to preference vectors while acknowledging such mapping may not be one to one and is situation-dependent. The preference vectors have the same syntactic and semantic structure as the vectors in the marketer's database

$$F : \{\mathbf{L}_m\} \rightarrow \{\mathbf{P}\}$$

$$\mathbf{P} = (p_0, p_1, p_2, \ldots, p_{n-1}); p_j \in \{0, 1, e\}$$

Narayanan and Shmatikov[21] discussed several ways to identify the $\mathbf{X}_i \in S$ that is the best match for a given **P**, thereby (potentially) de-anonymizing **P**. This article takes a somewhat different approach, attempting to characterize the dynamics of the de-anonymization problem as the length of the location trace grows.

Suppose a location trace of length *m* is mapped into a preference vector **P** of length *n*. **P** will have some *t* non-erased coordinates and *n* – *t* erased coordinates. Assume that as *m* increases, *t* increases or remains the same.[o] This follows from the fact that as data collectors obtain more location information, they typically increase their knowledge about the associated individual.

Now consider those vectors in the marketing database for which individual preferences on these *t* coordinates are known. Within *S* there will be some N_m vectors with support for all *t* non-erased coordinates of **P**. These N_m vectors form a subset $C \subset S$. For each vector in *C*, delete all but the *t* coordinates of interest (those corresponding to the non-erased coordinates of **P**). We now have a set C' of N_m vectors of length *t*. The problem of de-anonymization now looks like an error-control coding problem, so which vector in C' provides the closest match to the non-erased coordinates of the preference vector **P**? The ability of the marketing database to distinguish between users can now be expressed (using coding-theoretic terminology) as the minimum distance between the vectors in C'. The minimum distance is the minimum number of coordinates in which any pair of vectors differ. In more compact form, this can be expressed as

$$d_{min} = min_{\mathbf{x}'_i, \mathbf{x}'_{j \in C', i \neq j}} \left| \{k | x_{i,k} \neq x_{j,k}, k \in (1,t)\} \right|$$

The greater the value of d_{min}, the greater the ability of a correlation attack to associate a location trace with a single record, and thus a single individual. When d_{min} is large, the individuals represented by the vectors in C' are readily distinguished from one another. On the other hand, if d_{min} is small or zero (as happens when two or more identical vectors are in C'), then the problem of de-anonymization becomes difficult or even impossible. Marketers are unable to distinguish between the individuals so are unable to determine with which individual to associate a given location trace.

Privacy-aware system designers can now develop rules of thumb for preserving anonymity in the face of correlation attacks by exploring the dynamics of the relationship between location traces $\mathbf{L}_m$, preference vectors **P**, and the minimum distance d_{min} of the corresponding set of vectors C':

- As the length *m* of a location trace $\mathbf{L}_m$ increases, the number of non-erased coordinates of a preference vector **P** increases; the reverse is also the case;
- As the number of non-erased coordinates of **P** increases, the length of the vectors in C' increases, while the cardinality of C' decreases; fewer vectors in *C* will have the requisite support as the number of coordinates requiring support increases. The overall effect is an increase in minimum distance and a corresponding increase in the efficacy of correlation attacks;
- As the number of non-erased coordinates of **P** decreases, the length of the vectors in C' decreases while the cardinality of C' increases; more vectors in *C* have the requisite support, as less support is required. The overall effect is a decrease in minimum distance and in the efficacy of correlation attacks.

It follows both intuitively and analytically that the number of non-erased coordinates in **P** should be kept as small as possible and can be done in either of two ways:

Reduce the length of location traces. If the preference map has less information on which to operate, it generates a preference vector with more erased coordinates;

Reduce the ability of the preference mapping to resolve a location trace into specific coordinate values in a preference vector. This can be done by reducing or eliminating the extent each trace location provides preference-vector information.

System designers can exploit these results to design anonymity-preserving location-based services.

Anonymous LBS. Consider a basic location-based service; call it "The Doppio Detector," giving users directions from their current location to the nearest espresso shop. For it to work, two basic types of information must be brought together: the subscriber's location at an appropriate level of granularity and a geographic database containing the locations of all nearby espresso shops. With it, the server or the user's handset can superimpose the user's location onto a geographic database, then generate directions through a routing algorithm.

The structure of an LBS can thus be generalized as performing two basic functions:

n Narayanan and Shmatikov[21] said most "auxiliary" databases are extremely sparse and would thus contain a large number of erasures.

o While useful for mathematical clarity, this assumption is not needed to support the results, so long as there is a general tendency for *t* to increase with *m*, which is the case as long as the marketer's database is not highly corrupted.

▸ Determine subscriber location to the desired level of granularity; and

▸ Use a database to map the location to the desired information (such as directions to an espresso shop).

Separating these functions clarifies the anonymity problem while opening up the range of available anonymity-preserving techniques. We begin by determining subscriber location. The best means for preserving anonymity is to do an independent GPS fix on a cellphone. The handset may thus acquire an accurate location estimate without releasing any information to the outside world. This is a general theme; the more that can be done within the handset and kept within the handset, the greater the preservation of anonymity.

However, this approach can be slow. If the handset is to download all necessary SV location information from the SVs themselves, the user may have to wait as long as 12.5 minutes, a potentially excruciating delay when one needs caffeine. If the process is to be sped up through provision of constellation information by the cellular service provider, some location information must be leaked to that provider. However, such data can be coarse; the network must know only the cell site that is serving the user to provide the data for SVs that are potentially visible to the handset. Such coarse location information provides relatively little information about the user's beliefs and preferences. Or to use the language of this article's unicity distance analysis, the preference mapping F operating on cell-site information will produce a preference vector with a large number of erased coordinates.

Khoshgozaran and Shahabi[17] suggested another approach to determining location anonymously: use the network to determine the location fix while preventing the network from knowing the subscriber's actual location. The mobile device biases the data used for the location fix by applying a randomly selected transform to the mobile's measurements. When the mobile receives the resulting location fix from the network, it removes the effects of the bias by adjusting the fix accordingly.

It follows from these options that obtaining a location fix of the desired granularity on the handset need not reduce the user's location privacy. However, the second piece of LBS, the mapping function, creates two significant obstacles to maintaining privacy, with the second posing a potential personal security concern:

Using access-point and cell-site location information, service providers are able to obtain location estimates with address-level precision.

Consistent input granularity. The mapping function requires input granularity consistent with the inherent granularity of the query; a user who wants directions to the nearest espresso shop needs directions, beginning with a position with street-level resolution; and

Known location. Many if not most LBS queries involve objects of known, fixed location; for example, a bookstore has a known location and is generally not in motion. A request for directions indicates the requesting cellphone user will probably be at the location sometime soon.

The following paragraphs consider general means for accomplishing the mapping function while retaining a measure of anonymity:

A release of data is said to provide k-anonymity protection "...if the information for each person contained in the release cannot be distinguished from at least k–1 individuals whose information also appears in the release."[25] It seems logical that such protection can be obtained for the LBS mapping function by stripping identifying information from k LBS requests, bundling and submiting them all at once. The LBS server then provides a combined response from which individual users are able to extract information responsive to their specific requests.

But who or what bundles the original k requests? Gruteser and Grunwald[14] suggested a trusted server that bundles and forwards requests on behalf of users, while Ghinita et al.[13] suggested a tamper-proof device on the frontend of an untrusted server that combines queries based on location. However, such approaches fall short of k-anonymity in that there may be side information (such as home location or a known place of business) that would allow the server to disaggregate one or more users from the bundled request. For example, I benefit little from a bundled request if the request

includes my home as a starting point; my request is too easily disaggregated from the bundle.

Bear in mind that system designers need not completely eliminate the transfer of location information; it would be sufficient to reduce the precision of the location information to where the preference mapping gives the attacker or marketer little with which to work.[p] Given the decreasing cost of memory and bandwidth, it is both efficacious and inexpensive to simply blur the location estimate provided with the request for mapping functionality.[q] An LBS user may, for example, submit a request to the Doppio Detector that includes his or her location as "somewhere in downtown Ithaca," rather than a specific address. The server will respond with a map that indicates the locations of all the espresso shops in downtown Ithaca. The user's handset can then use its more precise knowledge of his or her location to determine the nearest espresso shop and generate directions accordingly.

Anonymity can also be preserved by limiting the length m of each location trace. This limitation is accomplished by preventing the LBS from determining which requests, if any, come from a given user.[r] As described in Wicker,[27] public-key infrastructure and encrypted authorization messages can be used to authenticate users of a service without providing their actual identities. Random tags can be used to route responses back to anonymous users. Anonymity for frequent users of an LBS may thus be protected by associating each request with a different random tag. All users of the LBS thus enjoy a form of k-anonymity. Coupled with coarse location estimates or random location offsets, this approach shows great promise for preserving user anonymity while allowing users to enjoy the benefits of location-based services.

Conclusion

The increasing precision of cellular-location estimates is at a critical threshold; using access-point and cell-site location information, service providers are able to obtain location estimates with address-level precision. Compilation of these estimates creates a serious privacy problem, as it can be highly revealing of user behavior, preferences, and beliefs. The subsequent danger to user safety and autonomy is substantial.

To determine the extent to which location data can be anonymized, this article has explored the Shannon-theoretic concept of unicity distance to reveal the dynamics of correlation attacks through which existing data records are used to attribute individual identities to allegedly anonymous information. With this model in mind, it has also laid out rules of thumb for designing anonymous location-based services. Critical to them is maintenance of a coarse level of granularity for any location estimate available to service providers and the disassociation of repeated requests for location-based services to prevent construction of long-term location traces.

Acknowledgments

This work is funded in part by the National Science Foundation TRUST Science and Technology Center and the NSF Trustworthy Computing Program. I gratefully acknowledge the technical and editorial assistance of Sarah Wicker, Jeff Pool, Nathan Karst, Bhaskar Krishnamachari, Kaveri Chaudhry, and Surbhi Chaudhry.

p Privacy-preserving data mining techniques (such as those developed by Evfimievski et al.[11]) may also provide solutions.

q Zang and Bolot[30] used the Shannon-theoretic concept of entropy to show the role of both temporally and spatially coarse data in preserving anonymity, conclusions I corroborate with this analysis.

r This follows Kifer and Machanavajjhala,[18] who said the privacy of an individual is preserved when it is possible to limit the inference of an attacker as to the participation of the individual in the data-generating process.

References

1. Agnew, J.A. *Place and Politics: The Geographical Mediation of State and Society.* Unwin Hyman, London, 1987.
2. Agre, P.E. *Computation and Human Experience.* Cambridge University Press, Cambridge, U.K., 1997.
3. Apple. Q&A on Location Data; http://www.apple.com/pr/library/2011/04/27Apple-Q-A-on-Location-Data.html
4. Bilton, N. 3G Apple iOS devices are storing users location data. *The New York Times* (Apr. 20, 2011).
5. Blumenthal, J., Reichenbach, F., and Timmermann, D. Position estimation in ad hoc wireless sensor networks with low complexity. In *Proceedings of the Second Joint Workshop on Positioning, Navigation, and Communication and First Ultra-Wideband Expert Talk* (Hannover, Germany, Mar. 2005), 41–49.
6. Clarke, R.A. Information technology and dataveillance. *Commun. ACM 31*, 5 (May 1988), 498–512.
7. Cresswell, T. *Place: A Short Introduction.* Wiley-Blackwell, Malden, MA, 2004.
8. Deleuze, G. Postscript on the societies of control. *October 59* (Winter1992), 3–7.
9. Djuknic, G.M., and Richton, R.E. Geolocation and assisted GPS. *Computer 34* (Feb. 2001), 123–125.
10. Durrell, L. *Balthazaar.* Faber & Faber, London, 1960.
11. Evfimievski, A., Srikant, R., Agrawal, R., and Gehrke, J. Privacy-preserving mining of association rules. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, July 23–26). ACM Press, New York, 2002, 217–228.
12. Federal Communications Commission. *Notice of Proposed Rulemaking Docket 94-102.* Washington, D.C., 1994.
13. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., and Tan, K.-L. Private queries in location-based services: Anonymizers are not necessary. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Vancouver, B.C., June 9–12). ACM Press, New York, 2008, 121–132.
14. Gruteser, M. and Grunwald, D. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the First International Conference on Mobile Systems, Applications, and Services* (San Francisco, May 5–8). ACM Press, New York, 2003, 31–42.
15. Hansell, S. AOL removes search data on vast group of Web users. *The New York Times* (Aug. 8, 2006).
16. Kaplan, E.D. *Understanding GPS Principles and Applications.* Artech House Publishers, Boston, 1996.
17. Khoshgozaran, A. and Shahabi, C. Blind evaluation of nearest-neighbor queries using space transformation to preserve location privacy. In *Proceedings of the 10th International Symposium on Spatial and Temporal Databases* (Boston, July 16–18). Springer-Verlag, Berlin, 2007, 239–257.
18. Kifer, D. and Machanavajjhala, A. No free lunch in data privacy. In *Proceedings of the SIGMOD 2011 International Conference on Management of Data* (Athens, June 12–16). ACM Press, New York, 2011, 193–204.
19. Malpas, J. *Place and Experience: A Philosophical Topography.* Cambridge University Press, Cambridge, U.K., 2007.
20. Morrissey, S. *iOS Forensic Analysis for iPhone, iPad, and iPod Touch.* Apress, New York, 2010.
21. Narayanan, A. and Shmatikov, V. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy* (Oakland, CA, May 18–21). IEEE Computer Society Press, Washington, D.C., 2008, 111–125.
22. Netflix Prize Rules; http://www.netflixprize.com//rules
23. Relph, E. *Place and Placelessness.* Routledge Kegan & Paul, London, 1976.
24. Shannon, C. Communication theory of secrecy systems. *Bell System Technical Journal 28*, 4 (Oct. 1949), 656–715.
25. Sweeney, L. k-anonymity: A model for protecting privacy. *International Journal Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 5 (Oct. 2002), 557–570.
26. Tribble, G.B. Testimony of Dr. Guy "Bud" Tribble, Vice President for Software Technology, Apple Inc.; http://judiciary.senate.gov/pdf/11-5-10%20Tribble%20Testimony.pdf
27. Wicker, S.B. Cellular telephony and the question of privacy. *Commun. ACM 54*, 7 (July 2011), 88–98.
28. Williamson, J. *Decoding Advertisements: Ideology and Meaning in Advertising.* Marion Boyars Publishers Ltd., London, 1978.
29. Yoshida, J. Enhanced 911 service spurs integration of GPS into cell phones. *EE Times* (Aug. 16 1999); http://www.eetimes.com/electronics-news/4038635/Enhanced-911-service-spurs-integration-of-GPS-into-cell-phones
30. Zang, H. and Bolot, J. C. Anonymization of location data does not work: A large-scale measurement study. In *Proceedings of the 17th Annual International Conference on Mobile Computing and Networking* (Las Vegas, Sept. 19–23). ACM Press, New York, 2011, 145–156.

Stephen B. Wicker (wicker@ece.cornell.edu) is a professor in the School of Electrical and Computer Engineering of Cornell University, Ithaca, NY, and member of the graduate fields of information science and computer science.

DOI:10.1145/2240236.2240256

Traditional bias toward journals in citation databases diminishes the perceived value of conference papers and their authors.

BY BJORN DE SUTTER AND AÄRON VAN DEN OORD

To Be or Not To Be Cited in Computer Science

THE SUSTAINABILITY AND nonconformism of conferences as premier publication venues in computer science is the subject of intense debate.[3,4,17,18] Evaluating scientists for promotion and budget allocation involves metrics like journal impact factors[14] and h-indexes[7] based on citation counts retrieved from Scopus and the Web of Science (WoS).

Many computer scientists view the historical focus of these databases on journals as a professional disadvantage, even though many conferences have been included in Scopus since 2004 and WoS since September 2008, including older ones entered later.

» key insights

- **The supposedly reliable Scopus and WoS include incomplete citation records for indexed CS journal and conference papers.**
- **Despite the difficulty of automatically retrieving Google Scholar records, these records can still be used to correct the missing records in Scopus and WoS.**
- **Corrected citation counts allow for much fairer evaluation of CS researchers and related conferences.**

Inclusion of proceedings and journals in Scopus and WoS is often viewed as a stamp of approval and relevance. By contrast, databases like CiteSeer[x] and Google Scholar (GS) also cover books, technical reports, and other less-important manuscripts. Moreover, whereas Scopus and WoS are generally viewed as providing correct information, GS is known to include erroneous records.[11]

Higher citation counts than those in Scopus and WoS can be obtained by extending the coverage of a citation count by, say, including citations of non-indexed publications[11] and by combining databases.[10] Despite the need to manually cleanse GS records of erroneous and irrelevant records,[2,5,10,11] GS is useful for extending

coverage as well.[1,4,10,11,18]

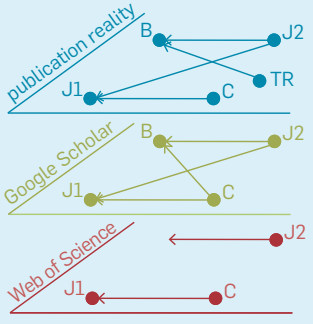
Meho and Rogers[10] concluded in 2008 that choosing WoS or Scopus did not have a significant effect on the citation-based ranking of human-computer interaction researchers. However, in a case study involving library and information researchers, Meho and Yang[11] observed the opposite, finding that conclusions drawn for one scientific domain cannot be generalized to other domains.

Complementing coverage studies, this article explores the inaccuracy of citation records, along with their effect on the perceived impact of CS conferences and on author ranking. Figure 1 outlines the difference between coverage and accuracy for an author of a book B and a journal article J1, with their citations visualized at the top of the figure. B is cited by another article J2 and by a technical report TR. J1 is cited by a conference paper C. TR is a preliminary version of C, with the same title and authors. The list of references in C is shorter than TR, so TR cites B, but C does not cite B.

The middle segment of the figure reflects GS citation records, with GS mistakenly attributing the citations by TR to C. GS also covers publications of lesser importance (such as books). Due to its erroneous and for certain policies irrelevant records, the GS citation count in the example is not reliable.

WoS records are visualized in the bottom segment of the figure. The first observation is that WoS does not index less-important manuscripts (such as TR and B). However, WoS sometimes does keep track of their citations by indexed papers (such as the citation of B by J2). With the manual-count method of Meho and Rogers,[10] these citations can still be counted, should citation-analysis policy demand it. Second, the citations of B by C and of J1 by J2 were never added to WoS. For policies that neglect citations by papers not indexed in WoS, the missing citation of B does not matter, but the missing citation of J1 always matters. Both the citing and cited papers have the WoS stamp of approval, so the citation should be counted. But when for some reason the database lacks a correct record of the citation, as in this example, it is not counted, and the author suffers professionally from undercitation. The study by Meho and Yang[11] on library and information researchers said 0.5%, 4.4%, and 12% of relevant citations were, at the time of the study missing from GS, Scopus, and WoS, respectively, due to database errors.

Figure 1. Publications (vertices) and citations (edges) recorded by databases.



Undercitation in some databases seems to be caused mostly by their use of inferior parsing technology.

Here, we evaluate undercitation resulting from such an error. Complementing the studies mentioned earlier, we recently uncovered a significant undercitation bias in Scopus and WoS against covered CS conferences, demonstrating how it weakens the CS community's effort to win greater appreciation for conference papers. We also found how variations in undercitation of individual authors make the ACM Digital Library (DL), Scopus, and WoS unreliable information sources for citation-based metrics. We also present an automated method that combines the coverage of GS with the quality assurance of Scopus and WoS to detect undercitation resulting from missing citations.

We do not question Scopus or WoS coverage. The analyses we perform for any such database involve only publications indexed in that database. Hence all undercitation results presented here are independent of database coverage. Moreover, we do not take a position for or against citation-based metrics, though their usefulness has been questioned,[12] and many refinements have been proposed.[13,14] Our results demonstrate only that unless a corrective method is used, as we do here, to correct raw counts obtained from Scopus and WoS, their in-

accuracy makes them unsuitable for CS research evaluation.

Relative Relevant Undercitation

To study the accuracy of database citation records, we measure the records' relative relevant undercitation (RRU); the RRU of a database query is the fraction of all (cited, citing) paper pairs for which both cited and citing papers are indexed in the database but for which the database has no record of the citing paper in the cited-by list of the cited paper. This fraction equals the underestimation of the citation count reported by the database within its own coverage; in Figure 1, the citation of J1 by J2 is missing in WoS, but the citation by C is present, resulting in an RRU of 50%.

To compute RRUs, we developed a Python tool for querying six online databases—the ACM DL, CiteSeer^x, DBLP, GS, Scopus, and WoS—by mimicking a researcher manually browsing a database by sending similar HTTP and parsing retrieved (HTML) data. Given a reference list of an author's papers, the tool first queries the databases by title; for papers not found by title, it tries searching by cited author. The search is limited to the papers in the reference list to prevent counting publications by other authors with the same name or initials.[10]

For each paper found in a database, the tool retrieves its cited-by list. In its extended mode of operation, it downloads the BibTeX or EndNote descriptions provided by the database for all entries in that list. In its fast mode the tool instead parses the HTML pages to identify the citing papers. As those pages display information in a less-uniform way than EndNote or BibTeX, the fast mode can produce less-accurate results. However, this mode is considerably faster for most databases than its extended-search mode, as fewer HTTP queries are needed. Most databases try to detect and block seemingly automated querying. To work around this filter, the tool is designed to sleep a random amount of time, say, 25 to 35 seconds between consecutive queries to GS. The result of this first search phase is a list of (cited, citing) paper pairs of citations, each recorded by at least one database. This list constitutes the tool's estimate of an author's publication genuine citation count.

In a second phase, the tool searches all databases for all papers occurring in the list. This search by title automates a search comparable to manual searches in other studies.[11] When both the cited paper and the citing paper of a citation are found in a database, the tool considers that citation relevant for that database. For each relevant citation, the tool searches the cited-by list of the cited paper. When the citing paper is in it, the tool labels the citation as found in the database. In such cases the citation is also included in automated citation counts provided by that database. When the citing paper is not found in the cited-by list, the tool labels the citation as relevant but missing. A database's RRU for a reference list of papers is the number of missing relevant citations divided by number of relevant citations.

As some citations may not be recorded in any searched database, our tool can underestimate RRUs. The risk of underestimation can be avoided with the manual-count method of Meho and Rogers,[10] though their experience suggests their labor-intensive method will not identify a significant number of additional relevant missing citations.

Due to erroneous database records, our tool can also unintentionally overestimate the number of relevant but missing citations in a database, thereby overestimating its RRU; we quantify this potential overestimation later.

Experiments

We used the tool in 2010 and 2011 to perform three complementary experiments: First, we set it to search all aforementioned databases for three authors, using its extended-search mode. Though we focus here on citation accuracy, the experiment also enabled us to compare a database's coverage on the basis of what the three authors would consider their own relevant output. Due to the tool's long running times—several weeks—we were able to study only three authors in the experiment. We next searched GS and WoS for 14 editors-in-chief of various CS transactions published by ACM and the IEEE Computer Society. Using the tool's fast mode, we thus limited searches to publication lists we obtained from DBLP. The experiment was less accurate and covered fewer databases than the first experiment but included many more authors and publications, enabling us to validate the trends we observed in the first experiment. Finally, we performed a similar experiment for GS and WoS for eight ACM and IEEE transactions published from 2000 to 2002 to study the influence of RRU on journal impact factors.

Experiment one. Three colleagues at Ghent University who began publishing around 1990 assembled a reference list of their own peer-reviewed conference and journal publications; Table 1 lists the number of publications in each database. In the Computer Systems Lab at Ghent University, we have permanent access to the ACM DL, and WoS, as well as to various free databases. This experiment was carried out from September 18 to October 13, 2010, when we also had temporary access to Scopus.

GS, Scopus, and WoS provide excellent coverage of journal papers. In line with the findings of others,[4–6,10,15]

Table 1. Number of journal/conference papers in authors' reference lists and in online publication databases.

Author (domain)	Reference	WoS	Scopus	ACM	Google	CiteSeer^x
Koen De Bosschere (compilers, computer architecture)	66/143	57/89	50/54	40/51	57/112	7/38
Bart Dhoedt (distributed computing, networks)	53/187	51/113	44/100	23/47	43/143	3/17
Wilfried Philips (image and video processing)	64/285	56/130	58/106	13/17	58/214	6/26

GS covers more conferences than other academic databases. Unlike some other studies[10] we did not find more extended conference coverage in Scopus compared to WoS. This might have been due to increased coverage in WoS in the studies.

Table 2 lists the citation counts in each database; in line with previous studies, the Scopus and WoS citation counts were only a small fraction of those in GS. Our tool thus relied heavily on the unreliable GS to compute RRUs.

Table 3 contrasts the numbers of relevant citations to the found numbers and are partitioned into four categories—J2J, C2J, J2C, and C2C—to distinguish whether citing and cited publications are journal (J) or conference (C) papers. The last column on the right combines Scopus and WoS, where a citation is considered relevant/found as soon as it is relevant/found in at least one of the two and missing if found in neither. Table 4 lists the h-indexes computed by the tool for the databases; the found h-indexes were based on found citations and the corrected h-indexes on relevant citations. The corrected h-indexes correspond to the h-indexes we would obtain if a database would fix all its missing citations. As the tool gives us a list of missing relevant citations, we requested corrections of citation records through the WoS correction-request form. Most of our requested corrections were applied within weeks. We used the tool to collect the numbers presented here before that correction. As the h-indexes are based on coverage, the corrected h-index in one database may be smaller than the counted h-index in another database.

The large RRUs in ACM, Scopus, and WoS indicate that missing relevant citations are an important cause of undercitation. We also observed a large variation in the RRUs of individual authors, affecting their ranking based on WoS and WoS+Scopus h-indexes. Unlike the ACM DL, which should be able to handle conferences with the same completeness as journals, Scopus and WoS reflect much more undercitation for conferences than for journals. So independent of their coverage, Scopus and WoS put conference-oriented authors at a disadvantage. Worse is their J2C undercitation. Large numbers of J2C citations is one of the strongest arguments for convincing scholars and researchers from non-CS disciplines to value CS conference papers, though it is precisely those citations that are most underestimated. For example, Koen (listed in the tables) might try to convince a promotion committee that conferences should be valued like journals in his domain by pointing to his high #J2C/(#J2J+#J2C) ratio of 43% in WoS. However, this ratio is not nearly as convincing as the 60% he achieved with corrected WoS records.

We see three potential causes for many of the missing citations: First is overcitation in other databases or inclusion of nonexisting citations; the next experiment demonstrates these possibilities occur to a limited degree. The remaining causes are the incorrect parsing of correct references and the occurrence of incorrect and incomplete references in papers, or so-called miscitations. Some papers have been miscited in more than 165 different ways,[16] with more miscitations among non-English names[8] and in papers with more authors.[9] For example, our own work is often miscited because the "De," "van," and "den" are incorrectly treated as middle names or because they are capitalized incorrectly. The RRUs we found in WoS and Scopus are more than an order of magnitude higher than the 0.5% and 4.4% found by Meho and Yang.[11] But that difference is not surprising; of all the scientific disciplines, librarians and information scientists probably produce the most accurate citations. Note, however, that our experiments

Table 2. Total number of citations per author and citations in each database.

Author	Total	WoS	Scopus	ACM	Google	CiteSeer^x
Koen	2280	384 (17%)	531 (23%)	217 (10%)	2156 (95%)	138 (6%)
Bart	1056	278 (26%)	313 (30%)	62 (6%)	842 (80%)	20 (2%)
Wilfried	1937	745 (38%)	975 (50%)	10 (1%)	1370 (71%)	46 (2%)

Table 3. Number of citations per author, category, and database and corresponding RRUs.

		WoS				Scopus				ACM				WoS + Scopus			
		J2J	C2J	J2C	C2C	J2J	C2J	J2C	C2C	J2J	C2J	J2C	C2C	J2J	C2J	J2C	C2C
Koen	relevant	209	205	312	278	201	249	190	226	73	110	142	147	246	288	352	339
	found	101	114	77	92	151	192	75	113	42	65	56	54	180	232	132	179
	RRU	**52%**	**44%**	**75%**	**67%**	**25%**	**23%**	**61%**	**50%**	**42%**	**41%**	**61%**	**63%**	**27%**	**19%**	**63%**	**47%**
Bart	relevant	194	141	127	104	156	122	112	107	24	34	25	26	211	156	149	132
	found	140	88	19	31	131	96	37	49	15	12	14	21	188	144	45	70
	RRU	**28%**	**38%**	**85%**	**70%**	**16%**	**21%**	**67%**	**54%**	**38%**	**65%**	**44%**	**19%**	**11%**	**8%**	**70%**	**47%**
Wilfried	relevant	411	240	211	101	507	323	209	141	6	4	10	9	531	385	275	204
	found	350	239	82	74	453	304	124	94	3	0	4	3	496	369	177	153
	RRU	**15%**	**0%**	**61%**	**27%**	**11%**	**6%**	**41%**	**33%**	**50%**	**100%**	**60%**	**67%**	**7%**	**4%**	**36%**	**25%**

consider only citations recorded by at least one database. So even if the occurrence of incorrect or incomplete references inflates a database's RRU, it apparently did not stop GS or other databases from recording those citations. Undercitation in some databases therefore seems to be caused mostly by their use of inferior parsing technology.

Experiment two. We ran a second experiment in April 2011 for seven editors-in-chief of ACM transactions and seven editors-in-chief of IEEE Computer Society transactions, aiming to validate the previously obtained RRUs on a larger sample set and assess our tool's accuracy in the presence of erroneous GS records. Due to the large amount of data, we ran the tool in its fast mode. Based on 14 reference publication lists obtained from DBLP, the tool collected 36,931 citations for 1,778 papers in GS and WoS, labeling 18,342 citations as relevant to WoS, of which 9,669 were in WoS, and the remaining 8,673 as missing.

Among the 8,673 missing citations, 1,678 cited conference papers were not cited according to WoS. For such papers, the possibility must be considered that conference data was entered into WoS long after the conferences took place and hence after the citing papers were entered. To estimate the likelihood of this potential cause of RRU, we performed an additional check, building on the assumption that if the late entering of conference data is a major cause of RRU, the result would likely be WoS reporting all papers of the covered conference editions as having zero citations. For each of the 14 editors-in-chief, we selected one conference paper with no citations according to WoS and with the most citations according to GS. For these papers, which were published from 1985 to 2009 and covered 445 of the 1,678 suspect citations, we manually verified that at least one paper of the same conference edition was cited at least once according to WoS. The result was positive in terms of finding citations for all 14 conferences, indicating that late entering of conference data is not likely a major cause of RRU in WoS. Even if late entering of data was a significant cause of undercitation in WoS, similar late entering apparently did not prevent GS from being more complete.

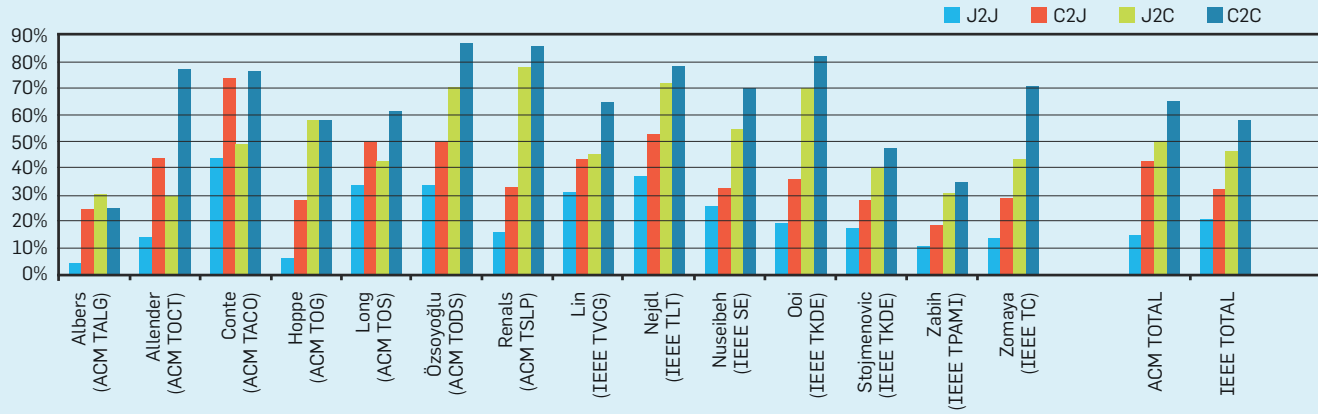
To estimate how much erroneous GS records inflate the RRU in WoS, we manually checked 15 randomly selected citations per author the tool had labeled as missing from WoS. From these $14 \times 15 = 210$ supposedly missing citations, 19 had been labeled incorrectly as such; the corresponding 95%-confidence interval based on the normal approximation is $9.5 \pm 3.9\%$. To compensate for this overestimation, we corrected the number of missing relevant citations as reported by the tool with 9.5% for computing the RRUs reported later in this article.

Most incorrect labels resulted from confusing multiple manuscripts with the same title and authors. Whereas the tool's fast mode was responsible for the confusion, the more extended-search mode as used in the first experiment would have prevented the error. However, in most cases of incorrect labeling, GS simply provided incorrect citation information. In the majority of such cases, GS provided a link to the citing document on CiteSeer^x. We inspected the .pdf documents cached on CiteSeer^x, discovering they indeed cited the cited paper. We also discovered these .pdfs are not from the published conference or journal papers CiteSeer^x claimed them to be but rather from technical reports and Ph.D. theses with the same title and authors but with longer reference lists due to lack of a page limit on the documents. While Google provides little public documentation on its information sources, the correlation between the errors in CiteSeer^x and GS points in the direction of CiteSeer^x as

Table 4. Found/corrected h-indexes for the various databases.

Author	WoS	Scopus	WoS + Scopus	ACM	Google	CiteSeer^x
Koen	12/16	13/17	15/19	8/14	24/25	7/7
Bart	9/12	8/11	11/12	4/6	13/15	3/4
Wilfried	13/15	15/17	16/18	2/4	20/20	4/5

Figure 2. WoS RRUs for 14 editors-in-chief.



the culprit for a considerable fraction of overcitation in GS. Whatever the cause, however, this overcitation with 9.5% is much smaller than the undercitation we found for other databases.

The conclusions of our first experiment remain valid, as confirmed in Figure 2; also, for the editors-in-chief, there was significant varying RRU for all types of citations, and the RRU was particularly large for J2C and C2C citations. Figure 3 visualizes the h-indexes and h-cores of the 14 editors-in-chief. An h-core consists of an author's *x* papers each cited *x* or more times, with *x* being the author's h-index.[7] The bars in Figure 3a represent the h-indexes computed on found citations in WoS, and the bars in Figure 3b represent corrected h-indexes based on relevant citations, confirming h-indexes based on found citations suffer significantly from undercitation. Some authors suffer more than others, to the point their ranking is altered significantly; for example, Ooi was in next-to-last place according to uncorrected WoS citation counts but in third place after correction.

Figure 3. WoS h-indexes and h-cores for 14 editors-in-chief.

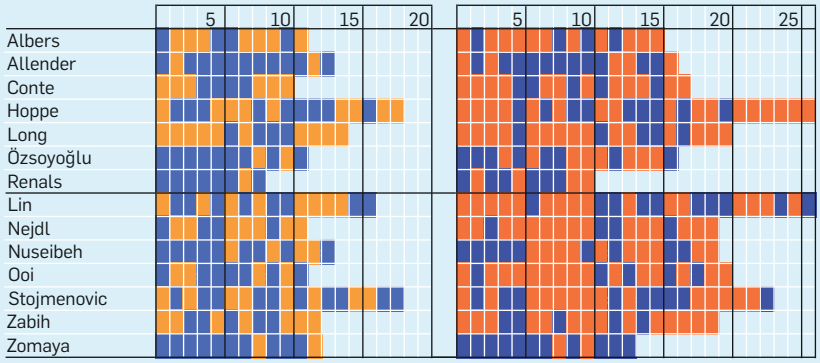


Figure 4. WoS RRUs for articles in eight transactions volumes, 2000–2002.

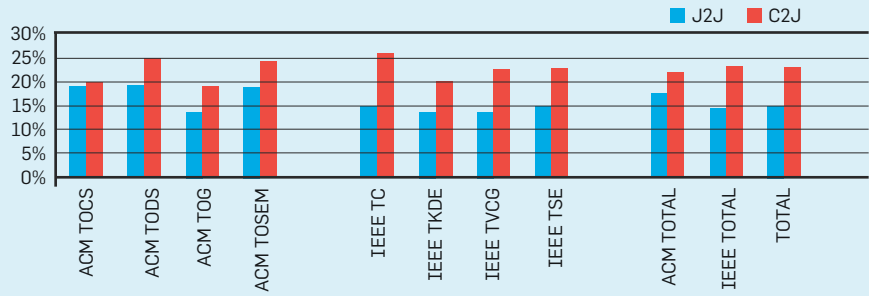
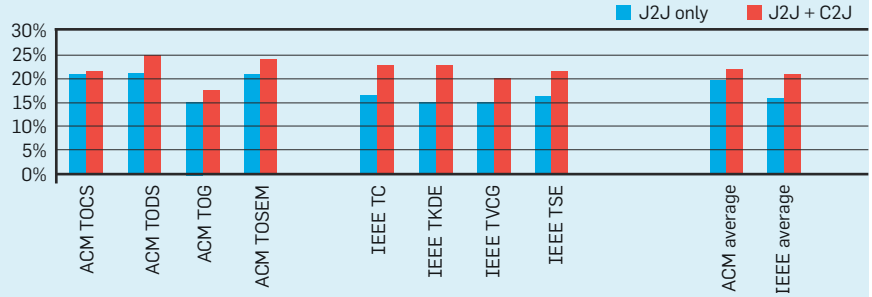


Figure 5. Underestimation of average journal impact factors, 2002–2003.



Figures 3a and 3b also show the contribution of conference papers to h-indexes. Each blue/orange box indicates a journal/conference paper in the h-core, ordered left to right from most cited to least cited. For example, Albers has an h-index of 11 in WoS; of the 11 papers in her h-core, the first, fifth, sixth, and 10th most-cited are journal papers. Based on WoS counts, 43% of all papers in the h-cores are conference papers, and based on the corrected counts, 60% are conference papers. Of the five most-cited papers per author, WoS reports 36% are conference papers, whereas the corrected citation counts report 56% are conference papers. These numbers are much higher than those presented by Bar-Ilan[1] as obtained from WoS in late 2008/early 2009. This higher count might result from increased coverage in WoS from 2009 to 2011, though we could not verify this conclusion. These results again confirm that using uncorrected WoS citation counts to estimate the importance of conferences for an author can lead to significant underestimation for that author. Moreover, such underestimation varies significantly from author to author; for Zomaya in Figure 2 and Figure 3, WoS attributes 2/12 h-core papers to conferences, which is close to the corrected number of 2/13. However, for Ooi in Figure 2 and Figure 3, WoS attributes 4/11 to conferences, which does not even approximate the corrected number 15/20. We conclude that GS should be used as a complementary source of information to obtain accurate citation counts, even when policy stipulates the citation analysis is limited to WoS coverage. Though this second experiment did not include the ACM DL or Scopus, this conclusion can be extended to those databases, of which the first experiment revealed compa-

rable levels of undercitation.

Experiment three. We applied a similar search for the articles published in eight ACM and IEEE transactions from 2000 to 2002, selecting these years to allow collection of citations over a significant period of time and to include four ACM and four IEEE transactions from related, largely overlapping domains. We excluded editorials and republished proceedings to ensure a fair comparison. For the 135 ACM articles and 770 IEEE articles in the considered volumes, our tool's fast mode collected 42,658 citations in GS and WoS in April 2011. Before applying a correction with 9.5%, our tool labeled 19,215 citations as found and relevant to WoS, and 5,193 as relevant but missing.

Figure 4 outlines the resulting, corrected RRUs, which are comparable to those of the other experiments, but the underestimation of C2J citations is over 10% less than what we observed in the first two experiments. The RRUs also reflect considerably less variation, suggesting C2J citations of the selected ACM and IEEE transactions are recorded more accurately in WoS than are the citations of other journals in which the sampled authors were published.

For J2J citations our tool confirmed much less variation between the ACM and the IEEE transactions than we observed for individual authors in the first two experiments. However, based on a T-test, the difference in average J2J RRU between ACM and IEEE is statistically significant; IEEE transactions papers are undercited less than ACM transactions papers. To determine whether this difference might have resulted from differences in citation formats, policies, or culture between ACM and IEEE, we analyzed the sources of the citations. While this analysis indicates ACM and IEEE papers favor citing within their own organization, the numbers were inconclusive with respect to the cause of the different RRUs.

Finally, we studied the effect of undercitation on journal impact factors by computing their average underestimation for 2002 and 2003. These impact factors are based on citations of papers published from 2000 to 2002, the years for which our tool crawled the databases. We used two methods: one in which only J2J citations of full articles (excluding letters, editorials, and republished proceedings) were counted, and one in which J2J and C2J were counted. For each method, we computed the impact factors based on citations the tool found in WoS and on corrected WoS counts. Including the C2J counts resulted in impact factors between 2.37x and 4.35x higher, averaging 3.63x higher for ACM transactions and 3.39x for IEEE transactions. Figure 5 outlines the underestimation of the impact factors resulting from missing citations. In line with the previous results, the underestimation was significant (15%–21%) when only J2J citations are counted, and higher (17%–25%) when C2J citations are included. Due to their higher RRU in WoS, the tool reported the impact factors for ACM transactions are underestimated considerably more than those for IEEE transactions.

These results indicate it is in the interest of ACM and IEEE to include C2J citations to compute journal impact factors and ensure they are recorded more accurately.

Conclusion

ACM, Scopus, and WoS must develop better reference-parsing technology to fix the significant undercitation in their databases. Due to the variations in undercitation, the ACM DL, Scopus, and WoS, and even combinations thereof, are unreliable information sources for the most commonly used citation-based metrics in CS. Moreover, Scopus and WoS databases reflect a significant bias against covered conference proceedings, resulting in underestimation of their impact.

Supposedly unreliable, broad databases like GS can be used to identify and correct undercitation problems in Scopus and WoS without undermining the virtues of their selective inclusion of high-impact conferences and journals. However, due to the inherently slow access to databases like GS, even an automated tool like ours is slow, to the point of being not generally applicable.

Finally, we found a correlation between transactions publishers and their transactions' undercitation, to the disadvantage of ACM. C

References

1. Bar-Ilan, J. Web of Science with the Conference Proceedings Citation Indexes: The case of computer science. *Scientometrics 83*, 3 (June 2010), 809–824.
2. Bar-Ilan, J. Which h-index? A comparison of WoS, Scopus, and Google Scholar. *Scientometrics 74*, 2 (Feb. 2008), 257–271.
3. Birman, K. and Schneider, F. Program committee overload in systems. *Commun. ACM 52*, 5 (May 2009), 34–37.
4. Freyne, J., Coyle, L., Smyth, B., and Cunningham, P. Relative status of journal and conference publications in computer science. *Commun. ACM 53*, 11 (Nov. 2010), 124–132.
5. García-Pérez, M. Accuracy and completeness of publication and citation records in the Web of Science, PsycINFO, and Google Scholar: A case study for the computation of h-indices in psychology. *Journal of the American Society for Information Science and Technology 61*, 10 (Oct. 2010), 2070–2085.
6. Harzing, A.-W. and van der Wal, R. Google Scholar as a new source for citation analysis. *Ethics in Science and Environmental Politics 8*, 1 (June 2008), 61–73.
7. Hirsch, J. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America 102*, 46 (Nov. 2005), 16569–16572.
8. Kotiaho, J. Papers vanish in mis-citation black hole. *Nature 398*, 6722 (Mar. 1999), 19–19.
9. Kotiaho, J., Tomkings, J., and Simmons, L. Unfamiliar citations breed mistakes. *Nature 400*, 6742 (July 1999), 307–307.
10. Meho, L. and Rogers, Y. Citation counting, citation ranking, and h-index of human-computer interaction researchers: A comparison of Scopus and Web of Science. *Journal of the American Society for Information Science and Technology 59*, 11 (Sept. 2008), 13–28.
11. Meho, L. and Yang, K. A new era in citation and bibliometric analyses: Web of Science, Scopus, and Google Scholar. *Journal of the American Society for Information Science and Technology 58*, 13 (Nov. 2007), 2015–2125.
12. Meyer, B., Choppy, C., Staunstrup, J., and Van Leeuwen, J. Research evaluation for computer science. *Commun. ACM 52*, 4 (Apr. 2009), 31–34.
13. Moed, H. and Van Leeuwen, T. Improving the accuracy of Institute for Scientific Information's journal impact factors. *Journal of the American Society for Information Science 46*, 6 (July 1995), 461–467.
14. Moed, H., De Bruin, R., and Van Leeuwen, T. New bibliometric tools for the assessment of national research performance: Database description, overview of indicators, and first applications. *Scientometrics 33*, 3 (July 1995), 381–422.
15. Noruzi, A. Google Scholar: The new generation of citation indexes. *Libri 55*, 4 (Dec. 2005), 170–180.
16. Price, N. What's in a name (or a number or a date)? *Nature 395*, 6702 (Oct. 1998), 538–538.
17. Vardi, M. Conferences vs. journals in computing research. *Commun. ACM 52*, 5 (May 2009), 5–5.
18. Wainer, J., Goldenstein, S., and Billa, C. Invisible work in standard bibliometric evaluation of computer science. *Commun. ACM 54*, 5 (May 2011), 141–146.

Bjorn De Sutter (bjorn.desutter@elis.ugent.be) is a professor in the Computer Systems Lab of the Department of Electronics and Information Systems at Ghent University, Ghent, Belgium.

Aäron van den Oord (aaron.vandenoord@elis.ugent.be) is a Ph.D. student in the Computer Systems Lab of the Department of Electronics and Information Systems at Ghent University, Ghent, Belgium.

DOI:10.1145/2240236.2240257

Using real event data to X-ray business processes helps ensure conformance between design and reality.

BY WIL VAN DER AALST

Process Mining

RECENT BREAKTHROUGHS IN process mining research make it possible to discover, analyze, and improve business processes based on event data. Activities executed by people, machines, and software leave trails in so-called event logs. What events (such as entering a customer order into SAP, a passenger checking in for a flight, a doctor changing a patient's dosage, or a planning agency rejecting a building permit) have in common is that all are recorded by information systems. Data volume and storage capacity have grown spectacularly over the past decade, while the digital universe and the physical universe are increasingly aligned. Business processes thus ought to be managed, supported, and improved based on event data rather than on subjective opinions or obsolete experience. Application of process mining in hundreds of organizations worldwide shows that managers and users alike tend to overestimate their knowledge of their own processes. Process mining results can thus be viewed as X-rays revealing what really goes on inside processes and can be used to diagnose problems and suggest proper treatment. The practical relevance of process mining and related interesting scientific challenges make process mining a hot topic in business process management (BPM). This article offers an introduction to process mining by discussing the core concepts and applications of the emerging technology.

Process mining aims to discover, monitor, and improve real processes by extracting knowledge from event logs readily available in today's information systems.[1,2] Although event data is everywhere, management decisions tend to be based on PowerPoint charts, local politics, or management dashboards rather than on careful analysis of event data. The knowledge hidden in event logs cannot be turned into actionable information. Advances in data mining made it possible to find valuable patterns in large datasets and support complex decisions based on the data. However, classical data mining problems (such as classification, clustering, regression, association rule learning, and sequence/episode mining) are not process-centric. Therefore, BPM approaches tend to resort to handmade models, and process mining research aims to bridge the gap between data mining and BPM. Metaphorically, process

» key insights

- **Although large organizations appreciate the value of big data, they rarely connect event data to process models; process mining represents the missing link between analysis of big data and business process management.**
- **Aligning event data and process models is essential for conformance checking and performance analysis; additionally, relating events to process models helps breathe life into otherwise static diagrams.**
- **The exponential growth of event data and the need for smarter, leaner processes motivates use of process mining.**

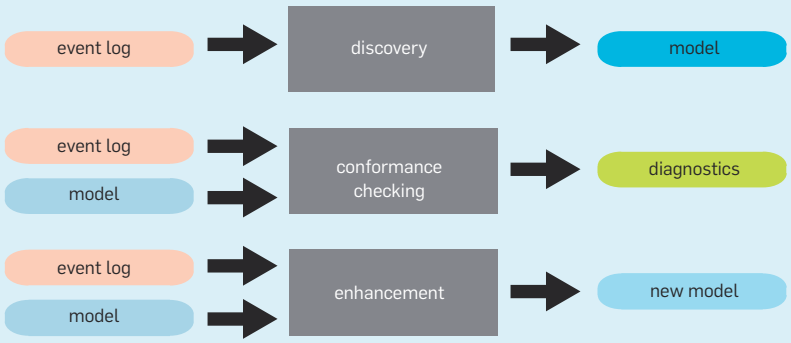
mining can be seen as taking X-rays to help diagnose/predict problems and recommend treatment.

An important driver for process mining is the incredible growth of event data[4,5] in any context—sector, economy, organization, and home—and system that logs events. For less than $600, one can buy, say, a disk drive with the capacity to store all of the world's music.[5] A 2011 study by Hilbert and Lopez[4] found that storage space worldwide grew from 2.6 optimally compressed exabytes (2.6×10^{18}B) in 1986 to 295 compressed exabytes in 2007. In 2007, 94% of all information storage capacity on Earth was digital, with the other 6% in the form of books, magazines, and other nondigital formats; in 1986, only 0.8% of all information-storage capacity was digital. These numbers reflect the continuing exponential growth of data.

The further adoption of technologies (such as radio frequency identification, location-based services, cloud computing, and sensor networks) will accelerate the growth of event data. However, organizations have problems using it effectively, with most still diagnosing problems based on fiction (such as PowerPoint slides and Visio diagrams) rather than on facts (such as event data). This is illustrated by the poor quality of process models in practice; for example, over 20% of the 604 process diagrams in SAP's reference model have obvious errors and their relation to actual business processes supported by SAP is unclear.[6] It is thus vital to turn the world's massive amount of event data into relevant knowledge and reliable insights—and this is where process mining can help.

The growing maturity of process mining is illustrated by the Process Mining Manifesto[9] released earlier this year by the IEEE Task Force on Process Mining (http://www.win.tue.nl/ieeetfpm/) supported by 53 organizations and based on contributions from 77 process-mining experts. The active contributions from end users, tool vendors, consultants, analysts, and researchers highlight the significance of process mining as a bridge between data mining and business process modeling.

Figure 1. The three basic types of process mining in terms of input and output.



The starting point for process mining is an event log in which each event refers to an activity, or well-defined step in some process, and is related to a particular case, or process instance. The events belonging to a case are ordered and can be viewed as one "run" of the process. Event logs may also store additional information about events; when possible, process mining techniques use extra information (such as the resource, person, or device executing or initiating the activity), the timestamp of the event, and data elements recorded with the event (such as the size of an order).

Event logs can be used to conduct three types of process mining (see Figure 1).[1] The first and most prominent is discovery; a discovery technique takes an event log and produces a model without using a priori information. For many organizations it is surprising that existing techniques are able to discover real processes based only on example behaviors recorded in event logs. The second type is conformance, where an existing process model is compared with an event log of the same process. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa. The third type is enhancement, where the idea is to extend or improve an existing process model using information about the actual process recorded in an event log. Whereas conformance checking measures alignment between model and reality, this third type of process mining aims to change or extend the a priori model; for instance, using timestamps in the event log, one can extend the model to show bottlenecks, service levels, throughput times, and frequencies.

Process Discovery

The goal of process discovery is to learn a model based on an event log. Events can have all kinds of attributes (such as timestamps, transactional information, and resource usage) that can be used for process discovery. However, for simplicity, we often represent events by activity names only. That way, a case, or process instance, can be represented by a trace describing a sequence of activities. Consider, for example, the event log in Figure 2 (from van der Aalst[1]), which contains 1,391 cases, or instances of some reimbursement process. There are 455 process instances following trace *acdeh*, with each activity represented by a single character: *a* = register request, *b* = examine thoroughly, *c* = examine casually, *d* = check ticket, *e* = decide,

f = reinitiate request, g = pay compensation, and h = reject request. Hence, trace *acdeh* models a reimbursement request that was rejected after a registration, examination, check, and decision step; 455 cases followed this path, which consists of five steps, so the first line in the table corresponds to $455 \times 5 = 2{,}275$ events. The whole log consists of 7,539 events.

Process-discovery techniques produce process models based on event logs (such as the one in Figure 2); for example, the classical α-algorithm produces model M_1 for this log. This process model is represented as a Petri net consisting of places and transitions. The state of a Petri net, or "marking," is defined by the distribution of tokens over places. A transition is enabled if each of its input places contains a token; for example, a is enabled in the initial marking of M_1, because the only input place of a contains a token (black dot). Transition e in M_1 is enabled only if both input places contain a token. An enabled transition may fire, thereby consuming a token from each of its input places and producing a token for each of its output places. Firing a in the initial marking corresponds to removing one token from start and producing two tokens, one for each output place. After firing a, three transitions—b, c, and d—are enabled. Firing b disables c because the token is removed from the shared input place (and vice versa). Transition d is concurrent with b and c; that is, it can fire without disabling another transition. Transition e becomes enabled after d and b or c have occurred. By executing e, three transitions—f, g, and h—become enabled; these transitions are competing for the same token, thus modeling a choice. When g or h is fired, the process ends with a token in place *end*. If f is fired, the process returns to the state just after executing a. Note that transition d is concurrent with b and c. Process mining techniques must be able to discover such advanced process patterns and should not be restricted to simple sequential processes.

Checking that all traces in the event log can be reproduced by M_1 is easy. The same does not hold for the second process model in Figure 2, as M_2 is able to reproduce only the most frequent trace *acdeh*. The model does not fit the log well because observed traces (such as *abdeg*) are not possible according to M_2. The third model is able to reproduce the entire event log, but M_3 also allows for traces (such as *ah* and *adddddddg*). M_3 is therefore considered "underfitting"; too much behavior is allowed because M_3 clearly overgeneralizes the observed behavior. Model M_4 is also able to reproduce the event log, though the model simply encodes the example traces in the log; we call such a model "overfitting," as the model does not generalize behavior beyond the observed examples.

In recent years, powerful process mining techniques have been developed to automatically construct a suitable process model, given an event log. The goal is to construct a simple model able to explain most observed behavior without overfitting or underfitting the log.

Conformance Checking

Process mining is not limited to process discovery; the discovered process is just the starting point for deeper analysis. Conformance checking and enhancement relate model and log, as in Figure 1. The model may have been made by hand or discovered through process discovery. In conformance checking, the modeled behavior and the observed behavior, or event log, are compared. When checking the conformance of M_2 with respect to the log in Figure 2, only the 455 cases following *acdeh* can be replayed from beginning to end. If the model would try to replay trace *acdeg*, it would get stuck after executing *acde* because g is not enabled. If it would try to replay trace *adceh*, it would get stuck after executing the first step because d is not (yet) enabled.

Among the approaches to diagnosing and quantifying conformance is one that looks to find an optimal alignment between each trace in the log and the most similar behavior in the model. Consider, for example, process model M_1, a fitting trace $\sigma 1 = adceg$, a non-fitting trace $\sigma 2 = abefdeg$, and the following three alignments:

$$\gamma_1 = \begin{array}{|c|c|c|c|c|}\hline a & d & c & e & g \\ \hline a & d & c & e & g \\ \hline \end{array}$$

and

$$\gamma_2 = \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline a & b & \gg & e & f & d & \gg & e & g \\ \hline a & b & d & e & f & d & b & e & g \\ \hline \end{array}$$

and

$$\gamma_3 = \begin{array}{|c|c|c|c|c|c|c|}\hline a & b & e & f & d & e & g \\ \hline a & b & \gg & \gg & d & e & g \\ \hline \end{array}$$

$\gamma 1$ shows perfect alignment between $\sigma 1$ and M1; all moves of the trace in the event log (top part of alignment) can be followed by moves of the model (bottom part of alignment). $\gamma 2$ shows an optimal alignment for trace $\sigma 2$ in the event log and model M_1; the first two moves of the trace in the event log can be followed by the model. However, e is not enabled after executing only a and b. In the third position of alignment $\gamma 2$, a d move of the model is not synchronized with a move in the event log. This move in just the model is denoted as $(\gg,d)$, signaling a conformance problem. In the next three moves model and log agree. The seventh position of alignment $\gamma 2$ involves a move in the model that is not also in the log: $(\gg,b)$. $\gamma 3$ shows another optimal alignment for trace $\sigma 2$. In $\gamma 3$ there are two situations where log and model do not move together: $(e,\gg)$ and $(f,\gg)$. Alignments $\gamma 2$ and $\gamma 3$ are both optimal if the penalties for "move in log" and "move in model" are the same. Both alignments have two $\gg$ steps, and no alignments are possible with fewer than two $\gg$ steps.

Conformance may be viewed from two angles: either the model does not capture real behavior (the model is wrong) or reality deviates from the desired model (the event log is wrong). The first is taken when the model is supposed to be descriptive, or captures or predicts reality; the second is taken when the model is normative, or used to influence or control reality.

Various types of conformance are available, and creating an alignment between log and model is just the starting point for conformance checking.[1] For example, various fitness (the ability to replay) metrics are available for determining the conformance of a business process model; a model has fitness 1 if all traces can be replayed from begin to end, and a model has fitness 0 if model and event log "disagree" on all events. In Figure 2, process models M_1, M_3, and M_4

have a fitness of 1, or perfect fitness, with respect to the event log. Model M_2 has a fitness 0.8 for the event log consisting of 1,391 cases. Intuitively, this means 80% of the events in the log are explained by the model. Our experience with conformance checking in dozens of organizations shows real-life processes often deviate from the simplified Visio or PowerPoint representations traditionally used by process analysts.

Model Enhancement

A process model can be extended or improved through alignment between event log and model, and a non-fitting process model can be corrected through the diagnostics provided by the alignment. If the alignment contains many $(e, \gg)$ moves, it might make sense to allow for skipping activity e in the model. Moreover, event logs may contain information about resources,

Figure 2. An event log and four potential process models—M_1, M_2, M_3, and M_4—aiming to describe observed behavior.

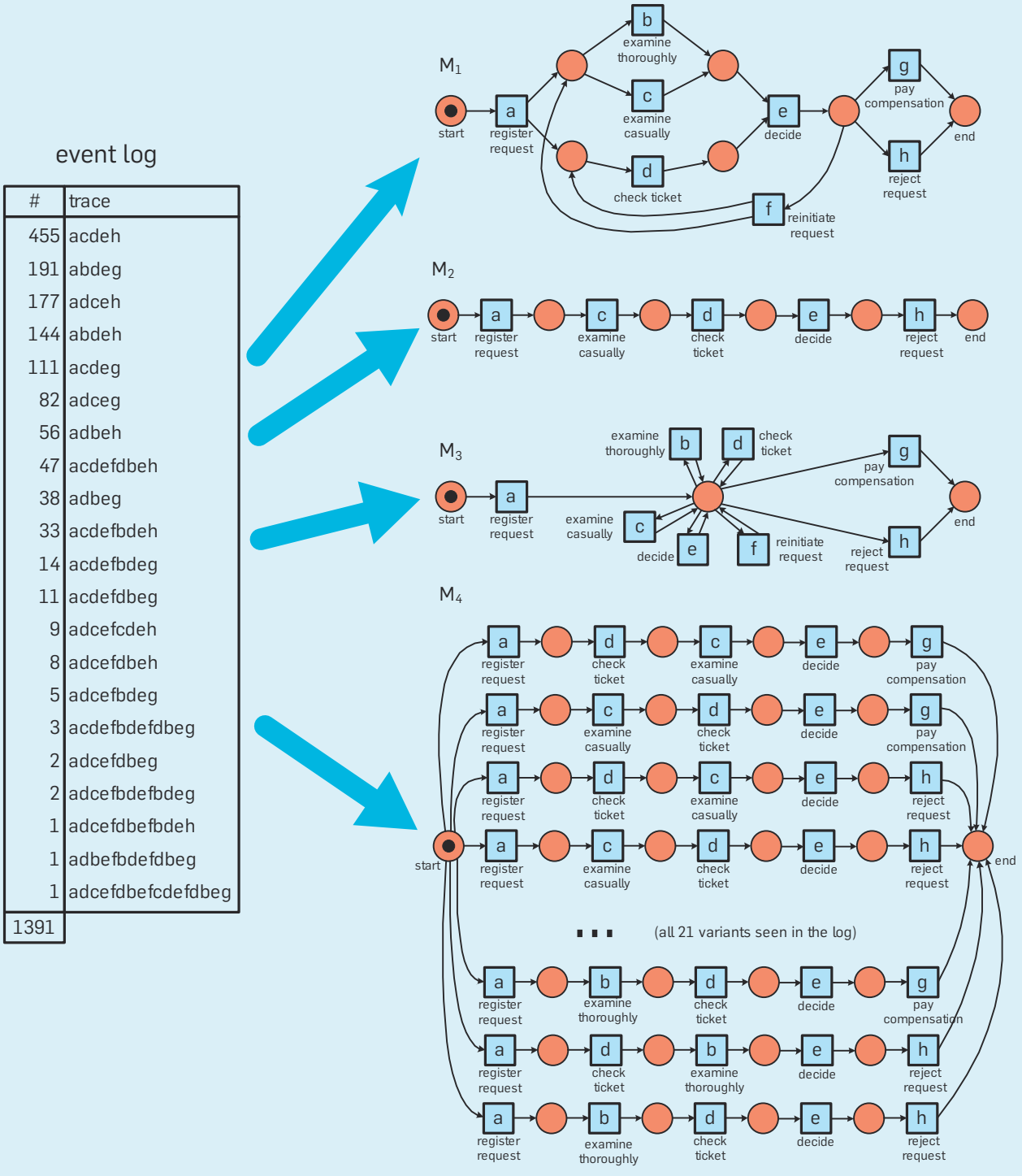
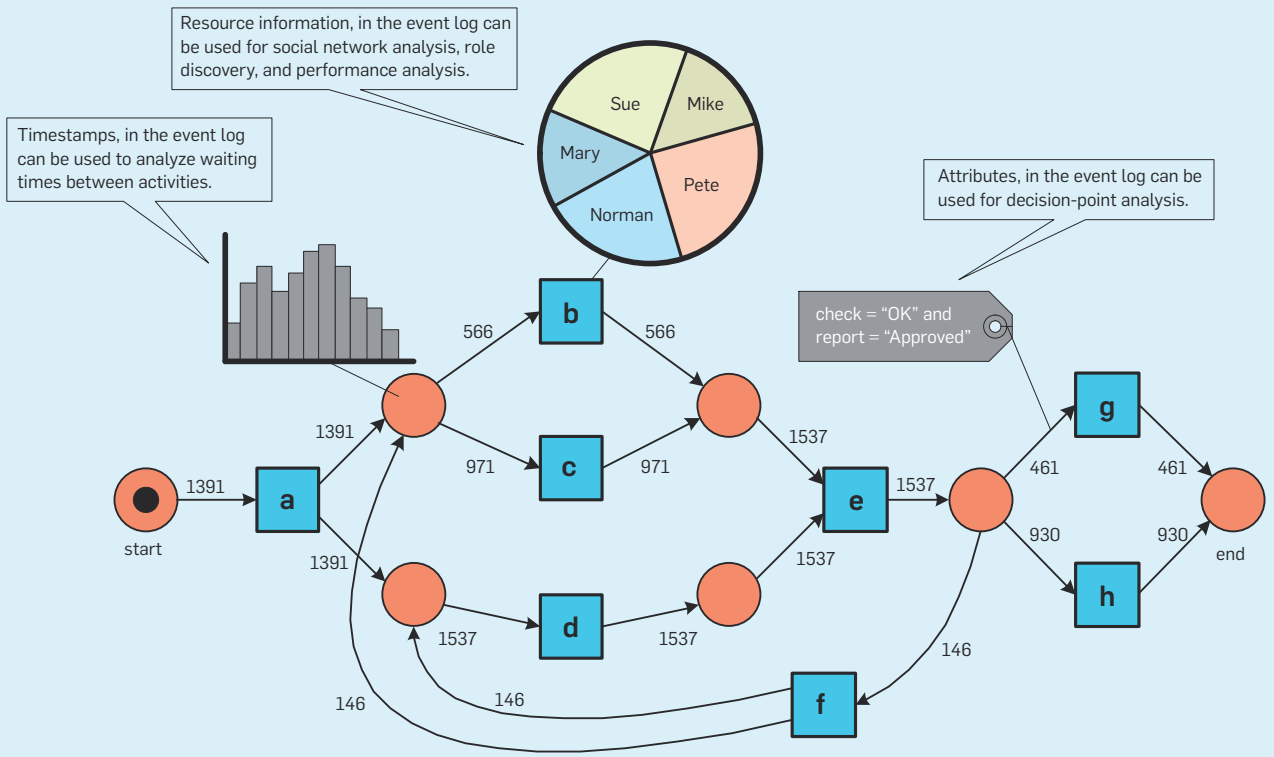


Figure 3. The process model can be extended using event attributes (such as timestamps, resource information, and case data); the model also shows frequencies, as in, say, 1,537 times a decision was made and 930 cases were rejected.



timestamps, and case data; for example, an event referring to activity "register request" and case "992564" may also have attributes describing the person registering the request (such as "John"), time of the event (such as "30-11-2011:14.55"), age of the customer (such as "45"), and claimed amount (such as "650 euro"). After aligning model and log the event log can be replayed on the model; while replaying, one can analyze these additional attributes; Figure 3 shows, for example, it is possible to analyze wait times between activities. Measuring the time difference between causally related events and computing basic statistics (such as averages, variances, and confidence intervals) makes it possible to identify the main bottlenecks.

Information about resources can help discover roles, or groups of people executing related activities frequently, through standard clustering techniques. Social networks can be constructed based on the flow of work, and resource performance (such as the relation between workload and service times) can be analyzed. Standard classification techniques can be used to analyze the decision points in the process model; for example, activity *e* ("decide") has three possible outcomes: "pay," "reject," and "redo." Using data known about the case prior to the decision, a decision tree can be constructed explaining the observed behavior.

Figure 3 outlines why process mining is not limited to control-flow discovery. Moreover, process mining is not limited to offline analysis and can be used for predictions and recommendations at runtime; for example, the completion time of a partially handled customer order can be predicted through a discovered process model with timing information.

Practical Value

Here, I focus on the practical value of process mining. As mentioned earlier, process mining is driven by the continuing exponential growth of event-data volume; for example, according to McKinsey Global Institute in 2010 enterprises stored more than seven exabytes of new data on disk drives, while consumers stored more than six exabytes of new data on such devices as PCs and notebooks.[5]

The remainder of the article explores how process mining provides value, referring to case studies that used our open-source software package ProM (http://www.processmining.org)[1] created and maintained by the process-mining group at Eindhoven University of Technology, though other research groups have contributed, including the University of Padua, Universitat Politècnica de Catalunya, University of Calabria, Humboldt-Universität zu Berlin, Queensland University of Technology, Technical University of Lisbon, Vienna University of Economics and Business, Ulsan National Institute of Science and Technology, K.U. Leuven, Tsinghua University, and University of Innsbruck. Besides ProM, approximately 10 commercial software vendors worldwide develop and distribute process-mining software, often embedded in larger tools

(such as Pallas Athena, Software AG, Futura Process Intelligence, Fluxicon, Businesscape, Iontas/Verint, Fujitsu, and Stereologic).

Provides insight. For the past 10 years we have used ProM in more than 100 organizations, including municipalities (such as Alkmaar, Harderwijk, and Heusden), government agencies (such as Centraal Justitieel Incasso Bureau, the Dutch Justice Department, and Rijkswaterstaat), insurance-related agencies (such as UWV), banks (such as ING Bank), hospitals (such as AMC Hospital in Amsterdam and Catharina hospital in Eindhoven), multinationals (such as Deloitte and DSM), high-tech system manufacturers and their customers (such as ASML, Philips Healthcare, Ricoh, and Thales), and media companies (such as Winkwaves). For each, we discovered some of their processes based on the event data they provided, with discovered processes often surprising even the stakeholders. The variability of processes is typically much greater than expected. Such insight represents tremendous value, as unexpected differences often point to sources of waste and mismanagement.

Improve performance. Event logs can be replayed on discovered or handmade process models to support conformance checking and model enhancement. Since most event logs contain timestamps, replay can be used to extend the model with performance information.

Figure 4 includes some performance-related diagnostics that can be obtained through process mining. The model was discovered based on 745 objections raised by citizens against the so-called Waardering Onroerende Zaken, or WOZ, valuation in a Dutch municipality. Dutch municipalities are required by law to estimate the value of houses and apartments within their borders. They use the WOZ value as a basis for determining real-estate property tax. The higher the WOZ value, the more tax an owner must pay. Many citizens appeal against the WOZ valuation, asserting it is too high.

Each of the 745 objections corresponds to a process instance. Together, these instances generated 9,583 events, all with timestamps; Figure 4 outlines how frequently the different paths are used in the model. The different stages, or "places" in Petri net jargon, of the model include color to highlight where, on average, most process time is spent; the purple stages of the process take the most time, the blue stages the least. It is also possible to select two activities and measure the time that passes between them. On average, 202.73 days pass from completion of activity "OZ02 Voorbereiden" (preparation) to completion of "OZ16 Uitspraak" (final judgment); this is longer than the average overall flow time of approximately 178 days. Approximately 416, or 56%, of the objections follow this route; the other cases follow the branch "OZ15 Zelf uitspraak" that takes, on average, less time.

Diagnostics, as in Figure 4, can be used to improve processes by removing bottlenecks and rerouting cases. Since the model is connected to event data, it is possible to drill down immediately and investigate groups of cases that take notably more time

Figure 4. Performance analysis based on 745 appeals against the WOZ valuation.

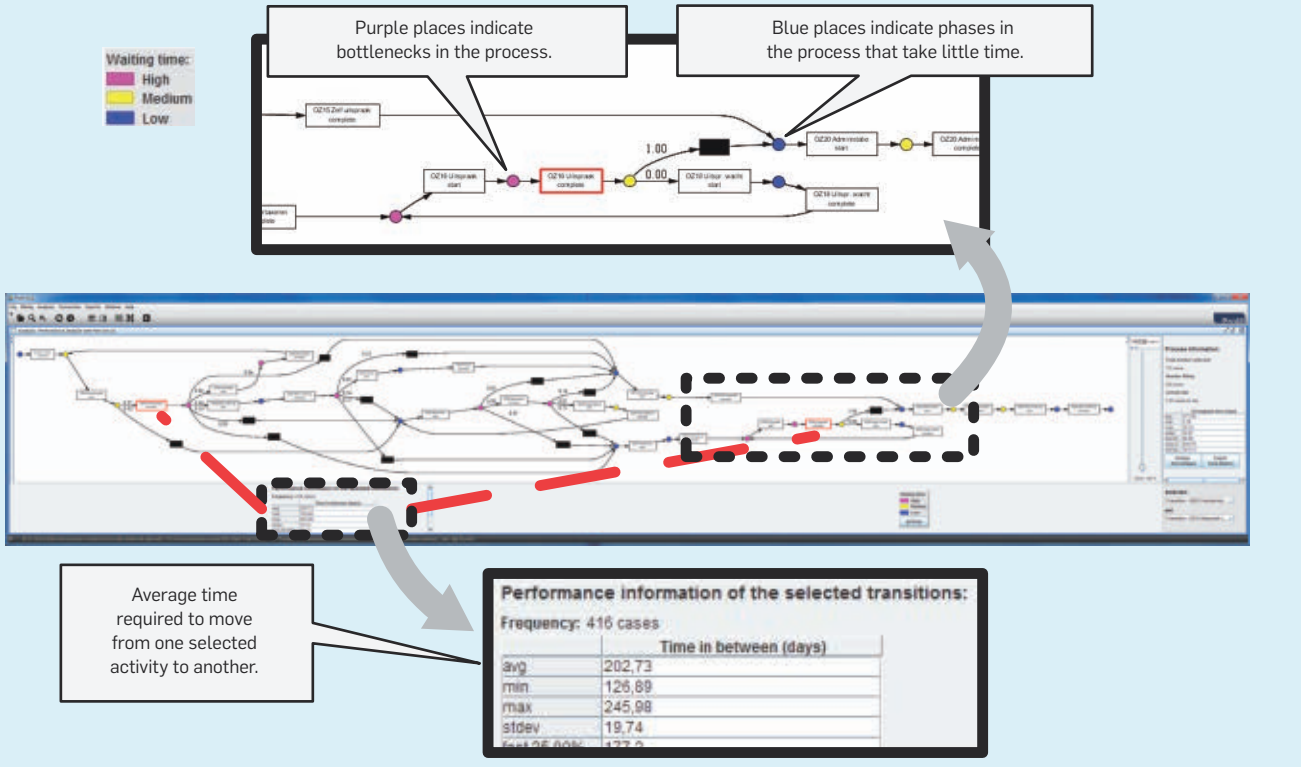


Figure 5. Conformance analysis showing deviations between event log and process model.

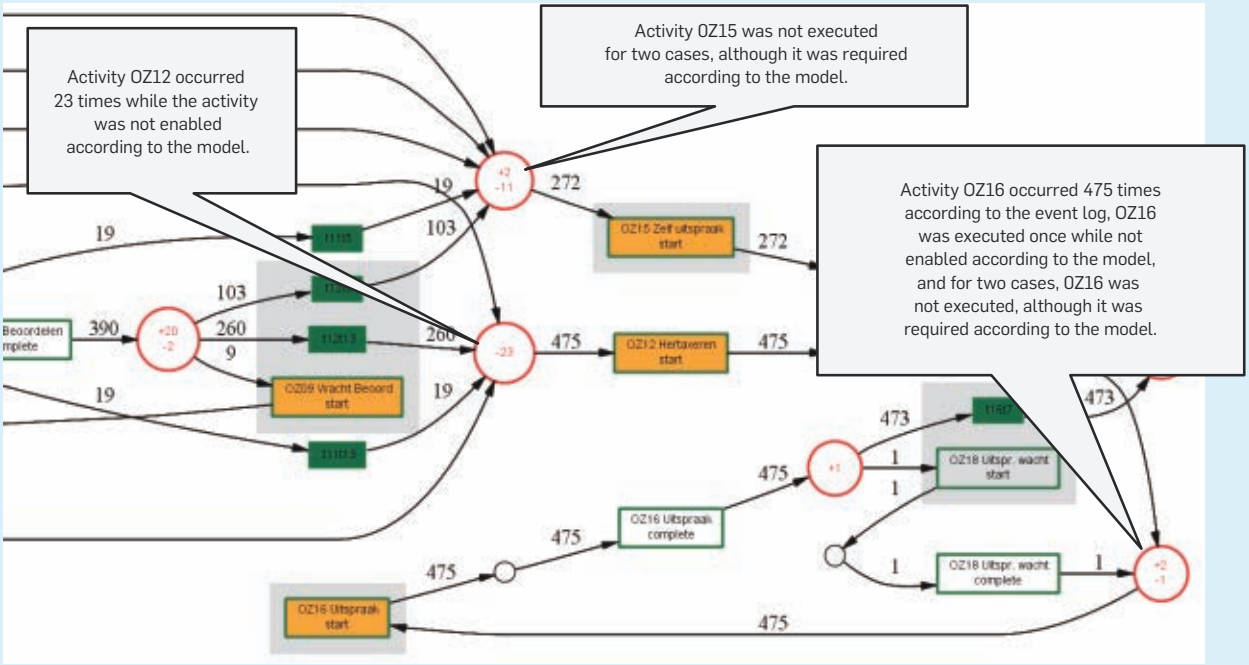
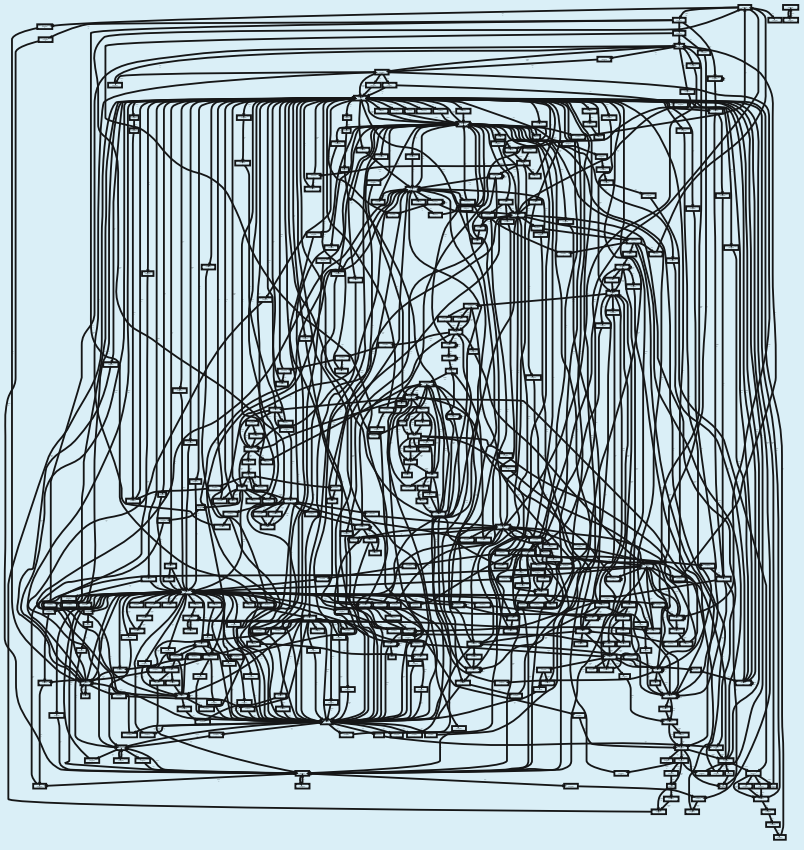


Figure 6. Process model discovered for a group of 627 gynecological oncology patients.



than others.[1]

Ensure conformance. Replay can also be used to check conformance (see Figure 5). Based on 745 appeals against the WOZ valuation, ProM was used to compare the normative model and the observed behavior, finding that 628 of the 745 cases can be replayed without encountering any problems. The fitness of the model and log is 0.98876214, indicating the model explains almost all recorded events. Despite the good fit, ProM identified all deviations; for example, "OZ12 Hertaxeren" (reevaluate property) occurred 23 times despite not being allowed according to the normative model, as indicated by the "-23" in Figure 5. With ProM the analyst can drill down to see what these cases have in common.

The conformance of the appeal process is high; approximately 99% of events are possible according to the model. Many processes have a much lower conformance; for example, it is not uncommon to find processes where only 40% of events are possible according to the model; for example, process mining revealed ASML's modeled test process strongly deviated from the real process.[8]

The increasing importance of corporate governance, risk, compliance management, and legislation (such as the Sarbanes-Oxley Act and the Basel II Accord) highlight the practical relevance of conformance checking. Process mining can help auditors check whether processes execute within certain boundaries set by managers, governments, and other stakeholders.[3] Violations discovered through process mining might indicate fraud, malpractice, risks, or inefficiency; for example, in the municipality for which the WOZ appeal process was analyzed, ProM revealed misconfigurations of its eiStream workflow-management system. Municipal employees frequently bypassed the system because system administrators could manually change the status of cases (such as to skip activities or roll back the process).[7]

Show variability. Handmade process models tend to provide an idealized view of the business process being modeled. However, such "PowerPoint reality" often has little in common with real processes, which have much more variability. To improve conformance and performance, process analysts should not naively abstract away this variability.

Process mining often involves spaghetti-like models; the one in Figure 6 was discovered based on an event log containing 24,331 events referring to 376 different activities describing the diagnosis and treatment of 627 gynecological oncology patients in the AMC Hospital in Amsterdam. The spaghetti-like structures are not caused by the discovery algorithm but by the variability of the process.

Although stakeholders should see reality in all its detail (see Figure 6), spaghetti-like models can be simplified. As with electronic maps, it is possible to seamlessly zoom in and out.[1] Zooming out, insignificant things are either left out or dynamically clustered into aggregate shapes, in the same way streets and suburbs amalgamate into cities in Google Maps. The significance level of an activity or connection may be based on frequency, costs, or time.

Improve reliability. Process mining can also help improve the reliability of systems and processes; for example, since 2007, we have used process mining to analyze the event logs of X-ray machines from Philips Healthcare[1] that record massive amounts of events describing actual use. Regulations in different countries require proof systems were tested under realistic circumstances; for this reason, process discovery was used to construct realistic test profiles. Philips Healthcare also used process mining for fault diagnosis to identify potential failures within its X-ray systems. By learning from earlier system failure, fault diagnosis was able to find the root cause for new emergent problems. For example, we used ProM to analyze the circumstances under which particular components are replaced, resulting in a set of "signatures," or historical fault patterns; when a malfunctioning X-ray machine exhibits a particular signature behavior, the service engineer knows what component to replace.

Enable prediction. Combining historic event data with real-time event data can also help predict problems before they occur; for instance, Philips Healthcare can anticipate that an X-ray tube in the field is about to fail by discovering signature fault patterns in the machine's event logs, so the tube can be replaced before the machine begins to malfunction. Many data sources today are updated in (near) real time, and sufficient computing power is available to analyze events as they occur. Process mining is not restricted to offline analysis and is useful for online operational support. Predictions can even be made for a running process instance (such as expected remaining flow time).[1]

Conclusion

Process-mining techniques enable organizations to X-ray their business processes, diagnose problems, and identify promising solutions for treatment. Process discovery often provides surprising insight that can be used to redesign processes or improve management, and conformance checking can be used to identify where processes deviate. This is relevant where organizations are required to emphasize corporate governance, risk, and compliance. Process-mining techniques are a means to more rigorously check compliance while improving performance.

This article introduced the basic concepts and showed that process mining can provide value in several ways. For more on process mining see van der Aalst,[1] the first book on the subject, and the Process Mining Manifesto[9] available in 13 languages; for sample logs, videos, slides, articles, and software see http://www.processmining.org.

Acknowledgments

I thank the members of the IEEE Task Force on Process Mining and all who contributed to the Process Mining Manifesto[9] and the ProM framework.

References

1. Aalst, W. van der. *Process Mining: Discovery, Conformance and Enhancement of Business Processes.* Springer-Verlag, Berlin, 2011.
2. Aalst, W. van der. Using process mining to bridge the gap between BI and BPM. *IEEE Computer 44*, 12 (Dec. 2011), 77–80.
3. Aalst, W. van der, Hee, K. van, Werf, J.M. van der, and Verdonk, M. Auditing 2.0: Using process mining to support tomorrow's auditor. *IEEE Computer 43*, 3 (Mar. 2010), 90–93.
4. Hilbert, M. and Lopez, P. The world's technological capacity to store, communicate, and compute information. *Science 332*, 6025 (Feb. 2011), 60–65.
5. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. *Big Data: The Next Frontier for Innovation, Competition, and Productivity.* Report by McKinsey Global Institute, June 2011; http://www.mckinsey.com/mgi
6. Mendling, J., Neumann, G., and Aalst, W. van der. Understanding the occurrence of errors in process models based on metrics. In *Proceedings of the OTM Conference on Cooperative information Systems* (Vilamoura, Algarve, Portugal, Nov. 25–30), F. Curbera, F. Leymann, and M. Weske, Eds. Lecture Notes in Computer Science Series, Vol. 4803. Springer-Verlag, Berlin, 2007, 113–130.
7. Rozinat, A. and Aalst, W. van der. Conformance checking of processes based on monitoring real behavior. *Information Systems 33*, 1 (Mar. 2008), 64–95.
8. Rozinat, A., de Jong, I., Günther, C., and Aalst, W. van der. Process mining applied to the test process of wafer scanners in ASML. *IEEE Transactions on Systems, Man and Cybernetics, Part C 39*, 4 (July 2009), 474–479.
9. Task Force on Process Mining. *Process Mining Manifesto.* In *Proceedings of Business Process Management Workshops*, F. Daniel, K. Barkaoui, and S. Dustdar, Eds. Lecture Notes in Business Information Processing Series 99. Springer-Verlag, Berlin, 2012, 169–194.

Wil van der Aalst (w.m.p.v.d.aalst@tue.nl) is a professor in the Department of Mathematics & Computer Science of the Technische Universiteit Eindhoven, the Netherlands, where he is chair of the Architecture of Information Systems group.

DOI:10.1145/2240236.2240258

Imagine money you can carry and spend without a trace.

BY SCOTT AARONSON, EDWARD FARHI, DAVID GOSSET, AVINATAN HASSIDIM, JONATHAN KELNER, AND ANDREW LUTOMIRSKI

Quantum Money

EVERYBODY LIKES MONEY. It is very easy to spend. With cash and credit cards, you can buy what you want when you want it. So why are quantum computing theorists trying to rethink money?

There are a few things we all take for granted about money. We trust credit card companies to keep our transactions private and to send the right amount of money to the right place quickly. When we use paper money, we are used to the fact that we have to carry it physically with us, and we accept the risk of occasional counterfeiting.

Today, we use two basic kinds of money. First, there is the kind we carry around—coins, bank notes, poker chips, and precious metals. These are objects that are made by a mint or dug out of the ground. It is easy to verify that money is valid. You can look for the security features on paper money, you can feel coins in your hand, and, if you really know what you are doing, you can assay precious metals. All of these kinds of physical money can be counterfeited, though—if you have the right equipment, you can print paper money, stamp out your own coins, or make alloys or platings that look a lot like solid precious metals. Some subway passes and copy cards are also examples of physical money—they contain small computer chips or magnetic strips that are actually worth a certain number of subway rides or copies. But these tend to be even easier to counterfeit.[22] In theory, any physical money can be counterfeited by just using the same production process as the one used to make the original.

The other kind of money is the kind that you entrust to someone else. Think bank accounts and credit lines. You can carry these around with you in the form of checks, credit cards, and debit cards—portable devices that let you instruct your bank to move money on your behalf. Unlike physical money, there is no point in copying your own credit card (it would not double the amount of money in your bank). With a credit card, you can carry as much value as you want without weighing down your pockets and you can send money across the globe nearly instantaneously. But credit cards have disadvantages: every time you pay someone, you need to tell your bank whom to send money to. This leaves a paper trail and does not work if your connection to your bank is down.

Neither of these kinds of money is ideal. For example, imagine that you are going to Las Vegas on a business trip and you want to play some high-stakes games at night. You might feel conspicuous carrying a fat wad of cash around the strip. If you use a credit

» key insights

- **Any digital good can be perfectly copied. This is a major headache for software companies (and for the entertainment industry), and is the reason that digital cash does not exist.**
- **The quantum mechanical "no-cloning' theorem means that in principle it is possible to design quantum systems that cannot be copied. Several recent works propose to use such systems for digital money.**
- **Further research may lead to a new form of digital rights management.**

ILLUSTRATION BY SPOOKY POOKA AT DEBUT ART

card, your significant other (not to mention anyone else who gets access to your bank statements) will know exactly how much money you gambled. What you really want is some kind of money that you can spend without leaving a trace and that you can carry as much of as you want without weighing down your pockets.

This kind of money would be digital: you could transmit it and fit as much of it as you want on some small handheld computer. It would be self-contained, so you could pay someone without any third party being involved. And it would be cryptographically secure: attackers could never produce a counterfeit bill that passes as real money even with extraordinary resources at their disposal.

The reason we do not have this kind of money today is not for lack of trying. Any digital piece of information that can be sent over a communication channel can be copied. This makes digital money seem impossible: if you had one hundred dollars on your computer, you could back up your computer, spend the money, restore your computer from the backup, and spend your money again.

Enter quantum mechanics. Physicists have known for years that if you possess a single quantum object and know nothing about it, then it is fundamentally impossible to copy that object perfectly. This is called the no-cloning theorem, and it gives us hope that *quantum* information could be used as the basis of a better kind of money.

So can we make the idealized money out of quantum mechanical objects rather than classical ones? In the rest of this article, we will survey recent work that has tackled this question. We will introduce the idea of quantum information, and we will talk about a few proposals for quantum money and some of the open problems in the field.

Quantum Mechanics

If you look closely enough, everything is made out of subatomic particles, and these particles obey the laws of quantum mechanics. Quantum mechanical systems store information in a way that is dramatically different from classical (that is, non-quantum) systems.

One of the simplest examples of a quantum system is a single electron. Electrons spin, and their spin can be characterized by a three-dimensional vector.[a] This vector, like any three-dimensional vector, has three components, S_x, S_y, and S_z. It is possible to do an experiment to measure the vertical component S_z of an electron's spin, but if you do the experiment, you will discover something strange: S_z can only take on two values, +1 and –1.

a The vector represents the angular momentum of the electron, but its physical interpretation is not important for this discussion.

Once you have measured S_z, you can measure it again and you will get the same answer as you got the first time around. S_x and S_y work the same way. So far, you might have thought that each component of the electron's spin stores one bit of information.

But if you try to measure more than one of the components, again something strange happens. Take an electron, measure S_z, and suppose that the outcome is +1. Now measure S_x (obtaining either +1 or −1) and then measure S_z again. You would expect to get $S_z = +1$ as before, but if you do this experiment you will get +1 half the time and −1 half the time. Measuring the electron's spin therefore changes the spin state of the electron. Physicists have come to realize that this is not a limitation of their experiments but rather that the universe fundamentally operates this way.

No matter what encoding you use or how perfect an apparatus you can build, you can only ever reliably encode one bit worth of recoverable classical information in the spin of an electron.[20] Nonetheless, an electron behaves very differently than a classical bit. If we use electron spins instead of classical bits to store information, we can perform tasks that are completely impossible with ordinary computers.

Figure 1. The no-cloning theorem.

Imagine that someone prepares a single qubit in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and gives it to you without telling you what α and β are. Your goal is to copy that qubit. We will call whatever algorithm you use (the supposed "quantum copy machine") C. You feed C the (unknown) qubit $|\psi\rangle$ and an output qubit that starts in the state $|0\rangle$ and your machine needs to output the original qubit and transform $|0\rangle$ into a copy of $|\psi\rangle$.

You do not know in advance what α and β are, so your copy machine has to work for any values. In particular, your machine needs to work if $\alpha = 1$ and $\beta = 0$, which means

$$C(|0\rangle|0\rangle) = |0\rangle|0\rangle.$$

Similarly, your copy machine needs to work if $\alpha = 0$ and $\beta = 1$, which means

$$C(|1\rangle|0\rangle) = |1\rangle|1\rangle.$$

But quantum mechanics is linear, so any copy machine you could possibly build has to be linear as well. This means that the operator C is linear, so we can do some linear algebra:

$$\begin{aligned} C(|\psi\rangle|0\rangle) &= C((\alpha|0\rangle + \beta|1\rangle)|0\rangle) \\ &= \alpha C(|0\rangle|0\rangle) + \beta C(|1\rangle|0\rangle) \\ &= \alpha|0\rangle|0\rangle + \beta|1\rangle|1\rangle. \end{aligned} \quad (1)$$

Your copy machine was supposed to copy *any* state, so the output should have been

$$\begin{aligned} &(\alpha|0\rangle + \beta|1\rangle)(\alpha|0\rangle + \beta|1\rangle) \\ &= \alpha^2|0\rangle|0\rangle + \alpha\beta|0\rangle|1\rangle \\ &\quad + \alpha\beta|1\rangle|0\rangle + \beta^2|1\rangle|1\rangle \end{aligned}$$

If both α and β are nonzero, then the output (1) of your machine is not correct. This means that your copy machine C cannot possibly work.

Qubits

An electron's spin is an example of a mathematical object called a qubit. A classical bit can take either of the two values 0 or 1. But a qubit is described mathematically by a normalized state in a two-dimensional complex vector space. We will use notation from physics to denote vectors that represent quantum states, writing a vector named v as $|v\rangle$. We can write any one-qubit state as

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle$$

where the states $|0\rangle$ and $|1\rangle$ form a basis for the 2D vector space and where α and β are complex numbers that satisfy $|\alpha|^2 + |\beta|^2 = 1$. If neither α nor β is zero, then we call the state $|q\rangle$ a *superposition* of $|0\rangle$ and $|1\rangle$ because the qubit $|q\rangle$ is, in a sense, in both states at once.

Just as one qubit can be in the state $|0\rangle$ or $|1\rangle$ or some superposition (linear combination) of both, n qubits can be in any superposition of the states

$$|0\ldots00\rangle, |0\ldots01\rangle, |0\ldots10\rangle, |0\ldots11\rangle, \ldots, |1\ldots11\rangle$$

So, an n qubit state is a vector in a 2^n-dimensional space.

The simplest kind of measurement one can perform on a single qubit is one that answers this question: is the qubit in a given state $|r\rangle = a|0\rangle + b|1\rangle$? Let us say our qubit is prepared in the state $|q\rangle$ as above and we make this measurement. Then there are two possible outcomes. We might get the answer YES, in which case the state of the system would change instantaneously from $|q\rangle$ to $|r\rangle$. The probability that this happens is given by the complex inner product squared of the two states in question

$$\Pr[\text{YES}] = |\alpha^* a + \beta^* b|^2.$$

If on the other hand we obtain the measurement outcome NO, then the state of the system would instantaneously change from $|q\rangle$ to the state $|r^\perp\rangle = b^*|0\rangle - a^*|1\rangle$ that is perpendicular to $|r\rangle$. This happens with probability

$$\Pr[\text{NO}] = |\alpha^* b - \beta^* a|^2 = 1 - \Pr[\text{YES}].$$

We can use this mathematical framework to explain the measurement statistics of electron spin. We define the states

$$|S_z = +1\rangle = |0\rangle$$
$$|S_z = -1\rangle = |1\rangle;$$

these two states form a basis for a one-qubit vector space. Then

$$|S_x = +1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|S_x = -1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

and

$$|S_y = +1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$$
$$|S_y = -1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle).$$

Measuring the spin component S_x is the same as measuring whether the state being tested is $|S_x = +1\rangle$; the outcome YES means $S_x = +1$ and the outcome NO means $S_x = -1$. If the spin started in the $|S_z = +1\rangle$ state then, upon measuring S_x, we will obtain +1 or −1 with equal probability and the state after the measurement would be either $|S_x = +1\rangle$ or $|S_x = -1\rangle$. If we then measure S_z again, we obtain +1 or −1 with equal probability.

Physicists are trying to build devices that can manipulate electrons or other qubits in a manner analogous to the way ordinary computers manipulate bits in their memories. Such a device, if it worked reliably and could store many qubits, would be a functioning quantum computer.

Many computer scientists and physicists believe that if we could build a quantum computer, we could use it to calculate things that would be intractable with classical computers.[2]

Qubits have another strange property: unlike classical bits, they cannot be copied. This is the content of the quantum no-cloning theorem, which says there is no such thing as a perfect quantum copy machine that can copy any quantum state you feed it. (See Figure 1 for the proof in the single qubit case.) There are also limits on how closely you can approximate a quantum copy machine.[7] The no-cloning theorem allows for cryptographic protocols that go beyond the abilities of classical computers. The best known example is quantum key distribution,[4] which allows two parties to communicate privately, using a quantum channel and an authenticated (public) classical channel.

The no-cloning theorem gives us hope that *quantum* information could be used as the basis of a better kind of money.

Quantum Money

The no-cloning theorem means we should not think of qubits the same way we think about bits. One might imagine using a handful of qubits as a form of money. A mint could produce some qubits using some process known only to it, and anyone else, given just those qubits, could not copy them by any means without further information. The no-cloning theorem does not immediately imply secure quantum money is possible; it only says that machines that can copy *all* quantum states are impossible, and a counterfeiter would be content with a machine that only copied quantum states that represented valid quantum money. A counterfeiter could also try to obtain additional information about the quantum money state by using the algorithm that a merchant would use to verify quantum money. By examining concrete schemes for quantum money, we will see how these kinds of attacks can be avoided.

We distinguish two broad categories of quantum money.

In the simpler version, a mint would produce a quantum bill consisting of some number of qubits. Anyone could store the quantum bill and move it around, maybe even trading it for something else. Whenever someone wants to verify the quantum bill is valid (for example, a merchant who is offered a quantum bill as payment), he or she sends the qubits to the mint and the mint checks that they are still in the correct state using some secret process. In this type of scheme, no one other than the mint knows how to verify the money. We call this *private-key quantum money* because the key—that is, the information needed to verify the money—is private to the mint.

The other type of quantum money is *public-key* quantum money. As before, a mint can produce a quantum state and anyone can move it or spend it. Anyone should be able to verify the money themselves without communicating with the mint. Public-key money, if it could be realized, would be the ideal money we discussed earlier.

In the first quantum cryptography paper ever written,[26] Stephen Wiesner described a way to implement private-key quantum money in a provably secure manner. (He wrote the paper in 1969, but it was not published until 1983.) In Wiesner's scheme, each quantum bill is a unique random quantum state,[b] which the mint labels with a serial number. The mint keeps track of the state that corresponds to the serial number of each quantum bill and it can use its knowledge of the state to verify the money.

In 1982, Bennett et al. made the first attempt at public-key quantum money.[5] Their scheme only allowed a piece of money to be spent once (they called their quantum states subway tokens, not bills). In hindsight, their scheme is insecure for two different reasons: first, it is based on an insecure protocol for 1-2 oblivious transfer, and second, it can be broken by anyone who can run Shor's algorithm[23,24] to factor large numbers. (In the early days of quantum cryptography, there was no reason to suspect either of these weaknesses. Shor's algorithm[23] and the general attack[14] on oblivious transfer were not known until more than a decade later.)

Surprisingly, the next paper about quantum money did not appear until 2003 when Tokunaga et al.[25] attempted to modify Wiesner's scheme to prevent the mint from

b In fact, this is the random state later used in the BB84 protocol[4] for quantum key distribution.

tracking each individual bill as it is used. They achieved this by requiring that the owner of a bill modify the bill before allowing the bank to verify it. The modification is done in a special way so that valid bills remain valid but are otherwise randomized so that the bank cannot tell them apart. This scheme has the significant disadvantage that upon discovering a single counterfeit bill, the bank is required to immediately invalidate every bill it has ever issued. In our opinion this scheme therefore has limited practical applicability.

The idea of public-key quantum money gained traction in the years that followed. Aaronson proved a "complexity-theoretic no-cloning theorem,"[1] which showed that even with access to a verifier, a counterfeiter with limited computational resources cannot copy an arbitrary state. Mosca and Stebila proposed[18] the idea of a quantum coin as distinct from a quantum bill—each quantum coin of a given denomination would be identical. Using the complexity-theoretic no-cloning theorem they argued it might be possible to implement a quantum coin protocol but they did not give a concrete implementation. Aaronson[1] proposed the first concrete scheme for public-key quantum money; however, this scheme was shown to be insecure in Lutomirski et al.[16] In the latter paper, the authors suggested the idea of collision-free quantum money. Unlike quantum coins, each collision-free quantum bill has a serial number and nobody, not even the mint, can produce two bills with the same serial number. This feature can be useful to prevent the mint from printing more money than it says it is printing. The mint posts a list of all serial numbers of every quantum bill ever produced, and we can be sure the mint produced at most one bill for each serial number on the list. In a subsequent paper, Farhi et al. proposed a concrete scheme they believed was both collision free and secure against counterfeiting.[11]

Here, we tell you how some of these proposals work.

Wiesner's Quantum Money

Wiesner's original quantum money scheme[26] works as follows. To produce a quantum bill using n qubits, the mint first chooses n one-qubit states randomly drawn from the set $\{|S_z=1\rangle, |S_z=-1\rangle, |S_x=1\rangle, |S_x=-1\rangle\}$. The mint then assigns that state a classical serial number. A piece of quantum money consists of the n qubit state and its serial number. The mint keeps a list of all serial numbers issued as well as a description of which state corresponds to which serial number. When you pay for something with a quantum bill, the merchant sends the quantum state and its serial number back to the mint for verification. The mint looks up the serial number and retrieves the description of the corresponding quantum state. Then the mint verifies the given state is the state that goes with the attached serial number. This kind of money cannot be forged by someone outside the mint. Since a would-be forger has no knowledge of the basis that each qubit was prepared in, the quantum no-cloning theorem says he or she cannot reliably copy the n qubit quantum state (Figure 2).

The main weakness in Wiesner's scheme is that the merchant must communicate with the bank to verify each transaction. So this scheme, although theoretically inspiring and provably secure, would not be much more powerful than credit cards. Wiesner's scheme is a private-key quantum money scheme because the mint must keep a private secret—the complete description of the state—to use for verification.

The resurgence of interest in quantum money is centered around the idea of public-key quantum money.

Challenges in Designing Public-Key Quantum Money

The resurgence of interest in quantum money is centered around the idea of public-key quantum money. As we have discussed, a public-key quantum money scheme would have the following properties.[16]

1. The mint can mint it. That is, there is an efficient algorithm to produce the quantum money state.

2. Anyone can verify it without communicating with the mint. That is, there is an efficient measurement anyone can perform that accepts money produced by the mint with high probability and minimal damage.

3. No one (except possibly the mint) can copy it. That is, no one other than the mint can efficiently pro-

duce states that are accepted by the verifier with better than exponentially small probability.

Why is public-key quantum money so hard to design? The difficulty of developing public-key quantum money arises from the fact that the verification algorithm—which is known to everyone in a public-key scheme—can be used by a would-be forger in an attempt to counterfeit the money.

Wiesner's scheme is provably secure on information-theoretic grounds if it is used properly. In Wiesner's scheme, only the bank has the additional information required to verify a given quantum bill is legitimate and therefore only the bank can copy the money.

It turns out that, if the mint is careless, then even the mere act of verifying bills can allow someone to create counterfeit bills.[15] Recall that in Wiesner's scheme, in every transaction the bill is sent by the merchant back to the mint for verification. If the money is confirmed to be valid, the mint sends back the valid bill to the merchant. What happens if the money is determined by the mint to be counterfeit? If the mint sends back the invalid bill, then a counterfeiter can successfully forge the money.

Let us see how this works. A counterfeiter can start with one good quantum bill, which in Wiesner's scheme is n one-qubit states

$$|\psi\rangle = |\psi_1\rangle|\psi_2\rangle \dots |\psi_n\rangle$$

along with a serial number the bank uses to verify the state. The counterfeiter can produce a random one-qubit state $|\phi_1\rangle$, and, setting aside the first qubit $|\psi_1\rangle$ of the original bill, he or she then sends the mint the state

$$|\psi'\rangle = |\phi_1\rangle|\psi_2\rangle \dots |\psi_n\rangle.$$

If the bill $|\psi'\rangle$ turns out to be valid (this happens with probability ½), the mint returns the bill, and in this case the mint's measurement will have changed the state to $|\psi\rangle$. So now the counterfeiter possesses both $|\psi\rangle$ and the original qubit $|\psi_1\rangle$ that was set aside, and so he or she has succeeded in copying the first qubit $|\psi_1\rangle$. On the other hand, if the mint determines the bill $|\psi'\rangle$ is not valid, then the state of the bill after the mint's measurement will be

$$|\psi_1^{\perp}\rangle|\psi_2\rangle \dots |\psi_n\rangle$$

where $|\psi_1^{\perp}\rangle$ is the one-qubit state orthogonal to $|\psi_1\rangle$. Note that the states of qubits 2 through n have not been changed by this process. So the counterfeiter can then throw away $|\psi_1^{\perp}\rangle$, replace it with a random state, and try again. After an average of two tries, the counterfeiter will have copied the first qubit of the quantum bill. Then the counterfeiter can repeat this whole procedure to copy the second qubit, the third qubit, and so on until all n qubits have been copied.[c]

So if the bank sends back quantum states it deems to be invalid quantum money, the whole scheme is unusable. This tells us how *not* to implement Wiesner's scheme in practice. But it also highlights the fact that having access to a verifier that returns the state and a verdict on the validity of the quantum money is in itself a powerful tool a forger can try to exploit, even if the forger cannot look inside the machine that verifies money. This type of attack is particularly applicable to public-key quantum money schemes, in which the verification algorithm is publicly known.

This attack was particularly simple against Wiesner's money because each bill consists of independent (that is, unentangled) qubits. A more general algorithm called quantum state restoration[10] works on entangled states as well: starting with a single valid quantum bill, a counterfeiter can make a sequence of measurements on the state and use the algorithm that verifies the bill to undo the damage caused by measuring the state. So any public-key quantum money scheme must be designed so that the attacker cannot gain enough information to copy the quantum money state by making a reasonable number of measurements on one copy of it. Can we hope to design a public-key quantum money scheme which has this property, or is the access to a verification algorithm already enough information to allow cloning of an arbitrary state? Aaronson answered this question in 2009 with a "complexity-theoretic no-cloning theorem."

The Complexity-Theoretic No-Cloning Theorem

As we mentioned earlier, the standard no-cloning theorem is not good enough to prove secure public-key quantum money is possible, since it does not take into account the counterfeiter can *check* whether a given state is valid quantum money or not. In fact, if a counterfeiter has unlimited time, then it is straightforward to counterfeit public-key quantum money: simply generate a random state and check if that state is valid money. If not, try again. In a secure money scheme, the probability that any attempt succeeds is exponentially small.

c The attack in Lutomirski[15] is different: it is deterministic and twice as fast, but it is less intuitive.

Figure 2. Wiesner's quantum money. Source: *Science*, Aug. 7, 1992.

The complexity-theoretic no-cloning theorem[1] says there is no *generic* attack much better than random guessing. What do we mean by a generic attack? Suppose there is a verification machine that checks whether or not a given state $|\phi\rangle$ is equal to a good quantum money state $|\psi\rangle$. The machine takes as input any quantum state $|\phi\rangle$; it outputs 0 if $|\phi\rangle = |\psi\rangle$ and 1 if $|\phi\rangle$ is orthogonal to $|\psi\rangle$. In either case, it also outputs the quantum state is left over after the measurement. Aaronson showed that, as long as that machine is a black box, it can fall into the hands of a counterfeiter without compromising the quantum money scheme. In other words, a counterfeiter with access to some quantum money as well as the verification machine would either need to take the machine apart to figure out how it worked or else use the machine an exponentially large number of times in order to make any more quantum money than he or she started with.

This theorem does not guarantee any particular scheme is secure. For every quantum money scheme that has been proposed, the states $|\psi\rangle$ that are "good" quantum money states are not completely unknown since they come from a restricted set of states generated by the mint's algorithm. If this set of states is small enough then having a "black box" verifier may allow a forger to copy a money state; we have already seen an example of this with Wiesner's scheme. And it might also be possible to design attacks on public-key quantum money that do not use the verifier as a black box. So in order to evaluate any public-key quantum money scheme, we will have to look at the details of the verifier and the set of valid quantum money states that are minted by the bank.

Figure 3. Quantum money from knots.

Figure 4. A knot.

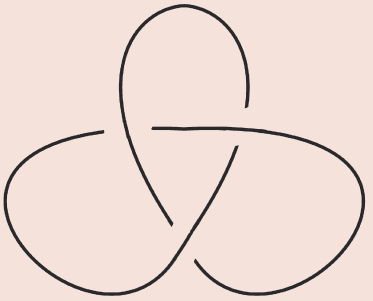
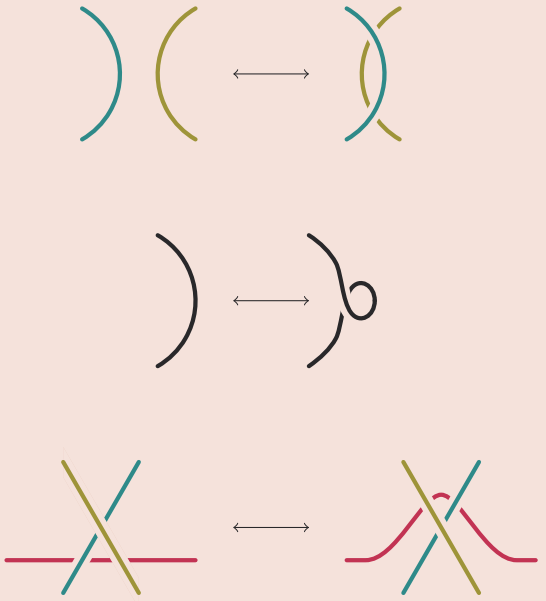


Figure 5. Reidemeister moves.



Quantum Coins

One of the first applications of the complexity-theoretic no-cloning theorem was given by Mosca and Stebila.[18] They showed it might be possible to have public-key quantum money scheme in which every piece of quantum money is identical: they called these *quantum coins*.[18,19]

Quantum coins, like ordinary coins, are all the same with no marks distinguishing each coin. One advantage of quantum coins is they are *anonymous*—no one can tell one coin from another, so it is difficult to keep track of where and when a particular coin was spent.

Mosca and Stebila had two results about quantum coins. They extended the complexity-theoretic no-cloning theorem to quantum coins. If a would-be counterfeiter has access to a machine that verifies quantum coins but cannot look inside that machine, then there is no way to make more coins than he or she started with in any reasonable amount of time. This result gives some hope a public-key quantum coin protocol could be discovered.

Their second result is based on blind quantum computation (introduced by Childs[9] and studied by Broadbent et al.[6]). Blind quantum computation is a protocol whereby a quantum computer with very limited resources (sometimes called a quantum calculator) runs a polynomial size quantum circuit with the help of a server, where the server does not learn anything about the circuit performed (except an upper bound on its size). In the protocol introduced by Mosca and Stebila, the merchant runs an obfuscated verification algorithm from which he or she learns nothing except the final answer: that it is or is not a valid coin. However, this requires (quantum) communication with the bank, and so this quantum coin scheme is a private-key protocol.

To date there is no published concrete proposal for public-key quantum coins.

Public-Key Quantum Money without Secrets

In all of the schemes we have discussed so far, the mint first generates some classical secret and then uses that secret to produce the quantum money. In any scheme like this, if anyone can figure out the secret, then they can use this secret to produce valid quantum money states with the same algorithm that the mint uses. A would-be forger can try to use the (publicly known) verification algorithm along with techniques such as quantum state restoration[10] to try to reverse-engineer the secret.

Lutomirski et al.[16] suggested a different approach to designing quantum money. Imagine a physical process (or a quantum measurement) that can simultaneously generate a random serial number y (drawn from an enormous set of possible serial numbers) and a corresponding quantum state $|\$_y\rangle$. For any given serial number y, a second algorithm would be able to verify some quantum state was indeed $|\$_y\rangle$. A key feature of this scheme is *collision-freedom*: no one can generate more than one copy of $|\$_y\rangle$ for any value of y. (Anyone can generate quantum states corresponding to *different* serial numbers.)

To use these states as money, the mint simply generates a pile of quantum states and corresponding classical serial numbers. The mint then publishes the list of all the serial numbers and the verification algorithm that can be used by anyone to check the validity of a given quantum money state. A quantum state matching a published serial number is valid money; any other state is not. If the mint published an actual list, then anyone could also verify the mint produced no more quantum money than it said it did; as a practical matter, though, the mint would probably use digital signatures instead of a list.

Lutomirski et al. also suggested a way such an algorithm might be designed. Consider a large set S and a function f that assigns each element of S a label. Suppose there is an exponential number of possible labels and an exponential number of elements of S that share each label. Each label corresponds to a serial number, and the state corresponding to the serial number y is a uniform superposition of all of the elements of S that have the label y. Mathematically,

$$|\$_y\rangle = \sum_{x \in S \text{ s.t. } f(x)=y} |x\rangle.$$

One advantage of quantum coins is that they are *anonymous*—no one can tell one coin from another, so it is difficult to keep track of where and when a particular coin was spent.

To produce a quantum money state, the mint first prepares a uniform superposition over all elements of S and measures the label that corresponds to the state. This results in a random label and, like all measurements, changes the state so the new state will always get the same measurement outcome. This means the superposition collapses to exactly those elements of S that have the measured label.

The verification procedure presented by Lutomirski et al. requires anyone who knows some x where $f(x)=y$ find another *random* x' with the same label y, and therefore f must be chosen so this is possible. A merchant who wants to verify a quantum bill first measures the label and confirms it matches the serial number of the bill, and then performs a more complicated quantum measurement to check the state is invariant under the operation that randomizes the elements that share the same label.

Lutomirski et al. conjecture that if f and S are appropriately chosen, then the resulting quantum money will be secure. In that paper, however, they did not describe an appropriate f and S.

Quantum Money from Knots

The only published scheme[11] for public-key quantum money that has not been shown to be insecure is an implementation of collision-free quantum money. In this scheme, the set S is a set of drawings of knots. We will have to take a quick detour into knot theory in order to describe this quantum money (Figure 3).

Most of us have some experience in our day-to-day lives with the basic properties of knots. Mathematicians who study knot theory have formalized these basic properties. For our purposes, a good place to start will be with some definitions. A knot is a mapping of the circle S^1 (like a loop of string) into three-dimensional space. For example, Figure 4 shows a knot.

Usually when we draw a knot, we use a two-dimensional diagram like the one in Figure 4. If we take a knot and then fiddle with it a bit (without

cutting the string it is made out of) and then draw it, we might end up with a different diagram. But we would still like to call it the same knot. So the question arises: which pictures represent the same knot? The three modifications to a knot diagram shown in Figure 5 are called the Reidemeister moves. It can easily be seen that by applying these moves you only move between topologically equivalent knots. It is also true (but more difficult to see) that any two diagrams representing the same knot can be mapped into one another using these moves.

There is no known good algorithm to determine whether two knot diagrams represent the same knot; it has only recently been discovered that knot equivalence is decidable.[13] But sometimes there is a way to tell that two diagrams do not represent the same knot; by using a *knot invariant*. A knot invariant is a property of a knot that is the same for all diagrams representing the same knot. If you can find a knot invariant that takes different values for the two diagrams in question, then you can be sure they represent different knots. (The converse of this is not generally true—there can be two different knots that share the same value for a particular knot invariant.) One of the first knot invariants to be discovered is called the Alexander polynomial—any knot has an associated Alexander polynomial, and its coefficients are integers that can be efficiently calculated from any diagram of that knot.

To make quantum money from knots, the set S in the general collision-free scheme is taken to be the set of knot diagrams, and label f associated with each diagram is its Alexander polynomial. Applying a sequence of random Reidemeister moves randomizes among knots with the same diagram, allowing the measurement that verifies the quantum money states. So the mint prepares the superposition over all diagrams and measures the Alexander polynomial's coefficients to make a quantum bill, and a merchant measures the coefficients and verifies the superposition is invariant under the Reidemeister moves.

(The actual scheme is somewhat more complicated because knot diagrams are inconvenient to work with—see Farhi et al.[11] for the technical details.)

While no one has proven that knot money is secure, attempts to break it seem to run into knot theory problems that have no known practical solutions.

What Does the Future Hold for Quantum Money?

Public-key quantum money is one of few quantum protocols that does something that is truly impossible classically, even under cryptographic assumptions. QKD can be used to encrypt information between two parties that did not coordinate keys in advance, but under reasonable security assumptions, lattice based cryptography can perform the same feat.[4,21] Assuming SHA1 is a pseudo random function, one can use it to implement strong coin flipping,[8,17] and encrypted communication channels enable fast Byzantine agreement.[3,12] However, no cryptographic assumption enables a digital analog of cash, as any string of bits that would represent a bill can always be copied.

The idea of some kind of quantum object that only one special entity can produce may have applications beyond being used as money. For example, software companies would like to be able to produce software programs that anyone can use but that no one can copy. Whether this is possible for any useful type of software remains to be seen.

Will a future government replace its currency with quantum money? Maybe. You could use it online to purchase things without transaction fees and without oversight from any third party. You could download your quantum money onto your qPhone (not yet trademarked) and use it to buy things from quantum vending machines. With the advent of quantum money, we hope everybody will like spending money a little bit more. C

References

1. Aaronson, S. Quantum copy-protection and quantum money. In *Annual IEEE Conference on Computational Complexity* (2009), 229–242.
2. Bacon, D. and van Dam, W. Recent progress in quantum algorithms. *Commun. ACM 53* (Feb. 2010), 84–93.
3. Ben-Or, M. and Hassidim, A. Fast quantum byzantine agreement. In *STOC* (2005), 481–485.
4. Bennett, C.H. and Brassard, G. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers Systems and Signal Processing* (Bangalore, India, 1984), 175–179.
5. Bennett, C.H., Brassard, G., Breidbart, S. and Wiesner, S. Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology—Proceedings of Crypto* (1983), volume 82, 267–275.
6. Broadbent, A., Fitzsimons, J. and Kashefi, E. Universal blind quantum computation. In *FOCS* (2009), 517–526.
7. Bužek, V. and Hillery, M. Quantum copying: Beyond the no-cloning theorem. *Phys. Rev. A 54*, 3 (1996), 1844–1852.
8. Chailloux, A. and Kerenidis, I. Optimal quantum strong coin flipping. In *50th Annual IEEE Symposium on Foundations of Computer Science, 2009 (FOCS'09)* (2009), IEEE, 527–533.
9. Childs, A.M. Secure assisted quantum computation. *Quant. Inform. Comput. 5*, 6 (2005), 456–466.
10. Farhi, E., Gosset, D., Hassidim, A., Lutomirski, A., Nagaj, D., and Shor, P. Quantum state restoration and single-copy tomography for ground states of hamiltonians. *Phys. Rev. Lett. 105*, 19 (Nov. 2010), 190503.
11. Farhi, E., Gosset, D., Hassidim, A., Lutomirski, A. and Shor, P. Quantum money from knots. In *Innovations in Theoretical Computer Science* (2012).
12. Feldman, P. and Micali, S. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Comput. 26*, 4 (1997), 873–933.
13. Hass, J. Algorithms for recognizing knots and 3-manifolds. *Chaos, Solitons & Fractals 9*(4–5) (1998), 569–581.
14. Lo, H.-K. Insecurity of quantum secure computations. *Phys. Rev. A 56* (Aug. 1997), 1154–1162.
15. Lutomirski, A. An online attack against Wiesner's quantum money. *arXiv:1010.0256*, 2010.
16. Lutomirski, A., Aaronson, S., Farhi, E., Gosset, D., Hassidim, A., Kelner, J. and Shor, P. Breaking and making quantum money: Toward a new quantum cryptographic protocol. In *Innovations in Computer Science* (2010).
17. Mochon, C. Quantum weak coin flipping with arbitrarily small bias. *Arxiv preprint arXiv:0711.4114* (2007).
18. Mosca, M. and Stebila, D. A framework for quantum money. Poster at *Quantum Information Processing (QIP)* (Brisbane, Australia, 2007).
19. Mosca, M. and Stebila, D. Quantum coins. In *Error-Correcting Codes, Finite Geometries, and Cryptography*. Contemporary Mathematics, Aiden A. Bruen and David L. Wehlau, eds. volume 523. American Mathematical Society, 2010, 35–47.
20. Nielsen, M.A. and Chuang, I.L. *Quantum Information and Computation*, Cambridge University Press, Cambridge, UK, 2000.
21. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th annual ACM symposium on Theory of computing* (2005), ACM, 84–93.
22. Ryan, R., Anderson, Z. and Chiesa, A. Anatomy of a subway hack. http://tech.mit.edu/V128/N30/subway/Defcon_Presentation.pdf (August 2008).
23. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS* (1994), IEEE Computer Society, 124–134.
24. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev. 41*, 2 (1999), 303–332.
25. Tokunaga, Y., Okamoto, T. and Imoto, N. Anonymous quantum cash. In *EQIS* (Aug. 2003).
26. Wiesner, S. Conjugate coding. *SIGACT News 15*, 1 (1983), 78–88.

Scott Aaronson (aaronson@csail.mit.edu) is an associate professor, CSAIL, MIT, Cambridge, MA.

Edward Farhi (farhi@mit.edu) is the Cecil and Ida Green Professor of Physics, and director of the Center for Theoretical Physics, MIT, Cambridge, MA.

David Gosset (dgosset@mit.edu) is a postdoctorate fellow in the Department of Combinatorics and Optimization and the Institute for Quantum Computing, University of Waterloo, Canada.

Avinatan Hassidim (avinatanh@gmail.com) is an assistant professor in the Department of Computer Science, Bar Ilan University, and works at Google Israel.

His research is supported by a grant from the German-Israeli Foundation (GIF) for Scientific Research and Development under contract number 1-2322-407.7/2011.

Jonathan Kelner (kelner@mit.edu) is an assistant professor of applied mathematics, and a member of CSAIL, MIT, Cambridge, MA.

Andrew Lutomirski (andy@luto.us) is co-founder of AMA Capital Management LLC, Palo Alto, CA.

Computing Reviews is on the move

Our new URL is

ComputingReviews.com

CR

COMPUTING REVIEWS

A daily snapshot of what is new and hot in computing

Previous A.M. Turing Award Recipients

1966 A.J. Perlis
1967 Maurice Wilkes
1968 R.W. Hamming
1969 Marvin Minsky
1970 J.H. Wilkinson
1971 John McCarthy
1972 E.W. Dijkstra
1973 Charles Bachman
1974 Donald Knuth
1975 Allen Newell
1975 Herbert Simon
1976 Michael Rabin
1976 Dana Scott
1977 John Backus
1978 Robert Floyd
1979 Kenneth Iverson
1980 C.A.R Hoare
1981 Edgar Codd
1982 Stephen Cook
1983 Ken Thompson
1983 Dennis Ritchie
1984 Niklaus Wirth
1985 Richard Karp
1986 John Hopcroft
1986 Robert Tarjan
1987 John Cocke
1988 Ivan Sutherland
1989 William Kahan
1990 Fernando Corbató
1991 Robin Milner
1992 Butler Lampson
1993 Juris Hartmanis
1993 Richard Stearns
1994 Edward Feigenbaum
1994 Raj Reddy
1995 Manuel Blum
1996 Amir Pnueli
1997 Douglas Engelbart
1998 James Gray
1999 Frederick Brooks
2000 Andrew Yao
2001 Ole-Johan Dahl
2001 Kristen Nygaard
2002 Leonard Adleman
2002 Ronald Rivest
2002 Adi Shamir
2003 Alan Kay
2004 Vinton Cerf
2004 Robert Kahn
2005 Peter Naur
2006 Frances E. Allen
2007 Edmund Clarke
2007 E. Allen Emerson
2007 Joseph Sifakis
2008 Barbara Liskov
2009 Charles P. Thacker
2010 Leslie G. Valiant
2011 Judea Pearl

ACM A.M. TURING AWARD NOMINATIONS SOLICITED

Nominations are invited for the 2012 ACM A.M. Turing Award. ACM's oldest and most prestigious award is presented for contributions of a technical nature to the computing community. Although the long-term influences of the nominee's work are taken into consideration, there should be a particular outstanding and trendsetting technical achievement that constitutes the principal claim to the award. The award carries a prize of $250,000 and the recipient is expected to present an address that will be published in an ACM journal. Financial support of the Turing Award is provided by the Intel Corporation and Google Inc.

Nominations should include:

1) A curriculum vitae, listing publications, patents, honors, other awards, etc.

2) A letter from the principal nominator, which describes the work of the nominee, and draws particular attention to the contribution which is seen as meriting the award.

3) Supporting letters from at least three endorsers. The letters should not all be from colleagues or co-workers who are closely associated with the nominee, and preferably should come from individuals at more than one organization. Successful Turing Award nominations usually include substantive letters of support from a group of prominent individuals broadly representative of the candidate's field.

For additional information on ACM's award program please visit: www.acm.org/awards/

Additional information on the past recipients of the A.M. Turing Award is available on: http://amturing.acm.org/byyear.cfm .

Nominations should be sent electronically by November 30, 2012 to:
Ravi Sethi, Avaya Labs c/o mcguinness@acm.org

Association for Computing Machinery

research highlights

DOI:10.1145/2240236.2240259

Technical Perspective
Example-Driven Program Synthesis for End-User Programming

By Martin C. Rinard

UNDERSTANDING HOW TO get a computer to perform a given task is a central question in computer science. For many years the standard answer has been to use a programming language to write a program the computer will then execute to accomplish the task.

An intriguing alternative, however, is to provide the computer with examples of inputs and corresponding outputs, then have the computer automatically generalize the examples to produce a program that performs the desired task for all inputs. Researchers have worked on this approach for decades, first in the LISP community,[4] then later in the inductive logic programming community,[1–3] to name two prominent examples. Given the relatively modest size of the programs the resulting techniques are able to produce, the field has evolved to focus largely on data mining, concept learning, knowledge discovery, and other applications (as opposed to mainstream software development).

The following paper focuses on an important emerging area—end user programming. As information technology has come to permeate our society, broader classes of users have developed the need for more sophisticated data manipulation and processing. While users in the past were satisfied with relatively simple interactive models of computation such as spreadsheets and other business applications, current users are now looking to automate custom data manipulations such as reformatting, reorganizing, simple calculations, or data cleaning. While such users may have a good command of the interactive functionality of their application, they often lack the expertise, time, or inclination to develop software specifically for their task.

The authors illustrate how to apply example-driven program synthesis to automate spreadsheet computations. This work, therefore, is of interest to the millions of people worldwide who use spreadsheets. The methodology consists of four basic steps:

- **Domain-specific language**: Develop a domain-specific language capable of representing the desired set of computations.
- **Data structure**: Develop a data structure that can efficiently represent the large set of programs that are consistent with a given input/output example.
- **Learn and intersect**: Generate data structures for representing the programs consistent with each individual input/output example, then intersect the data structures to obtain a representation of the programs consistent with all examples.
- **Rank** the resulting set of programs, preferring more general programs over less general programs. Users can then view the results of the ranked programs on different inputs to guide the program selection process.

This approach effectively addresses many of the issues that complicate example-driven approaches. Domain-specific languages help focus the synthesis process by avoiding the generality of standard programming languages (that can produce very large program search spaces that are intractable to manipulate efficiently). Compact program representations make it possible to manipulate large numbers of programs efficiently. An effective ranking algorithm helps users quickly identify a desirable program (out of the potentially unbounded number of programs that are consistent with the provided examples). And the interactive program evaluation mechanisms (automatic identification of inputs on which candidate programs produce different outputs) help users navigate the space of synthesized programs.

The authors have successfully applied this approach to three classes of spreadsheet programs: syntactic string manipulations (such as converting telephone numbers into a standard format), semantic string manipulations that make use of some background system knowledge stored as relational tables (such as transforming strings in inventory tables or transforming dates), and table layout computations (which reorganize data stored in tables). All of these systems have been successfully integrated with Microsoft Excel and have been tested on multiple examples from Excel help forums.

In the future, the need for users to obtain ever more sophisticated custom behavior will only increase. Given the significant obstacles that traditional programming approaches pose for the vast majority of users, we can expect to see a proliferation of solutions that help non-programmers accomplish software development tasks that have traditionally been the exclusive domain of software professionals. Given the difficulty of specifying and implementing large software systems, these solutions will (at least initially) focus on the automatic generation of relatively small but still useful solutions to everyday problems. This paper provides an outstanding example of the kinds of useful solutions that non-programmers can now automatically obtain and demonstrates the kind of sophisticated implementation techniques that will make such automatic program synthesis systems feasible for a variety of problem domains. C

References

1. Lavrac, N. and Dzeroski, S. *Inductive Logic Programming—Techniques and Applications*. Ellis Horwood, NY, 1994.
2. Muggleton, S. and De Raedt, L. Inductive logic programming: Theory and methods. *Journal of Logic Programming 19*, 20 (1994), 629–679.
3. Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P.A., Inoue, K. and Srinivasan, A. ILP turns 20—Biography and future challenges. *Machine Learning 86*, 1 (2012), 3–23.
4. Smith, D.R. The synthesis of LISP programs from examples: A survey. *Automatic Program Construction Techniques*, A.W. Biermann, G. Guiho, and Y. Kodratoff, Eds. Macmillan, 1984, 307–324.

Martin C. Rinard (rinard@csail.mit.edu) is a professor in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, Cambridge, MA.

DOI:10.1145/2240236.2240260

Spreadsheet Data Manipulation Using Examples

By Sumit Gulwani, William R. Harris, and Rishabh Singh

Abstract

Millions of computer end users need to perform tasks over large spreadsheet data, yet lack the programming knowledge to do such tasks automatically. We present a *programming by example* methodology that allows end users to automate such repetitive tasks. Our methodology involves designing a domain-specific language and developing a synthesis algorithm that can learn programs in that language from user-provided examples. We present instantiations of this methodology for particular domains of tasks: (a) syntactic transformations of strings using restricted forms of regular expressions, conditionals, and loops, (b) semantic transformations of strings involving lookup in relational tables, and (c) layout transformations on spreadsheet tables. We have implemented this technology as an add-in for the Microsoft Excel Spreadsheet system and have evaluated it successfully over several benchmarks picked from various Excel help forums.

1. INTRODUCTION

The IT revolution over the past few decades has resulted in two significant advances: the digitization of massive amounts of data and widespread access to computational devices. It is thus not surprising that more than 500 million people worldwide use spreadsheets for storing and manipulating data. These business *end users* have myriad diverse backgrounds and include commodity traders, graphic designers, chemists, human resource managers, finance professionals, marketing managers, underwriters, compliance officers, and even mailroom clerks—they are not professional programmers, but they need to create small, *often one-off*, applications to perform business tasks.[4]

Unfortunately, the state of the art of interfacing with spreadsheets is far from satisfactory. Spreadsheet systems, like Microsoft Excel, come with a maze of features, but end users struggle to find the correct features to accomplish their tasks.[12] More significantly, programming is still required to perform tedious and repetitive tasks such as transforming names or phone numbers or dates from one format to another, cleaning data, or extracting data from several text files or Web pages into a single document. Excel allows users to write macros using a rich inbuilt library of string and numerical functions, or to write arbitrary scripts in Visual Basic or .NET programming languages. However, since end users are not proficient in programming, they find it too difficult to write desired macros or scripts. Moreover, even skilled programmers might hesitate to write a script for a *one-off* repetitive task.

We performed an extensive case study of spreadsheet help forums and observed that string and table processing is a very common class of programming problems that end users struggle with. This is not surprising given that various languages such as Perl, Awk, and Python were designed to support string processing, and that new languages such as Java and C# provide rich support for string processing. During our study, we also observed how novice users specified their desired programs to expert users: most specifications consisted solely of one or more input–output examples. Since input–output examples may underspecify a program, the interaction between a novice and an expert often involved multiple rounds of communication over multiple days. Inspired by this observation, we developed a *programming by example* (PBE), or *inductive synthesis*, methodology[15] that has produced synthesizers that can automatically generate a wide range of string/table manipulating programs in spreadsheets from input–output examples. Each synthesizer takes the role of the forum expert, removing a human from the interaction loop and enabling users to solve their problems in a few seconds instead of few days.

This paper is organized as follows. We start with a brief overview of our PBE methodology (Section 2). We then describe an application of this methodology to perform syntactic string manipulation tasks (Section 3).[6] This is followed by an extension that automates more sophisticated semantic string manipulations requiring background knowledge, which can often be encoded as relational tables (Section 4).[18] We also describe an application of this methodology to perform layout transformations on tables (Section 5).[8] In Section 6, we discuss related work, and in Section 7, we conclude and discuss future work.

2. OVERVIEW

In this section, we outline a general methodology that we have used for developing inductive synthesizers for end-user programming tasks, along with how a user can interact with the synthesizers. In the first step of our methodology, we identify a *domain* of useful tasks that end users struggle with

This paper is based on "Automating String Processing in Spreadsheets using Input-Output," S. Gulwani, published in *POPL* (2011); "Learning Semantic String Transformations from Examples," R. Singh and S. Gulwani, *PVLDB 5* (2012), in press; and "Spreadsheet Table Transformations from Examples," W.R. Harris and S. Gulwani, published in *PLDI* (2011).

Harris's work was done during an internship at Microsoft Research.
Singh's work was done during two internships at Microsoft Research.

and can clearly describe with examples, by studying help forums or performing user studies (this paper presents two domains: string manipulation and table manipulation). We then develop the following.

Domain-specific language: We design a domain-specific language L that is expressive enough to capture several real-world tasks in the domain, but also restricted enough to enable efficient learning from examples.

Data structure for representing consistent programs: The number of programs in L that are consistent with a given set of input–output examples can be huge. We define a data structure D based on a version-space algebra[14] to succinctly represent a large set of such programs.

Algorithm for synthesizing consistent programs: Our synthesis algorithm for language L applies two key procedures: (i) `Generate` learns the set of all programs, represented using data structure D, that are consistent with a given single example. (ii) `Intersect` intersects these sets (each corresponding to a different example).

Ranking: We develop a scheme that ranks programs, preferring programs that are more general. Each ranking scheme is inspired by Occam's razor, which states that a smaller and simpler explanation is usually the correct one. We define a partial order relationship between programs to compare them. Any partial order can be used that efficiently orders programs represented in the version-space algebra used by the data structure D. Such an order can be applied to efficiently select the top-ranked programs from among a set represented using D. The ranking scheme can also take into account any *test inputs* provided by the user (i.e., new additional inputs on which the user may execute a synthesized program). A program that is undefined on any test input or generates an output whose characteristics are different from that of training outputs can be ranked lower.

2.1. Interaction models

A user provides to the synthesizer a small number of examples, and then can interact with the synthesizer according to multiple models. In one model, the user runs the top-ranked synthesized program on other inputs in the spreadsheet and checks the outputs produced by the program. If any output is incorrect, the user can fix it and reapply the synthesizer, using the fix as an additional example. However, requiring the user to check the results of the synthesized program, especially on a large spreadsheet, can be cumbersome. To enable easier interaction, the synthesizer can run *all* synthesized programs on each new input to generate a set of corresponding outputs for that input. The synthesizer can highlight for the user the inputs that cause multiple distinct outputs. Our prototypes, implemented as Excel add-ins, support this interaction model.

A second model accommodates a user who requires a reusable program. In this model, the synthesizer presents the set of consistent programs to the user. The synthesizer can show the top k programs or walk the user through the data structure that succinctly represents all consistent programs and let the user select a program. The programs can be shown using programming-language syntax, or they can be described in a natural language. The differences between different programs can be explained by synthesizing a *distinguishing input* on which the programs behave differently.[10] The user can reapply the synthesizer with the distinguishing input and its desired output as an additional example.

3. SYNTACTIC TRANSFORMATIONS

Spreadsheet users often struggle with reformatting or cleaning data in spreadsheet columns. For example, consider the following task.

Example 1 (Phone Numbers). *An Excel user wants to uniformly format the phone numbers in the input column, adding a default area code of "425" if the area code is missing.*

Input v_1	Output
323-708-7700	323-708-7700
(425)-706-7709	425-706-7709
510.220.5586	510-220-5586
235 7654	425-235-7654
745-8139	425-745-8139

Such tasks can be automated by applying a program that performs syntactic string transformations. We now present an expressive domain-specific language of string-processing programs that supports limited conditionals and loops, syntactic string operations such as substring and concatenate, and matching based on regular expressions.[6]

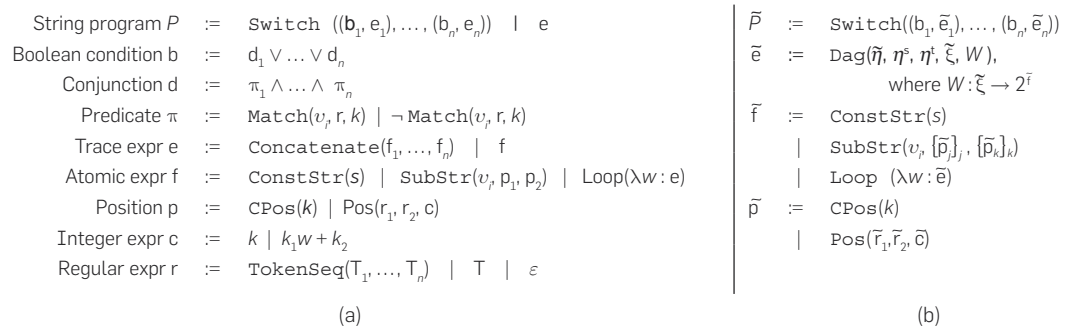
3.1. Domain-specific language

Our domain-specific programming language for performing syntactic string transformations is given in Figure 1(a). A string program P is an expression that maps an input state σ, which holds values for m string variables $v_1, \ldots, v_m$ (denoting the multiple input columns in a spreadsheet), to an output string s. The top-level string expression P is a `Switch` constructor whose arguments are pairs of Boolean expressions b and trace expressions e. The set of Boolean expressions in a `Switch` construct must be disjoint, that is, for any input state, at most one of the Boolean expressions can be true. The value of P in a given input state σ is the value of the trace expression that corresponds to the Boolean expression satisfied by σ. A Boolean expression b is a propositional formula in Disjunctive Normal Form (DNF). A predicate `Match`(v_i, r, k) is satisfied if and only if v_i contains at least k nonoverlapping matches of regular expression r. (In general, any finite set of predicates can be used.)

A *trace expression* `Concatenate`($f_1, \ldots, f_n$) is the concatenation of strings represented by atomic expressions $f_1, \ldots, f_n$. An *atomic expression* f is either a constant-string expression `ConstStr`, a substring expression constructed from `SubStr`, or a loop expression constructed from `Loop`.

The substring expression `SubStr`(v_i, p_1, p_2) is defined partly by two *position expressions* p_1 and p_2, each of which implicitly refers to the (subject) string v_i and must evaluate to a position within the string v_i. (A string with $\square$ characters has $\square + 1$ positions, numbered from 0 to $\square$ starting from left.) `SubStr`(v_i, p_1, p_2) is the substring of string v_i in between positions p_1 and p_2. For a nonnegative

Figure 1. (a) Syntax of syntactic string-processing programs. (b) Data structure for representing a set of such programs.



constant k, CPos(k) denotes the k^{th} position in the subject string. For a negative constant k, CPos(k) denotes the $(\square + 1 + k)^{th}$ position in the subject string, where $\square$ = Length(s). Pos(r_1,r_2,c) is another position expression, where r_1 and r_2 are regular expressions and integer expression c evaluates to a nonzero integer. Pos(r_1,r_2,c) evaluates to a position t in the subject string s such that r_1 matches some suffix of $s[0{:}t]$, and r_2 matches some prefix of $s[t{:}\square]$, where $\square$ = Length(s). Furthermore, if c is positive (negative), then t is the $|c|^{th}$ such match starting from the left side (right side). We use the expression $s[t_1{:}t_2]$ to denote the substring of s between positions t_1 and t_2. The substring construct is quite expressive. For example, in the expression SubStr(v_i, Pos(r_1,r_2,c), Pos(r_3,r_4,c)), r_2 and r_3 describe the characteristics of the substring in v_i to be extracted, while r_1 and r_4 describe the characteristics of the surrounding delimiters. We use the expression SubStr2(v_i, r, c) as an abbreviation to denote the c^{th} occurrence of regular expression r in v_i, that is, SubStr(v_i, Pos(ε, r, c), Pos(r, ε, c)).

A regular expression r is ε (which matches the empty string, and therefore can match at any position of any string), a token T, or a token sequence TokenSeq(T_1, ..., T_n). This restricted choice of regular expressions enables efficient enumeration of regular expressions that match certain parts of a string. We use the following finite (but easily extended) set of tokens: (a) StartTok, which matches the beginning of a string, (b) EndTok, which matches the end of a string, (c) a token for each special character, such as hyphen, dot, semicolon, comma, slash, or left/right parenthesis/bracket, and (d) two tokens for each character class C, one that matches a sequence of one or more characters in C, and another that matches a sequence of one or more characters that are not in C. Examples of a character class C include numeric digits (0–9), alphabetic characters (a–zA–Z), lowercase alphabetic characters (a–z), uppercase alphabetic characters (A–Z), alphanumeric characters, and whitespace characters. UpperTok, NumTok, and AlphNumTok match a nonempty sequence of uppercase alphabetic characters, numeric digits, and alphanumeric characters, respectively. DecNumTok matches a nonempty sequence of numeric digits and/or decimal point. HyphenTok and SlashTok match the hyphen character and the slash character, respectively.

The task described in Example 1 can be expressed in our domain-specific language as

```
Switch ((b1, e1), (b2, e2)), where
b1 ≡ Match(v1, NumTok, 3)
b2 ≡ ¬Match(v1, NumTok, 3)
e1 ≡ Concatenate(SubStr2(v1, NumTok, 1), ConstStr("–"),
                 SubStr2(v1, NumTok, 2), ConstStr("–"),
                 SubStr2(v1, NumTok, 3))
e2 ≡ Concatenate(ConstStr("425–"), SubStr2(v1, NumTok, 1),
                 ConstStr("–"), SubStr2(v1, NumTok, 2))
```

The atomic expression Loop(λw : e) is the concatenation of e_1, e_2, ..., e_n, where e_i is obtained from e by replacing all occurrences of integer w by i, and n is the smallest integer such that evaluation of e_{n+1} is undefined. (It is also possible to define more interesting termination conditions, e.g., based on position expressions or predicates.) A trace expression e is undefined when (i) a constituent CPos(k) expression refers to a position not within its subject string, (ii) a constituent Pos(r_1, r_2, c) expression is such that the subject string does not contain c occurrences of a match bounded by r_1 and r_2, or (iii) a constituent SubStr(v_i, p_1, p_2) expression has position expressions that are both defined but the first refers to a position that occurs later in the subject string than the position indicated by the second. The following example illustrates the utility of the loop construct.

Example 2 (Generate Abbreviation). *The following task was presented originally as an* Advanced Text Formula.[23]

Input v_1	Output
Association of Computing Machinery	ACM
Principles Of Programming Languages	**POPL**
Foundations of Software Engineering	**FSE**

This task can be expressed in our language as

```
Loop(λw : Concatenate(SubStr2(v1, UpperTok, w))).
```

Our tool synthesizes this program from the first example row and uses it to produce the entries in the second and third rows (shown here in boldface for emphasis) of the output column.

3.2. Synthesis algorithm

The synthesis algorithm first computes, for each input–output example (σ, s), the set of *all* trace expressions that map input σ to output s (using procedure `Generate`). It then intersects these sets for similar examples and learns conditionals to handle different cases (using procedure `Intersect`). The size of such sets can be huge; therefore, we must develop a data structure that allows us to succinctly represent and efficiently manipulate huge sets of program expressions.

Data structure: Figure 1(b) describes our data structure for succinctly representing sets of programs from our domain-specific language. $\widetilde{P}$, $\widetilde{e}$, $\widetilde{f}$, and $\widetilde{p}$ denote representations of, respectively, a set of string programs, a set of trace expressions, a set of atomic expressions, and a set of position expressions. $\widetilde{r}$ and $\widetilde{c}$ represent a set of regular expressions and a set of integer expressions; these sets are represented explicitly.

The `Concatenate` constructor used in our string language is generalized to the `Dag` constructor `Dag`$(\widetilde{\eta}, \eta^s, \eta^t, \widetilde{\xi}, W)$, where $\widetilde{\eta}$ is a set of nodes containing two distinctly marked source and target nodes η^s and η^t, $\widetilde{\xi}$ is a set of edges over nodes in $\widetilde{\eta}$ that defines a Directed Acyclic Graph (DAG), and W maps each $\xi \in \widetilde{\xi}$ to a set of atomic expressions. The set of all `Concatenate` expressions represented by a `Dag`$(\widetilde{\eta}, \eta^s, \eta^t, \widetilde{\xi}, W)$ constructor includes exactly those whose ordered arguments belong to the corresponding edges on any path from η^s to η^t. The `Switch`, `Loop`, `SubStr`, and `Pos` constructors are all overloaded to construct sets of the corresponding program expressions that are shown in Figure 1(a). The `ConstStr` and `CPos` constructors can be regarded as producing singleton sets.

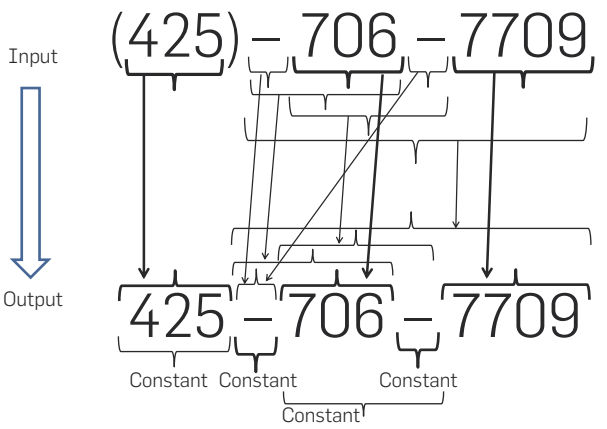
The data structure supports efficient implementation of various useful operations including intersection, enumeration of programs, and their simultaneous execution on a given input. The most interesting of these is the intersection operation, which is similar to regular automata intersection. The additional challenge is to intersect edge labels—in the case of automata, the labels are simply sets of characters, while in our case, the labels are sets of string expressions.

Procedure `Generate`: The number of trace expressions that can generate a given output string from a given input state can be huge. For example, consider the second input–output pair in Example 1, where the input state consists of one string "(425)-706-7709" and the output string is "425-706-7709". Figure 2 shows a small sampling of different ways of generating parts of the output string from the input string using `SubStr` and `ConstStr` constructors. Each substring extraction task itself can be expressed with a huge number of expressions, as explained later. The following are three of the trace expressions represented in the figure, of which only the second one, shown in the figure in bold, expresses the program expected by the user:

1. Extract substring "425". Extract substring "-706-7709".
2. Extract substring "425". Print constant "-". Extract substring "706". Print constant "-". Extract substring "7709".
3. Extract substring "425". Extract substring "-706". Print constant "-". Extract substring "7709".

We apply two crucial observations to succinctly generate and represent all such trace expressions. First, the logic for generating some substring of an output string is completely decoupled from the logic for generating another disjoint substring of the output string. Second, the total number of different substrings/parts of a string is quadratic (and not exponential) in the size of that string.

Figure 2. Small sampling of different ways of generating parts of an output string from the input string.



The `Generate` procedure creates a Directed Acyclic Graph (DAG) `Dag`$(\widetilde{\eta}, \eta^s, \eta^t, \widetilde{\xi}, W)$ that represents the *trace set* of all trace expressions that generate a given output string from a given input state. `Generate` constructs a node corresponding to each position within the output string and constructs an edge from a node corresponding to any position to a node corresponding to any later position. Each edge corresponds to some substring of the output and is annotated with the set of all atomic expressions that generate that substring. We describe below how to generate the set of all such `SubStr` expressions. Any `Loop` expressions are generated by first generating candidate expressions (by *unifying* the sets of trace expressions associated with the substrings $s[k_1 : k_2]$ and $s[k_2 : k_3]$, where k_1, k_2, and k_3 are the boundaries of the first two loop iterations, identified by considering all possibilities), and then validating them.

The number of substring expressions that can extract a given substring from a given string can be huge. For example, following is a small sample of various expressions that extract "706" from the string "425-706-7709" (call it v_1).

- Second number: `SubStr2`(v_1, `NumTok`, 2).
- 2nd last alphanumeric token: `SubStr2`(v_1, `AlphNumTok`, –2).
- Substring between the first hyphen and the last hyphen: `SubStr`(v_1, `Pos`(`HyphenTok`, ε, 1), `Pos`(ε, `HyphenTok`, –1)).
- First number that occurs between hyphen on both ends. `SubStr`(v_1, `Pos`(`HyphenTok`, `TokenSeq`(`NumTok`, `HyphenTok`), 1), `Pos`(`TokenSeq`(`HyphenTok`, `NumTok`), `HyphenTok`, 1)).
- First number preceded by a number–hyphen sequence. `SubStr`(v_1, `Pos`(`TokenSeq`(`NumTok`, `HyphenTok`), `NumTok`, 1), `Pos`(`TokenSeq`(`NumTok`, `HyphenTok`, `NumTok`), ε, 1)).

The substring-extraction problem can be decomposed into two *independent* position-identification problems, each of which can be solved independently. The solutions to the substring-extraction problem can also be maintained succinctly by independently representing the solutions to the two position-identification problems. Note the representation of the SubStr constructor in Eq. 1 in Figure 1(b).

Procedure Intersect: Given a trace set for each input–output example, the Intersect procedure generates the top-level Switch constructor. Intersect first partitions the examples, so that inputs in the same partition are handled by the same conditional in the top-level Switch expression, and then intersects the trace sets for inputs in the same partition. If a set of inputs are in the same partition, then the intersection of trace sets is nonempty. Intersect uses a greedy heuristic to minimize the number of partitions by starting with singleton partitions and then iteratively merging partitions that have the highest *compatibility score*, which is a function of the size of the resulting partition and its potential to be merged with other partitions.

Intersect then constructs a classifier for each of the resultant partitions, which is a Boolean expression that is satisfied by exactly the inputs in the partition. The classifier for each partition and the intersection of trace sets for the inputs in the partition serve as the Boolean condition and corresponding trace expression in the constructed Switch expression, respectively.

Ranking: We prefer Concatenate and TokenSeq expressions that have fewer arguments. We prefer SubStr expressions to both ConstStr expressions (it is less likely for constant parts of an output string to also occur in the input) and Concatenate expressions (if there is a long substring match between the input and output, it is more likely that the corresponding part of the output was produced by a single substring extraction). We prefer a Pos expression to CPos expression (giving less preference to extraction expressions based on constant offsets). StartTok and EndTok are our most preferred tokens; otherwise, we prefer tokens corresponding to a larger character class (favoring generality).

The implementation of the synthesis algorithm is less than 5,000 lines of C# code, and takes less than 0.1 s on average for a benchmark suite of more than 100 tasks obtained from online help forums and the Excel product team.

4. SEMANTIC TRANSFORMATIONS

Some string transformation tasks also involve manipulating strings that need to be interpreted as more than a sequence of characters, for example, as a column entry from some relational table, or as some standard data type such as date, time, currency, or phone number. For example, consider the following task from an Excel help forum.

Example 3. *A shopkeeper wants to compute the selling price of an item (Output) from its name (Input v_1) and selling date (Input v_2). The inventory database of the shop consists of two tables: (i) MarkupRec table that stores id, name and markup percentage of items, and (ii) CostRec table that stores id, purchase date (in month/year format), and purchase price of items. The selling price of an item is computed by adding its purchase price (for the corresponding month) to its markup charges, which in turn is calculated by multiplying the markup percentage by the purchase price.*

Input v_1	Input v_2	Output
Stroller	10/12/2010	$145.67 + 0.30*145.67
Bib	23/12/2010	$3.56 + 0.45*3.56
Diapers	21/1/2011	**$21.45 + 0.35*21.45**
Wipes	2/4/2009	**$5.12 + 0.40*5.12**
Aspirator	23/2/2010	**$2.56 + 0.30*2.56**

MarkupRec

Id	Name	Markup
S33	Stroller	30%
B56	Bib	45%
D32	Diapers	35%
W98	Wipes	40%
A46	Aspirator	30%
…	…	…

CostRec

Id	Date	Price
S33	12/2010	$145.67
S33	11/2010	$142.38
B56	12/2010	$3.56
D32	1/2011	$21.45
W98	4/2009	$5.12
A46	2/2010	$2.56
…	…	…

To perform the above task, the user must perform a join of the two tables on the common item Id column to lookup the item Price from its Name (v_1) and selling Date (substring of v_2). We present an extension to the trace expression (from Section 3.1) that can also manipulate strings present in such relational tables.[18]

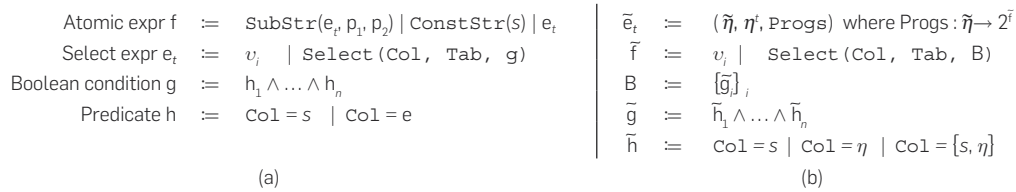
4.1. Domain-specific language

We extend the trace expression (from Section 3.1), as shown in Figure 3(a), to obtain the semantic string transformation language that can also perform table lookup operations. The atomic expression f is modified to represent a constant string, a select expression, or a substring of a select expression. A *select expression* e_t is either an input string variable v_i or a lookup expression denoted by Select(Col, Tab, g), where Tab is a relational table identifier and Col is a column identifier of the table. The Boolean condition g is an ordered conjunction of column predicates $h_1 \wedge \ldots \wedge h_n$, where a column predicate h is an equality comparison between the content of some column of the table and a constant or a trace expression e. We require columns present in these conditions to together constitute a *primary key* of the table to ensure that the select queries produce a single string as opposed to a set of strings.

The task in Example 3 can be represented in the language as

Concatenate (f_1, ConstStr("+0."), f_2, ConstStr("*"), f_3)
where $f_1 \equiv$ Select(Price, CostRec, Id = $f_4 \wedge$ Date = f_5),
$f_4 \equiv$ Select(Id, MarkupRec, Name = v_1),
$f_5 \equiv$ SubStr(v_2, Pos(SlashTok, ε, 1), Pos(ε, EndTok, 1)),
$f_2 \equiv$ SubStr2(f_6, NumTok, 1), $f_3 \equiv$ SubStr2(f_1, DecNumTok, 1),
$f_6 \equiv$ Select(Markup, MarkupRec, Name = v_1).

Figure 3. Extensions to the syntax and data structure in Figure 1 for semantic processing.



The expression f_4 looks up the `Id` of the item (present in v_1) from the `MarkupRec` table and f_5 generates a substring of the date present in v_2, which are then used together to lookup the `Price` of the item from the `CostRec` table (f_1). The expression f_6 looks up the `Markup` percentage of the item from the `MarkupRec` table and f_2 generates a substring of this lookup value by extracting the first numeric token (thus removing the % sign). Similarly, the expression f_3 generates a substring of f_1, removing the $ symbol. Finally, the top-level expression concatenates the strings obtained from expressions f_1, f_2, and f_3 with constant strings "+0." and "*".

This extended language also enables manipulation of strings that represent standard data types, whose semantic meaning can be encoded as a database of relational tables. For example, consider the following date manipulation task.

Example 4 (Date Manipulation). *An Excel user wanted to convert dates from one format to another, and the fixed set of hard-coded date formats supported by Excel 2010 do not match the input and output formats. Thus, the user solicited help on a forum.*

Input v_1	Output
6-3-2008	Jun 3rd, 2008
3-26-2010	**Mar 26th, 2010**
8-1-2009	**Aug 1st, 2009**
9-24-2007	**Sep 24th, 2007**

We can encode the required background knowledge for the date data type in two tables, namely a `Month` table with 12 entries: (1, `January`), ..., (12, `December`) and a `DateOrd` table with 31 entries (1, `st`), (2, `nd`), ..., (31, `st`). The desired transformation is represented in our language as

```
Concatenate(SubStr(Select(MW, Month, MN = e1),
       Pos(StartTok, ε, 1), CPos(3)), ConstStr(" "), e2,
       Select(Ord, DateOrd, Num = e2), ConstStr(", "), e3)
```

where e_1 = `SubStr2`(v_1, `NumTok`, 1), e_2 = `SubStr2`(v_1, `NumTok`, 2), e_3 = `SubStr2`(v_1, `NumTok`, 3). (`MW`,`MN`) and (`Num`,`Ord`) denote the columns of the `Month` and `DateOrd` tables, respectively.

4.2. Synthesis algorithm

We now describe the key extensions to the synthesis algorithm for syntactic transformations (Section 3.2) to obtain the synthesis algorithm for semantic transformations.

Data structure: Figure 3(b) describes the data structure that succinctly represents the large set of programs in the semantic transformation language that are consistent with a given input–output example. The data structure consists of a generalized expression $\tilde{e}_t$, generalized Boolean condition $\tilde{g}$, and generalized predicate $\tilde{h}$ (which, respectively, denote a set of select expressions, a set of Boolean conditions g, and a set of predicates h). The generalized expression $\tilde{e}_t$ is represented using a tuple ($\tilde{\eta}$, η^t, `Progs`) where $\tilde{\eta}$ denotes a set of nodes containing a distinct target node η^t (representing the output string), and `Progs` : $\tilde{\eta} \rightarrow 2^{\tilde{f}}$ maps each node $\eta \in \tilde{\eta}$ to a set consisting of input variables v_i or generalized select expressions `Select(Col, Tab, B)`. A generalized Boolean condition $\tilde{g}_i$ corresponds to some primary key of table T and is a conjunction of generalized predicates $\tilde{h}_j$, where each $\tilde{h}_j$ is an equality comparison of the j^{th} column of the corresponding primary key with a constant string s or some node $\tilde{\eta}$ or both. The two key aspects of this data structure are (i) the use of intermediate nodes for sharing sub-expressions to represent an exponential number of expressions in polynomial space and (ii) the use of Conjunctive Normal Form (CNF) form of Boolean conditions to represent an exponential number of conditionals $\tilde{g}$ in polynomial space.

Procedure `Generate`: We first consider the simpler case where there are no syntactic manipulations on the table lookups and the lookups are performed using exact string matches, that is, the predicate h is `Col` = e_t instead of `Col` = e. The `Generate` procedure operates by iteratively computing a set of nodes ($\tilde{\eta}$), where each node $\eta \in \tilde{\eta}$ represents a string `val`(η) that corresponds to some table entry or an input string. `Generate` performs an iterative forward reachability analysis of the string values that can be generated in a single step (i.e., using a single `Select` expression) from the string values computed in the previous step based on *string equality* and assigns the `Select` expression to the `Progs` map of the corresponding node. The base case of the procedure creates a node for each of the input string variables. After performing the analysis for a bounded number of iterations, the procedure returns the set of select expressions of the node that corresponds to the output string s, that is, `Progs`[`val`$^{-1}$(s)].

The `Generate` procedure for the general case, which also includes syntactic manipulations on table lookups, requires a relaxation of the above-mentioned reachability criterion of strings that is based on string equality. We now

define a table entry to be reachable from a set of previously reachable strings if the entry can be generated from the set of reachable strings using the `Generate` procedure of Section 3.2. The rest of the reachability algorithm operates just as before.

Procedure `Intersect`: A basic ingredient of the `Intersect` procedure for syntactic transformations is a method to intersect two `Dag` constructs, each representing a set of trace expressions. We replace this by a method to intersect two tuples ($\widetilde{\eta}_1$, η_1^t, $\texttt{Progs}_1$) and ($\widetilde{\eta}_2$, η_2^t, $\texttt{Progs}_2$), each representing a set of extended trace expressions. The tuple after intersection is ($\widetilde{\eta}_1 \times \widetilde{\eta}_2$, ($\eta_1^t$,$\eta_2^t$), $\texttt{Progs}_{12}$), where $\texttt{Progs}_{12}((\widetilde{\eta}_1, \widetilde{\eta}_2))$ is given by the intersection of $\texttt{Progs}_1(\widetilde{\eta}_1)$ and $\texttt{Progs}_2(\widetilde{\eta}_2)$.

Ranking: We prefer expressions of smaller depth (fewer nested chains of `Select` expressions) and ones that match longer strings in table entries for indexing. We prefer lookup expressions that use distinct tables (for join queries) as opposed to using the same table twice. We prefer conditionals with fewer predicates. We prefer predicates that compare columns with other table entries or input variables (as opposed to comparing columns with constant strings).

We implemented our algorithm as an extension to the Excel add-in (Section 3.2) and evaluated it successfully over more than 50 benchmark problems obtained from various help forums and the Excel product team. For each benchmark, our implementation learned the desired transformation in <10 s (88% of them taking <1 s each) requiring at most three input–output examples (with 70% of them requiring only one example). The data structure had size between 100 and 2,000 (measured as the number of terminal symbols in the data-structure syntax), with an average size of 600, and typically represented 10^{20} expressions.

5. TABLE LAYOUT TRANSFORMATIONS

End users often transform a spreadsheet table not by changing the data stored in the cells of a table, but instead by changing how the cells are grouped or arranged. In other words, users often transform the *layout* of a table.[8]

Example 5. *The following example input table and subsequent example output table were provided by a novice on an Excel user help thread to specify a layout transformation:*

	Qual 1	Qual 2	Qual 3
Andrew	01.02.2003	27.06.2008	06.04.2007
Ben	31.08.2001		05.07.2004
Carl		18.04.2003	09.12.2009

Andrew	Qual 1	01.02.2003
Andrew	Qual 2	27.06.2008
Andrew	Qual 3	06.04.2007
Ben	Qual 1	31.08.2001
Ben	Qual 3	05.07.2004
Carl	Qual 2	18.04.2003
Carl	Qual 3	09.12.2009

The example input contains a set of dates on which tests were given, where each date is in a row corresponding to the name of the test taker, and in a column corresponding to the name of the test. For every date, the user needs to produce a row in the output table containing the name of the test taker, the name of the test, and the date on which the test was taken. If a date cell in the input is empty, then no corresponding row should be produced in the output.

5.1. Domain-specific language

We may view every program P that transforms the layout of a table as constructing a map m_P from the cells of an input table to the coordinates of the output table. For a cell c in an input table, if $m_P(c) = (\text{row}, \text{col})$, P fills the cell in the output table at the coordinate (row, col) with the data in c. A program in our language of layout transformations is defined syntactically as a finite collection of *component programs*, each of which builds a map from input cells to output coordinates (Figure 4: table program). We designed our language on the principle that most layout transformations can be implemented by a set of component programs that construct their map using one of the two complementary procedures: *filtering* and *associating*.

When a component program filters, it scans the cells of the input table in row-major order, selects a subset of the cells, and maps them in order to a subrange of the coordinates in the output table. A *filter program* $\texttt{Filter}(\varphi, \texttt{SEQ}_{i,j,k})$ (Figure 4: filter program) is a *mapping condition* φ, which is a function whose body is a conjunction of predicates over input cells drawn from a fixed set and an *output sequencer* $\texttt{SEQ}_{i,j,k}$, where i, j, and k are nonnegative integers. For a mapping condition φ and sequencer $\texttt{SEQ}_{i,j,k}$, the filter program $\texttt{Filter}(\varphi, \texttt{SEQ}_{i,j,k})$ scans an input table and maps each cell that satisfies φ to the coordinates in the output table between columns i and j, starting at row k, in row-major order.

For the tables in Example 5, the filter program $F_1 = \texttt{Filter}(\lambda c.(c.data \neq \text{“”} \wedge c.col \neq 1 \wedge c.row \neq 1), \texttt{SEQ}_{3,3,1})$ maps each date, that is, each nonempty cell not in column 1 and not in row 1, to its corresponding cell in column 3 of the output table, starting at row 1. Call this map m_{F_1}.

A table program can also construct a map using spatial relationships between cells in the input table and spatial relationships between coordinates in the output table; we call this construction *association*. When a table program associates, it takes a cell c in the input table mapped by some filter program F, picks a cell c_1 in the input table whose coordinate is related to c, finds the coordinate $m_F(c)$ that c maps to under m_P, picks a coordinate d_1 whose coordinate is related to $m_F(c)$, and maps c_1 to d_1.

An associative program $A = \texttt{Assoc}(F, s_0, s_1)$ (Figure 4: Associative program) is constructed from a filter program F and two *spatial functions* s_0 and s_1, each of which may be of

Figure 4. Syntax of layout transformations.

Table program $P := \texttt{TabProg}([\mathsf{K}]_i)$
Component program K := F | A
Filter program F := $\texttt{Filter}(\varphi, \texttt{SEQ}_{i,j,k})$
Associative program A := $\texttt{Assoc}(\mathsf{F}, \mathsf{S}_1, \mathsf{S}_2)$
Spatial function S := $\texttt{RelCol}_i$ | $\texttt{RelRow}_j$

the form $\texttt{RelCol}_i$ or $\texttt{RelRow}_j$. The spatial function $\texttt{RelCol}_i$ takes a cell c as input, and returns the cell in the same row as c and in column i. The spatial function $\texttt{RelRow}_j$ takes a cell c as input, and returns the cell in row j and in the same column as c. For each cell c in the domain of m_K, the map of A contains an entry $m_A(s_0(c)) = s_1(m_F(c))$.

For the example tables in Example 5, and the filter program F_1 introduced above that maps to column 3 of the example output table, the associative program $A_1 = \texttt{Assoc}(F_1, \texttt{RelCol}_1, \texttt{RelCol}_1)$ constructs a map to every cell in column 1 of the output table. To construct its map, A_1 takes each cell c in the input table mapped by F_1, finds the cell $\texttt{RelCol}_1(c)$ in the same row as c and in column 1, finds the coordinate $m_{F_1}(c)$ that F_1 maps c to, finds the coordinate $\texttt{RelCol}_1(m_{F_1}(c))$, and maps $\texttt{RelCol}_1(c)$ to $\texttt{RelCol}_1(m_{F_1}(c))$: that is, A_1 sets $m_{A_1}(\texttt{RelCol}_1(c)) = \texttt{RelCol}_1(m_{F_1}(c))$. Similarly, the associative program $A_2 = \texttt{Assoc}(F_1, \texttt{RelRow}_1, \texttt{RelCol}_2)$ constructs a map to every cell in column 2 of the example output table. The table program $\texttt{TabProg}(\{F_1, A_1, A_2\})$ takes the input table in Example 5 and maps to every cell in the output table.

It is possible that the ranges of constituent component programs of a table program may overlap on a given input table. In such a case, if two cells with different values are mapped to the same output coordinate, then we say that the table program is undefined on the input table.

5.2. Synthesis algorithm

The synthesis algorithm generates the set of all table programs that are consistent with each example, intersects the sets, and picks a table program from the intersection that is consistent with all of the examples.

Data structure for sets of table programs: To compactly represent sets of table programs, our synthesis algorithm uses a table program itself. Let a component program K be *consistent* with an input–output example if when K is applied to the example input and K maps an input cell c, then the cell in the output table at coordinate $m_K(c)$ has the same data as cell c in the input table. Let a set of component programs $\mathcal{K}$ *cover* an example if, for each cell coordinate d in the example output, there is some component $K \in \mathcal{K}$ and cell c in the input table such that $d = m_K(c)$. Let a table program $\texttt{TabProg}(\mathcal{K})$ be *consistent with an example* if $\mathcal{K}$ is consistent with the example and $\mathcal{K}$ covers the example. For a fixed input–output example, $\texttt{TabProg}(\mathcal{K})$ *stores* $\texttt{TabProg}(\mathcal{K}')$ if $\mathcal{K}' \subseteq \mathcal{K}$ covers the example.

Procedure `Generate`**:** From a single input–output example, `Generate` constructs a table program that stores the set of all table programs that are consistent with the example by constructing the set $\mathcal{K}^*$ of all consistent component programs, in three steps. First, from the example input and output, `Generate` defines a set of spatial functions and map predicates. Second, from the set of map predicates, `Generate` collects the set of all consistent filter programs. Third, from the set of consistent filter programs, `Generate` constructs the set of consistent associative programs. To generate associative programs, `Generate` combines each consistent filter program with pairs of spatial functions defined in the first step and checks if the resulting associative program is consistent. If so, then `Generate` adds the associative program to the set of consistent component programs. The table program $\texttt{TabProg}(\mathcal{K}^*)$ stores all table programs that are consistent with the example if any exist, and is thus called the *complete table program* of the example.

Procedure `Intersect`**:** Given two sets of table programs represented as table programs stored in $\texttt{TabProg}(\mathcal{K}_0)$ and $\texttt{TabProg}(\mathcal{K}_1)$, `Intersect` efficiently constructs the intersection of the sets as all consistent table programs stored in $\texttt{TabProg}(\mathcal{K}_0 \cap \mathcal{K}_1)$.

The synthesis algorithm applies `Generate` to construct the complete table program for each example, applies `Intersect` to the set of complete table programs, and checks if the resulting table program $\texttt{TabProg}(\mathcal{K}_I)$ is consistent with each of the examples. If so, it returns $\texttt{TabProg}(\mathcal{K}'_I)$ for some subset $\mathcal{K}'_I$ of $\mathcal{K}_I$ that covers each of the examples. The exact choice of $\mathcal{K}'_I$ depends on the ranking criteria.

Ranking: We prefer table programs that are constructed from smaller sets of component programs, as such table programs are intuitively simpler. The subset order over sets of component programs thus serves as a partial order for ranking. Also, suppose that a table program P_0 uses a filter program F_0, while another table program P_1 uses a filter program F_1 that builds the same map as F_0, but whose condition is a conjunction of fewer predicates than the condition of F_0. Then, we prefer P_1, as the condition used by F_1 is intuitively more general.

To evaluate our synthesis algorithm, we implemented it as a plug-in for the Excel spreadsheet program and applied it to input–output tasks taken directly from over 50 real-world Excel user help threads. The tool automatically inferred programs for each task. The tool usually ran in under 10 s and inferred a program that behaved as we expected, given the user's description in English of their required transformation. If the highest-ranking program inferred by the tool behaved in an unexpected way on an input, it inferred a program that behaved as expected after taking at most two additional examples.

6. RELATED WORK

The Human Computer Interaction (HCI) community has developed *programming by demonstration* (PBD) systems[1] for data cleaning, which require the user to specify a complete demonstration or trace visually on the data instead of writing code: SMARTedit[14] for text manipulation, Simultaneous Editing[16] for string manipulation, and Wrangler[11] for table transformations. Our system is based on programming by example (as opposed to demonstration); it requires the user to provide only the initial and final states, as opposed to also providing the intermediate states. This renders our system more usable,[13] but at the expense of complicating the learning problem. Furthermore, we synthesize programs over a more sophisticated language consisting of conditionals and loops.

The database community has studied the *view synthesis* problem,[2, 22] which aims to find a succinct query for a given relational view instance. Our semantic string

transformation synthesis also infers similar queries, but infers from very few examples and over a richer language that combines lookup operations with syntactic manipulations. Our table layout synthesis addresses the challenges of dealing with spreadsheet tables, which, unlike database relations, have ordering relationships between rows and carry meta-information in some cells.

The PADS project in the programming languages community has simplified ad hoc data-processing tasks for programmers by developing domain-specific languages for describing data formats, and learning algorithms for inferring such formats using annotations provided by the user.[3] The learned format can then be used by programmers to implement custom data-analysis tools. In contrast, we focus on automating the entire end-to-end process for relatively simpler tasks, which include not only learning the text structure from the inputs, but also learning the desired transformation from the outputs, without any user annotations.

The area of program synthesis is gaining renewed interest. However, it has traditionally focused on synthesizing nontrivial algorithms[20] (e.g., graph algorithms[9] and program inverses[19]) and discovering intricate code-snippets (e.g., bit-vector tricks,[7] switching logic in hybrid systems[21]). In this paper, we apply program synthesis to discover simpler programs required by the much larger class of spreadsheet end-users. Various classes of techniques have been explored for program synthesis: exhaustive search, logical reasoning, probabilistic inference, and version-space algebras (for a recent survey, see Gulwani[5]). Because the data manipulations required by end users are usually relatively simple, we can apply version-space algebras, which allow real-time performance, unlike other techniques. Version-space algebras were pioneered by Mitchell for refinement-based learning of Boolean functions,[17] while Lau et al. extended the concept to learning more complex functions in a PBD setting.[14] Our synthesis algorithms lift the concepts of version-space algebra to the PBE setting, for a fairly expressive string expression language involving conditionals and loops.

7. CONCLUSION AND FUTURE WORK

General-purpose computational devices, such as smartphones and computers, are becoming accessible to people at large at an impressive rate. In the future, even robots will become household commodities. Unfortunately, programming such general-purpose platforms has never been easy, because we are still mostly stuck with the model of providing step-by-step, detailed, and syntactically correct instructions on *how* to accomplish a certain task, instead of simply describing *what* the task is. Program synthesis has the potential to revolutionize this landscape, when targeted for the right set of problems and using the right interaction model.

This paper reports our initial experience with designing domain-specific languages and inductive synthesizers for string and table transformations. Our choice of domains was motivated by our study of help-forum problems that end users struggled with. A next step is to develop a general framework that can allow synthesizer writers to easily develop domain-specific synthesizers of the kind described in this paper, similar to how declarative parsing frameworks allow a compiler writer to easily write a parser.

Acknowledgments

We thank Guy L. Steele Jr. for providing insightful and detailed feedback on multiple versions of this draft. C

References

1. Cypher, A., ed. *Watch What I Do: Programming by Demonstration*, MIT Press, 1993.
2. Das Sarma, A., Parameswaran, A., Garcia-Molina, H., Widom, J. Synthesizing view definitions from data. In *ICDT* (2010).
3. Fisher, K., Walker, D. The PADS project: an overview. In *ICDT* (2011).
4. Gualtieri, M. Deputize end-user developers to deliver business agility and reduce costs. In *Forrester Report for Application Development and Program Management Professionals* (Apr. 2009).
5. Gulwani, S. Dimensions in program synthesis. In *PPDP* (2010).
6. Gulwani, S. Automating string processing in spreadsheets using input-output examples. In *POPL* (2011).
7. Gulwani, S., Jha, S., Tiwari, A., Venkatesan, R. Synthesis of loop-free programs. In *PLDI* (2011).
8. Harris, W.R., Gulwani, S. Spreadsheet table transformations from examples. In *PLDI* (2011).
9. Itzhaky, S., Gulwani, S., Immerman, N., Sagiv, M. A simple inductive synthesis methodology and its applications. In *OOPSLA* (2010).
10. Jha, S., Gulwani, S., Seshia, S., Tiwari, A. Oracle-guided component-based program synthesis. In *ICSE* (2010).
11. Kandel, S., Paepcke, A., Hellerstein, J., Heer, J. Wrangler: Interactive visual specification of data transformation scripts. In *CHI* (2011).
12. Ko, A.J., Myers, B.A., Aung, H.H. Six learning barriers in end-user programming systems. In *VL/HCC* (2004).
13. Lau, T. Why P B D systems fail: Lessons learned for usable AI. In *CHI 2008 Workshop on Usable AI* (2008).
14. Lau, T., Wolfman, S., Domingos, P., Weld, D. Programming by demonstration using version space algebra. *Mach. Learn. 53*(1–2) (2003).
15. Lieberman, H. *Your Wish is My Command: Programming by Example*, Morgan Kaufmann, 2001.
16. Miller, R.C., Myers, B.A., Interactive simultaneous editing of multiple text regions. In *USENIX Annual Technical Conference* (2001).
17. Mitchell, T.M. Generalization as search. *Artif. Intell. 18*, 2 (1982).
18. Singh, R., Gulwani, S. Learning semantic string transformations from examples. *PVLDB 5* (2012), in press.
19. Srivastava, S., Gulwani, S., Chaudhuri, S., Foster, J.S. Path-based inductive synthesis for program inversion. In *PLDI* (2011).
20. Srivastava, S., Gulwani, S., Foster, J. From program verification to program synthesis. In *POPL* (2010).
21. Taly, A., Gulwani, S., Tiwari, A. Synthesizing switching logic using constraint solving. In *VMCAI* (2009).
22. Tran, Q.T., Chan, C.Y., Parthasarathy, S. Query by output. In *SIGMOD* (2009).
23. Walkenbach, J. *Excel 2010 Formulas*, John Wiley and Sons, 2010.

Sumit Gulwani (sumitg@microsoft.com), Microsoft Research, Redmond, WA.

William R. Harris (wrharris@cs.wisc.edu), Univ. of Wisconsin, Madison, WI.

Rishabh Singh (rishabh@csail.mit.edu), MIT CSAIL, Cambridge, MA.

DOI:10.1145/2240236.2240261

Technical Perspective
Proving Programs Continuous

By Andreas Zeller

PROVING A PROGRAM'S correctness is usually an all-or-nothing game. Either a program is correct with respect to its specification or it is not. If our proof succeeds, we have 100% correctness; if our proof does not succeed, we have nothing. Formal correctness proofs are difficult, because one must symbolically cover the entire range of possible inputs—and the slightest gap in the input leaves us with a gap in the proof.

But what if it turned out the risk posed by leaving a gap is actually small? This, of course, is the assumption of testing: If I tested some function with a sample of values, and it works correctly for this sample, I have reasons to assume it will also work for similar values. This is something I can assume if the behavior of my function is *continuous*—if it computes the correct square root for 10, 100, and 1,000, it should also do the right thing for any value in between.

One may think this is a dangerous assumption: Simply because my program has worked well for these three values, why should it work for any other? A program is free to do anything; since it need not obey mathematical or physical laws, it can behave in an entirely different way for any new value. This logical view is true in principle. In real life, however, programmers prefer abstractions that are easy to understand and to reason about. The reason testing works in practice is that programmers naturally strive toward continuity.

While the intuitive idea is easy to grasp, the concept of continuity so far has widely evaded formal treatment; in particular, it was not possible to automatically reason about continuity in the presence of loops. This is where the work of Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublinerman comes into play. Their framework can formally verify programs for continuity, proving that small changes to the input only cause small changes to the output. They show that several programs such as sorting or shortest path are continuous—or even Lipschitz continuous, implying that perturbations to a function's input cause only proportional changes to its output. Such a function would also be declared robust, meaning it will behave predictably even when inputs are uncertain or erroneous.

Being able to formally reason about continuity and robustness lets us see programs as driven not only by logic, but also analytical calculus; and this view can be very helpful for understanding why programs generally tend to work well even if only coarsely tested.

Being able to formally reason about continuity and robustness lets us see programs as driven not only by logic, but also analytical calculus; and this view can be very helpful for understanding why programs generally tend to work well even if only coarsely tested. This work also bridges the gap between programs and control theory, allowing for ample cross-fertilization between the fields; indeed, one can think of mathematical optimizations of program code just as the adoption of programming concepts by control theory. So, should we treat programs as driven by logic, by calculus, or both? I encourage you to read the following paper to see the manifold connections between logic and calculus in computer programs.

Andreas Zeller (zeller@cs.uni-saarland.de) is a professor of software engineering at Saarland University in Saarbrücken, Germany.

DOI:10.1145/2240236.2240262

Continuity and Robustness of Programs

By Swarat Chaudhuri, Sumit Gulwani, and Roberto Lublinerman

Abstract

Computer scientists have long believed that software is different from physical systems in one fundamental way: while the latter have continuous dynamics, the former do not. In this paper, we argue that notions of continuity from mathematical analysis are relevant and interesting even for software. First, we demonstrate that many everyday programs are *continuous* (i.e., arbitrarily small changes to their inputs only cause arbitrarily small changes to their outputs) or *Lipschitz continuous* (i.e., when their inputs change, their outputs change at most proportionally). Second, we give an mostly-automatic framework for verifying that a program is continuous or Lipschitz, showing that traditional, discrete approaches to proving programs correct can be extended to reason about these properties. An immediate application of our analysis is in reasoning about the *robustness* of programs that execute on uncertain inputs. In the longer run, it raises hopes for a toolkit for reasoning about programs that freely combines logical and analytical mathematics.

1. INTRODUCTION

It is accepted wisdom in computer science that the dynamics of software systems are inherently discontinuous, and that this fact makes them fundamentally different from physical systems. More than 25 years ago, Parnas[15] attributed the difficulty of engineering reliable software to the fact that "the mathematical functions that describe the behavior of [software] systems are not continuous." This meant that the traditional analytical calculus—the mathematics of choice when one is analyzing the dynamics of, say, fluids—did not fit the needs of software engineering too well. Logic, which can reason about discontinuous systems, was better suited to being the mathematics of programs.

In this paper, we argue that this wisdom is to be taken with a grain of salt. First, many everyday programs are continuous in the same sense as in analysis—that is, arbitrarily small changes to its inputs lead to arbitrarily small changes to its outputs. Some of them are even *Lipschitz continuous*—that is, perturbations to the program's inputs lead to at most proportional changes to its outputs. Second, we show that analytic properties of programs are not at odds with the classical, logical methods for program verification, giving a logic-based, mostly-automated method for formally verifying that a program is continuous or Lipschitz continuous. Among the immediate applications of this analysis is reasoning about the *robustness* of programs that execute under uncertainty. In the longer run, it raises hopes for a unified theory of program analysis that marries logical and analytical approaches.

Now we elaborate on the above remarks. Perhaps the most basic reason why software systems can violate continuity is *conditional branching*—that is, constructs of the form "**if** B **then** P_1 **else** P_2." Continuous dynamical systems arising in the physical sciences do not typically contain such constructs, but most nontrivial programs do. If a program has a branch, then even the minutest perturbation to its inputs may cause it to evaluate one branch in place of the other. Thus, we could perhaps conclude that any program containing a branch is ipso facto discontinuous.

To see that this conclusion is incorrect, consider the problem of sorting an array of numbers, one of the most basic tasks in computing. Every classic algorithm for sorting contains conditional branches. But let us examine the *specification* of a sorting algorithm: a mathematical function *Sort* that maps arrays to their sorted permutations. This specification is not only continuous but Lipschitz continuous: change any item of an input array A by $\pm\epsilon$, and each item of $Sort(A)$ changes at most by $\pm\epsilon$. For example, suppose A and A' are two input arrays as below, with A' obtained by perturbing each item of A at most by ± 1. Then $Sort(A')$ can be obtained by perturbing each item of $Sort(A)$ at most by ± 1.

$$A: 1,3,5,3,4 \qquad A': 1,2,4,4,3$$
$$Sort(A): 1,3,3,4,5 \qquad Sort(A'): 1,2,3,4,4$$

Similar observations hold for many of the classic computations in computer science, for example, shortest path and minimum spanning tree algorithms. Our program analysis extends and automates methods from the traditional analytical calculus to prove the continuity or Lipschitz continuity of such computations. For instance, to verify that a conditional statement within a program is continuous, we generalize the sort of insight that a high-school student uses to prove the continuity of a piecewise function like

$$\mathrm{abs}(x) = \begin{cases} -x & \text{if} \quad x < 0 \\ x & \text{if} \quad x \geq 0. \end{cases}$$

This paper is based on two previous works: "Continuity Analysis of Programs," by S. Chaudhuri, S. Gulwani, and R. Lublinerman, published in *POPL* (2010), 57–70, and "Proving Programs Robust," by S. Chaudhuri, S. Gulwani, R. Lublinerman, and S. Navidpour, published in *FSE* (2011), 102–112.

Intuitively, abs(x) is continuous because its two "pieces" x and $-x$ are continuous, and because x and $-x$ agree on values in the neighborhood of $x = 0$, the point where a small perturbation can cause abs(x) to switch from evaluating one piece to evaluating the other. Our analysis uses the same idea to prove that "**if** B **then** P_1 **else** P_2" is continuous: it inductively verifies that P_1 and P_2 are continuous, then checks, often automatically, that P_1 and P_2 become semantically equivalent at states where the value of B can flip on a small perturbation.

When operating on a program with loops, our analysis searches for an inductive proof of continuity. To prove that a continuous program is Lipschitz continuous, we inductively compute a collection of *Lipschitz matrices* that contain numerical bounds on the slopes of functions computed along different control paths of the program.

Of course, complete software systems are rarely continuous. However, verification technique like ours allows us to identify modules of a program that satisfy continuity properties. A benefit of this is that such modules are amenable to analysis by continuous methods. In the longer run, we can imagine a reasoning toolkit for programs that combines continuous analysis techniques, for example, numerical optimization or symbolic integration, and logical methods for analyzing code. Such a framework would expand the classical calculus to functions encoded as programs, a representation worthy of first-class treatment in an era where much of applied mathematics is computational.

A more immediate application of our framework is in the analysis of programs that execute on *uncertain* inputs, for example noisy sensor data or inexact scientific measurements. Unfortunately, traditional notions of functional correctness do not guarantee predictable program execution on uncertain inputs: a program may produce the correct output on each individual input, but even small amounts of noise in the input could change its output radically. Under uncertainty, traditional correctness properties must be supplemented by the property of *robustness*, which says that small perturbations to program's inputs do not have much effect on the program's output. Continuity and Lipschitz continuity can both serve as definitions of robustness, and our analysis can be used to prove that a program is robust.

The rest of the paper is organized as follows. In Section 2, we formally define continuity and Lipschitz continuity of programs and give a few examples of computations that satisfy these properties. In Section 3, we give a method for verifying a program's continuity, and then extend it to an analysis for Lipschitz continuity. Related work is presented in Section 4; Section 5 concludes the paper with some discussion.

2. CONTINUITY, LIPSCHITZ CONTINUITY, AND ROBUSTNESS

In this section, we define continuity[2] and Lipschitz continuity[3] of programs and show how they can be used to define robustness. First, however, we fix the programming language Imp whose programs we reason about.

Imp is a "core" language of imperative programs, meaning that it supports only the most central features of imperative programming—assignments, branches, and loops. The language has two discrete data types—integers and arrays of integers—and two continuous data types—reals and arrays of reals. Usual arithmetic and comparisons on these types are supported. In conformance with the model of computation under which algorithms over reals are typically designed, our reals are infinite-precision, and elementary operations on them are assumed to be given by unit-time oracles.

Each data type in Imp is associated with a *metric*.[a] This metric is our notion of distance between values of a given type. For concreteness, we fix, for the rest of the paper, the following metrics for the Imp types:

- The integer and real types are associated with the Euclidean metric $d(x, y) = |x - y|$.
- The metric over arrays (of reals or integers) of the same length is the L_∞*-norm*: $d(A_1, A_2) = \max_i\{|A_1[i] - A_2[i]|\}$. Intuitively, an array changes by ϵ when its size is kept fixed, but one or more of its items change by ϵ. We define $d(A_1, A_2) = \infty$ if A_1 and A_2 have different sizes.

The syntax of arithmetic expressions E, Boolean expressions B, and programs *Prog* is as follows:

$$
\begin{aligned}
E \quad & ::= x \mid A[i] \mid c \mid E + E \mid E \cdot E \\
B \quad & ::= E > 0 \mid B \wedge B \mid B \vee B \mid \neg B \\
\mathit{Prog} \quad & ::= \textbf{skip} \mid x := E \mid A[i] := E \mid \mathit{Prog}; \mathit{Prog} \\
& \qquad \textbf{if } B \textbf{ then } \mathit{Prog} \textbf{ else } \mathit{Prog} \mid \textbf{while } B \textbf{ do } \mathit{Prog}.
\end{aligned}
$$

Here x is a typed variable, c is a typed constant, A is an array variable, i an integer variable or constant, + and · respectively represent addition and multiplication over scalars (reals or integers), and the Boolean operators are as usual. We assume an orthogonal type system that ensures that all expressions and assignments in our programs are well-typed. The set of variables appearing in P is denoted by $Var(P) = \{x_1, \ldots, x_n\}$.

As for semantics, for simplicity, let us restrict our focus to programs that terminate on all inputs. Let *Val* be a universe of typed *values*. A *state* of P is a vector $\sigma \in Val^n$. Intuitively, for all $1 \le i \le n$, $\sigma(i)$ is the value of the variable x_i at state σ. The set of all states of P is denoted by $\Sigma(P)$.

The semantics of the program P, an arithmetic expression e occurring in P, and a Boolean expression b in P are now respectively given by maps $[\![P]\!]$: $\Sigma(P) \to \Sigma(P)$, $[\![e]\!]$: $\Sigma(P) \to Val$, and $[\![b]\!]$: $\Sigma(P) \to \{true, false\}$. Intuitively, for each state σ of P, $[\![e]\!](\sigma)$ and $[\![b]\!](\sigma)$ are respectively the values of e and b at σ, and $[\![P]\!](\sigma)$ is the state at which P terminates after starting execution from σ. We omit the inductive definitions of these maps as they are standard.

Our definition of continuity of programs is an adaptation of the traditional ϵ-δ definition of continuous functions. As a program can have multiple inputs and outputs, we define continuity with respect to an *input variable* x_i and an *output*

[a] Recall that a *metric* over a set S is a function d: $S \times S \to \mathbb{R} \cup \{\infty\}$ such that for all x, y, z, we have (1) $d(x, y) \ge 0$, with $d(x, y) = 0$ iff $x = y$; (2) $d(x, y) = d(y, x)$; and (3) $d(x, y) + d(y, z) \ge d(x, z)$.

variable x_j. Intuitively, if P is continuous with respect to input x_i and output x_j, then an arbitrarily small change to the initial value of any x_i, while keeping the remaining variables fixed, must only cause an arbitrarily small change to the final value of x_j. Variables other than x_j are allowed to change arbitrarily.

Formally, consider states σ, σ' of P and any $\epsilon > 0$. Let x_i be a variable of type τ, and let d_τ denote the metric over type τ. We say that σ and σ' are ϵ-*close* with respect to x_i, and write $\sigma \approx_{\epsilon,i} \sigma'$, if $d_\tau(\sigma(i), \sigma'(i)) < \epsilon$. We call σ' an ϵ-*perturbation* of σ with respect to x_i, and write $\sigma \equiv_{\epsilon,i} \sigma'$, if σ and σ' are ϵ-close with respect to x_i, and further, for all $j \neq i$, we have $\sigma(j) = \sigma'(j)$. Now we define:

Definition 1 (Continuity). *A program P is* continuous *at a state σ with respect to an input variable x_i and an output variable x_j if for all $\epsilon > 0$, there exists a $\delta > 0$ such that for all $\sigma' \in \Sigma(P)$,*

$$\sigma \equiv_{\delta,i} \sigma' \Rightarrow [\![P]\!](\sigma) \approx_{\epsilon,j} [\![P]\!](\sigma')$$

An issue with continuity is that it only certifies program behavior under *arbitrarily small* perturbations to the inputs. However, we may often want a definition of continuity that establishes a *quantitative* relationship between changes to a program's inputs and outputs. Of the many properties in function analysis that accomplish this, *Lipschitz continuity* is perhaps the most well known. Let $K > 0$. A function $f : \mathbb{R} \to \mathbb{R}$ is *K-Lipschitz continuous*, or simply *K-Lipschitz*, if a $\pm\epsilon$-change to x can change $f(x)$ by at most $\pm K\epsilon$. The constant K is known as the *Lipschitz constant* of f. It is easy to see that if f is K-Lipschitz for some K, then f is continuous at every input x.

We generalize this definition in two ways while adapting it to programs. First, as for continuity, we define Lipschitz continuity with respect to an input variable x_i and an output variable x_j. Second, we allow Lipschitz constants that depend on the *size of the input*. For example, suppose x_i is an array of length N, and consider ϵ-changes to it that do not change its size. We consider P to be *N-Lipschitz* with respect to x_i if on such a change, the output x_j can change at most by $N \cdot \epsilon$. In general, a Lipschitz constant of P is a function $K : \mathbb{N} \to \mathbb{R}_{\geq 0}$ that takes the size of x_i as an input.

Formally, to each value v, let us associate a *size* $\|v\| > 0$. If v is an integer or real, then $\|v\| = 1$; if v is an array of length N, then $\|v\| = N$. We have:

Definition 2 (Lipschitz continuity). *Let $K : \mathbb{N} \to \mathbb{R}_{\geq 0}$. The program P is K-*Lipschitz *with respect to an input x_i and an output x_j if for all $\sigma, \sigma' \in \Sigma(P)$ and $\epsilon > 0$,*

$$\sigma \equiv_{\epsilon,i} \sigma' \wedge (\|\sigma(i)\| = \|\sigma'(i)\|) \Rightarrow [\![P]\!](\sigma) \approx_{m,j} [\![P]\!](\sigma')$$

where $m = K(\|\sigma(i)\|) \cdot \epsilon$.

More generally, we could define Lipschitz continuity of P within a subset Σ' of its state space. In such a definition, σ and σ' in Definition 2 are constrained to be in Σ', and no assertion is made about the effect of perturbations on states outside Σ'. Such a definition is useful because many realistic programs are Lipschitz only within certain regions of their input space, but for brevity, we do not consider it here.

Now we note that many of the classic computations in computer science are in fact continuous or Lipschitz.

Example 1 (Sorting). *Let Sort be a sorting program that takes in an array A of reals and returns a sorted permutation of A. As discussed in Section 1, Sort is 1-Lipschitz in input A and output A, because any ϵ-change to the initial value of A (defined using our metric on arrays) can produce at most an ϵ-change to its final value. Note that this means that Sort is continuous in input A and output A at every program state.*

What if A was an array of integers? Continuity still holds in this case, but for more trivial reasons. Since A is of a discrete type, the only arbitrarily small *perturbation that can be made to A is no perturbation at all. Obviously, the program output does not change in this case. However, reasoning about continuity turns out to be important even under these terms. This is apparent when we try to prove that Sort is 1-Lipschitz when A is an array of integers. The easiest way to do this is to "cast" the input type of the program into an array of reals and prove that the program is 1-Lipschitz even after this modification, and this demands a proof of continuity.*

Example 2 (Shortest paths). *Let SP be a correct implementation of a shortest path algorithm. We view the graph G on which SP operates as an array of reals such that G[i] is the weight of the i-th edge. An ϵ-change to G thus amounts to a maximum change of $\pm\epsilon$ to any edge weight of G, while keeping the node and edge structure intact.*

The output of SP is the array d of shortest path distances in G—that is, d[i] is the length of the shortest path from the source node to the i-th node u_i of G. We note that SP is N-Lipschitz in input G and output d (N is the number of edges in G). This is because if each edge weight in G changes by an amount ϵ, a shortest path weight can change at most by $N \cdot \epsilon$.

On the other hand, suppose SP had a second output: an array π whose i-th element is a sequence of nodes forming a minimal-weight path between src and u_i. An ϵ-change to G may add or subtract elements from π—that is, perturb π by the amount ∞. Therefore, SP is not K-Lipschitz with respect to the output π for any K.

Example 3 (Minimum spanning trees). *Consider any algorithm MST for computing a minimum-cost spanning tree in a graph G with real edge weights. Suppose MST has a variable c that holds, on termination, the cost of the minimum spanning tree. MST is N-Lipschitz (hence continuous everywhere) in the input G and the output c.*

Example 4 (Knapsack). *Consider the Knapsack problem from combinatorial optimization. We have a set of items $\{1,\ldots, N\}$, each item i associated with a real* cost *c[i] and a real* value *v[i]. We also have a nonnegative, real-valued budget. The goal is to identify a subset Used $\subseteq \{1,\ldots, N\}$ such that the constraint $\Sigma_{j \in Used}\, c[i] \leq$ Budget is satisfied, and the value of $tot_v = \Sigma_{j \in Used}\, v[i]$ is maximized.*

Let our output variable be tot_v. As a perturbation can turn a previously feasible solution infeasible, a program Knap solving

this problem is discontinuous with respect to the input c and also with respect to the input Budget. At the same time, Knap is N-Lipschitz with respect to the input v: if the value of each item changes by $\pm\epsilon$, the value of tot_v can change by $\pm N\epsilon$.

Continuity and Lipschitz continuity can be used as definitions of *robustness*, a property that ensures that a program behaves predictably on uncertain inputs. Uncertainty, in this context, can be modeled by either nondeterminism (the value of x has a margin of error $\pm\epsilon$) or a probability distribution. For example, a robust embedded controller does not change its control decisions abruptly because of uncertainty in its sensor-derived inputs. The statistics computed by a robust scientific program are not much affected by measurement errors in the input dataset.

Robustness has benefits even in contexts where a program's relationship with uncertainty is not adversarial. The input space of a robust program does not have isolated "peaks"—that is, points where the program output is very different from outputs on close-by inputs. Therefore, we are more likely to cover a program's behaviors using random tests if the program is robust. Also, robust computations are more amenable to randomized and approximate program transformations[20] that explore trade-offs between a program's quality of results and resource consumption. Transformations of this sort can be seen to deliberately introduce uncertainty into a program's operational semantics. If the program is robust, then this extra uncertainty does not significantly affect its observable behavior, hence such a transformation is "safer" to perform. More details are available in our conference paper on robustness analysis.[3]

Now, if a program is Lipschitz, then we can give quantitative upper bounds on the change to its behavior due to uncertainty in its inputs, and further, this bound is small if the inputs are only slightly uncertain. Consequently, Lipschitz continuity is a rather strong robustness property. Continuity is a weaker definition of robustness—a program computing e^x is continuous, even though it hugely amplifies errors in its inputs. Nonetheless, it captures freedom from a common class of robustness violations: those where uncertainty in the inputs alters a program's control flow, and this change leads to a significant change in the program's output.

3. PROGRAM VERIFICATION

Now we present our automated framework for proving a program continuous[2] or Lipschitz.[3] Our analysis is *sound*—that is, a program proven continuous or Lipschitz by our analysis is indeed continuous. However, as the analysis targets a Turing-complete language, it is *incomplete*—for example, a program may be continuous even if the analysis does not deem it so.

3.1. Verifying continuity

First we show how to verify the continuity of an IMP program. We use as running examples three of the most well-known algorithms of computing: Bubble Sort, the Bellman–Ford shortest path algorithm, and Dijkstra's shortest path algorithm (Figure 2). As mentioned in Example 1, a program is always continuous in input variables of discrete types. Therefore, to make the problem more interesting, we assume that the input to Bubble Sort is an array of reals. As before, we model graphs by arrays of reals where each item represents the weight of an edge.

Given a program P, our task is to derive a syntactic *continuity judgment* for P, defined as a term $b \vdash Cont(P, In, Out)$, where b is a Boolean formula over *Var*, and *In* and *Out* are sets of variables of P. Such a judgment is read as "For each $x_i \in In$ and $x_j \in Out$ and each state σ where b is true, P is continuous in input x_i and output x_j at σ." We break down this task into the task of deriving judgments $b \vdash Cont(P', In, Out)$ for programs P' that are syntactic substructures of P. For example, if P is of the form "**if** b **then** P_1 **else** P_2," then we recursively derive continuity judgments for P_1 and P_2.

Continuity judgments are derived using a set of syntactic proof rules—the rules can be converted in a standard way into an automated program analysis that iteratively assigns continuity judgments to subprograms. Figure 1 shows the most important of our rules; for the full set, see the original reference.[2] To understand the syntax of the rules, consider the rule BASE. This rule derives a *conclusion* $b \vdash Cont(P, In, Out)$, where b, *In*, and *Out* are arbitrary, from the *premise* that P is either "**skip**" or an assignment.

The rule SEQUENCE addresses sequential composition of programs, generalizing the fact that the composition of two continuous functions is continuous. One of the premises of this rule is a *Hoare triple* of the form $\{b_1\}P\{b_2\}$. This is to be read as "For any state σ that satisfies b_1, $[\![P]\!](\sigma)$ satisfies b_2. (A standard program verifier can be used to verify this premise.) The rule IN-OUT allows us to restrict or generalize the set of input and output variables with respect to which a continuity judgment is made.

The next rule—ITE—handles conditional statements, and is perhaps the most interesting of our rules. In a conditional branch, a small perturbation to the input variables can cause control to flow along a different branch, leading to a syntactically divergent behavior. For instance, this happens in Lines 3–4 in the Bubble Sort algorithm in Figure 2—perturbations to items in A can lead to either behaviors

Figure 1. Key rules in continuity analysis.

$$\text{(BASE)}\ \frac{P \text{ is } \textbf{skip} \text{ or } x := e}{b \vdash Cont(P, In, Out)}$$

$$\text{(SEQUENCE)}\ \frac{\begin{array}{ll} b_1 \vdash Cont(P_1, In_1, Out_1) & In_2 \subseteq Out_1 \\ b_2 \vdash Cont(P_2, In_2, Out_2) & \{b_1\}P_1\{b_2\} \end{array}}{b_1 \vdash Cont(P_1; P_2, In_1, Out_2)}$$

$$\text{(IN-OUT)}\ \frac{\begin{array}{lll} b \vdash Cont(P, In, Out) & & b' \Rightarrow b \\ Out' \subseteq Out & In' \subseteq In & Var(P) \cap V = \emptyset \end{array}}{b' \vdash Cont(P, In' \cup V, Out' \cup V)}$$

$$\text{(ITE)}\ \frac{\begin{array}{l} c \vdash Cont(P_1, In, Out) \quad c \vdash Cont(P_2, In, Out) \\ c \wedge \mathcal{B}(b) \vdash (P_1 \equiv_{Out} P_2) \end{array}}{c \vdash Cont(\textbf{if } b \textbf{ then } P_1 \textbf{ else } P_2, In, Out)}$$

$$\text{(LOOP)}\ \frac{\begin{array}{ll} P \equiv \textbf{while } b \textbf{ do } R & \{c \wedge b\}R\{c\} \quad c \vdash Cont(R, X, X) \\ c \vdash Sep(P, X) & \mathcal{B}(b) \wedge c \vdash (R \equiv_X \textbf{skip}) \end{array}}{c \vdash Cont(P, X, X)}$$

Figure 2. Bubble sort, the Bellman-Ford algorithm, and Dijkstra's algorithm.

```
BUBBLESORT (A : array of reals)
 1  for j := 1 to (|A| − 1);
 2      do for i := 1 to (|A| − 1);
 3          do if (A[i] > A[i + 1])
 4              then t := A[i]; A[i] := A[i + 1]; A[i + 1] := t;

BELLMANFORD(G : array of reals, src : int)
 1  ...
 2  for i := 1 to (|G| − 1)
 3      do for each edge (v, w) of G
 4          do if d[v] + G(v, w) < d[w]
 5              then d[w] := d[v] + G(v, w)

DIJKSTRA(G : array of reals, src : int)
 1  ...
 2  while W ≠ ∅
 3      do choose edge (v, w) ∈ W such that d[w] is minimal;
 4         remove (v, w) from W;
 5         if d[w] + G[w, v] < d[v]
 6            then d[v] := d[w] + G[w, v]
```

of either "swapping $A[i]$ and $A[i + 1]$" or "leaving A unchanged."

The core idea behind the rule ITE is to show that such a divergence does not really matter, because at the program states where arbitrarily small perturbations to the program variables can "flip" the value of the guard b of the conditional statement (let us call the set of such states the *boundary* of b), the branches of the conditional are arbitrarily close in behavior.

Precisely, let us construct from b the following formula:

$$\mathcal{B}(b) = \begin{cases} \mathit{false} & \text{if } b \text{ only uses discrete variables} \\ e = 0 & \text{if } b \text{ equals } (e > 0) \text{ and } e \text{ uses at least one variable of a continuous data type} \\ \mathcal{B}(b_1) \vee \mathcal{B}(b_2) & \text{if } b \text{ equals } (b_1 \vee b_2) \text{ or } (b_1 \wedge b_2) \\ \mathcal{B}(b') & \text{if } b \text{ equals } \neg b' \end{cases}$$

Note that $\mathcal{B}(b)$ represents an overapproximation of the boundary of b. Also, for a set of output variables *Out* and a Boolean formula c, let us call programs P_1 and P_2 *Out-equivalent under* c, and write $c \vdash (P_1 \equiv_{Out} P_2)$, if for each state σ that satisfies c, the states $[\![P_1]\!](\sigma)$ and $[\![P_2]\!](\sigma)$ agree on the values of all variables in *Out*. We assume an oracle that can determine if $c \vdash (P_1 \equiv_{Out} P_2)$ for given c, P_1, P_2, and *Out*. In practice, such equivalence questions can often be solved fully automatically using modern automatic theorem provers.[19] Now, to derive a continuity judgment for a program "**if** b **then** P_1 **else** P_2" with respect to the outputs *Out*, ITE shows that P_1 and P_2 become *Out*-equivalent under the condition $\mathcal{B}(b)$.

The rule LOOP derives continuity judgments for **while**-loops. The idea here is to prove the body R of the loop continuous, then inductively argue that the entire loop is continuous too. In more detail, the rule derives a continuity judgment $c \vdash \mathit{Cont}(R, X, X)$, where c is a *loop invariant*—a property that is always true at the loop header—and X is a set of variables. Now consider any state σ satisfying c. An arbitrarily small perturbation to this state leads to an arbitrarily small change in the value of each variable in X at the end of the first iteration, which only leads to an arbitrarily small change in the value of each variable in X at the end of the second iteration, and so on. Continuity follows.

Some subtleties need to be considered, however. An execution from a slightly perturbed state may terminate earlier or later than it would in the original execution. Even if the loop body is continuous, the extra iterations in either the modified or the original execution may cause the states at the loop exit to be very different. We rule out such scenarios by asserting a premise called *synchronized termination*. A loop "**while** b **do** R" fulfills this property with respect to a loop invariant c and a set of variables X, if $\mathcal{B}(b) \wedge c \vdash R \equiv_X \mathbf{skip}$). Under this property, even if the loop reaches a state where a small perturbation can cause the loop to terminate earlier (similarly, later), the extra iterations in the original execution have no effect on the program state. We can ignore these iterations in our proof.

Second, even if the loop body is continuous in input x_i and output x_j for every $x_i, x_j \in X$, an iteration may drastically change the value of program variables not in X. If there is a data flow from these variables to some variable in X, continuity will not hold. We rule out this scenario through an extra condition. Consider executions of P whose initial states satisfy a condition c. We call a set of variables X of P *separable* under c if the value of each $z \in X$ on termination of any such execution is independent of the initial values of variables not in X. We denote the fact that X is separable in this way by $c \vdash \mathit{Sep}(P, X)$.

To verify that P is continuous in input x_i and output x_j at state σ, we derive a judgment $b \vdash \mathit{Cont}(P, \{x_i\}, \{x_j\})$, where b is true at σ. The correctness of the method follows from the following *soundness theorem*:

THEOREM 1. *If the rules in Figure 1 can derive the judgment* $b \vdash \mathit{Cont}(P, \mathit{In}, \mathit{Out})$, *then for all* $x_i \in \mathit{In}$, $x_j \in \mathit{Out}$, *and* σ *such that* $[\![b]\!] = \mathit{true}$, P *is continuous in input* x_i *and output* x_j *at* σ.

EXAMPLE 5 (WARMUP). *Consider the program* "**if** $(x > 2)$ **then** $x := x/2$ **else** $x := -5x + 11$." $\mathcal{B}(x > 2)$ *equals* $(x = 2)$ *and* $(x = 2) \vdash (x := x/2) \equiv_{\{x\}} (x := -5x + 11)$. *By* ITE, *the program is continuous in input* x *and output* x.

Let us now use our rules on the algorithms in Figure 2.

EXAMPLE 6 (BUBBLE SORT). *Consider the implementation of Bubble Sort in Figure 2. (We assume it to be rewritten as a* **while***-program in the obvious way.) Our goal is to derive the judgment* $\mathit{true} \vdash \mathit{Cont}(\mathit{BubbleSort}, \{A\}, \{A\})$.

Let $X = \{A, i, j\}$, *and let us write* $R_{\langle p,q \rangle}$ *to denote the code fragment from line* p *to line* q *(both inclusive). Also, let us write* $c \vdash \mathit{Term}(\mathbf{while}\ b\ \mathbf{do}\ R, X)$ *as an abbreviation for* $\mathcal{B}(b) \wedge c \vdash (R \equiv_X \mathbf{skip})$.

It is easy to show that $true \vdash Sep(BubbleSort, X)$ *and* $true \vdash Sep(R_{\langle 2,4\rangle}, X)$. *Each loop guard only involves discrete variables, hence we derive* $true \vdash Term(BubbleSort, X)$ *and* $true \vdash Term(R_{\langle 2,4\rangle}, X)$.

Now consider $R_{\langle 3,4\rangle}$. *As* $\mathcal{B}(A[i] > A[i+1])$ *equals* $(A[i] = A[i+1])$ *and* $(A[i] = A[i+1]) \vdash (\textbf{skip} \equiv_X R_{\langle 4,4\rangle})$, *we have* $true \vdash Cont(R_{\langle 3,4\rangle}, X, X)$, *then* $true \vdash Cont(R_{\langle 2,4\rangle}, X, X)$, *and then* $true \vdash Cont(BubbleSort, X, X)$. *Now the* In-Out *rule derives the judgment we are after.*

Example 7 (Bellman–Ford). *Take the Bellman–Ford algorithm. On termination,* $d[u]$ *contains the shortest path distance from the source node src to the node u. We want to prove that* $true \vdash Cont(BellmanFord, \{G\}, \{d\})$.

We use the symbols $R_{\langle p,q\rangle}$ *and Term as before. Clearly, we have* $true \vdash Sep(R_{\langle 3,5\rangle}, X)$ *and* $true \vdash Term(R_{\langle 3,5\rangle}, X)$, *where* $X = \{G, v, w\}$. *The two branches of the conditional in Line 4 are X-equivalent at* $\mathcal{B}(d[v] + G(v, w) < d[w])$, *hence we have* $true = \vdash Cont(R_{\langle 4,5\rangle}, X, X)$, *and from this judgment,* $true \vdash Cont(R_{\langle 3,5\rangle}, X, X)$. *Similar arguments can be made about the outer loop. Now we can derive the fact* $true \vdash Cont(BellmanFord, X, X)$; *weakening, we get the judgment we seek.*

Unfortunately, the rule Loop is not always enough for continuity proofs. Consider states σ and σ' of a continuous program P, where σ' is obtained by slightly perturbing σ. For Loop to apply, executions from σ and σ' must converge to close-by states at the end of each loop iteration. However, this need not be so. For example, think of Dijkstra's algorithm. As a shortest path computation, this program is continuous in the input graph G and the output d—the array of shortest path distances. But let us look at its main loop in Figure 2.

Note that in any iteration, there may be several items w for which $d[w]$ is minimal. But then, a slightly perturbed initial value of d may cause a loop iteration to choose a different w, leading to a drastic change in the value of d at the end of the iteration. Thus, individual iterations of this loop are not continuous, and we cannot apply Loop.

In prior work,[2] we gave a more powerful rule, called *epoch induction*, for proving the continuity of programs like the one above. The key insight here is that if we group some loop iterations together, then continuity becomes an inductive property of the groupings. For example, in Dijkstra's algorithm, a "grouping" is a maximal set S of successive loop iterations that are tied on the initial value of $d[w]$. Let σ_0 be the program state before the first iteration in S is executed. Owing to arbitrarily small perturbations to σ_0, we may execute iterations in S in a very different order. However, an iteration that ran after the iterations in S in the original execution will still run after the iterations in S. Moreover, for a fixed σ_0, the program state, once all iterations in S have been executed, is the same, no matter what order these iterations were executed in. Thus, a small perturbation cannot significantly change the state at the end of S, and the set of iterations S forms a continuous computation.

We have implemented the rules in Figure 1, as well as the epoch induction rule, in the form of a mostly-automatic program analysis. Given a program, the analysis iterates through its control points, assigning continuity judgments to subprograms until convergence is reached. Auxiliary tasks such as checking the equivalence of two straight-line program fragments (needed by rule Ite) are performed automatically using the Z3[8] SMT-solver. Human intervention is expected in two forms. First, in applications of the epoch induction rule, we sometimes expect the programmer to write annotations that define appropriate "groupings" of iterations. Second, in case a complex loop invariant is needed for the proof (e.g., when one of the programs in a program equivalence query is a nested loop), the programmer is expected to supply it. There are heuristics and auxiliary tools that can be used to automate these steps, but our current system does not employ them.

Given the incompleteness of our proof rules, a natural empirical question for us was whether our system can verify the continuity of the continuous computing tasks described in Section 2. To answer this question, we chose several 13 continuous algorithms (including algorithms) over real and real array data types. Our system was able to verify the continuity of 11 of these algorithms, including the shortest path algorithms of Bellman-Ford, Dijkstra, and Floyd-Warshall; Merge Sort and Selection Sort in addition to Bubble Sort; and the minimum spanning tree algorithms of Prim and Kruskal. Among the algorithms we could not verify were Quick Sort. Please see Chaudhuri et al.[2] for more details.

3.2. Verifying Lipschitz continuity

Now we extend the above verification framework to one for Lipschitz continuity. Let us fix variables x_i and x_j of the program P respectively as the input and the output variable. To start with, we assume that x_i and x_j are of continuous data types—reals or arrays of reals.

Let us define a *control flow path* of a program P as the sequence of assignment or **skip**-statements that P executes on some input (we omit a more formal definition). We note that since our arithmetic expressions are built from additions and multiplications, each control flow path of P encodes a continuous—in fact differentiable—function of the inputs. Now suppose we can show that each control flow path of P is a K-Lipschitz computation, for some K, in input x_i and output x_j. This does not mean that P is K-Lipschitz in this input and output: a perturbation to the initial value of x_i can cause P to execute along a different control flow path, leading to a drastically different final state. However, if P is continuous *and* the above condition holds, then P is K-Lipschitz in input x_i and output x_j.

Our analysis exploits the above observation. To prove that P is K-Lipschitz in input x_i and output x_j, we establish that (1) P is continuous at all states in input x_i and output x_j and (2) each control flow path of P is K-Lipschitz in input x_i and output x_j. Of these, the first task is accomplished using the analysis from Section 3.1. To accomplish the second task, we compute a data structure—a set of *Lipschitz matrices*—that contains upper bounds on the slopes of any computation that can be carried out in a control flow path of P.

More precisely, let P have n variables named $x_1,\ldots, x_n$, as before. A *Lipschitz matrix* J of P is an $n \times n$ matrix, each of whose elements is a function $K : \mathbb{N} \to \mathbb{R}_{\geq 0}$. Elements of J are represented either as numeric constants or as symbolic expressions (for example, $N + 5$), and the element in the i-th row and j-th column of J is denoted by $J(i, j)$. Our analysis associates P with sets $\mathcal{J}$ of such matrices via judgments $P: \mathcal{J}$. Such a judgment is read as follows: "For each control flow path C in P and each x_i, x_j, there is a $J \in \mathcal{J}$ such that C is $J(j, i)$-Lipschitz in input x_i and output x_j."

The Lipschitz matrix data structure can be seen as a generalization of the *Jacobian* from vector calculus. Recall that the Jacobian of a function $f : \mathbb{R}^n \to \mathbb{R}^n$ with inputs $x_1,\ldots, x_n \in \mathbb{R}$ and outputs $x'_1,\ldots, x'_n \in \mathbb{R}$ is the matrix whose (i,j)-th entry is $\frac{\partial x'_i}{\partial x_j}$. If f is differentiable, then for each x'_i and x_j, f is K-Lipschitz with respect to input x_j and output x'_i, where K is any upper bound on $\left|\frac{\partial x'_i}{\partial x_j}\right|$. In our setting, each control flow path represents a differentiable function, and we can verify the Lipschitz continuity of this function by propagating a Jacobian along the path. On the other hand, owing to branches, the program P may not represent a differentiable, or even continuous, function.

However, note that it is correct to associate a conditional statement "**if** b **then** P_1 **else** P_2" with the set of matrices $(\mathcal{J}_1 \cup \mathcal{J}_2)$, where the judgments $P_1 : \mathcal{J}_1$ and $P_2 : \mathcal{J}_2$ have been made inductively. Of course, this increases the number of matrices that we have to track for a subprogram. But the proliferation of such matrices can be curtailed using an approximation that *merges* two or more of them.

This merge operation $\sqcup$ is defined as $(J_1 \sqcup J_2)(i,j) = \max(J_1(i,j), J_2(i,j))$ for all J_1, J_2, i, j. Suppose we can correctly derive the judgment $P : \mathcal{J}$. Then for any $J_1, J_2 \in \mathcal{J}$, it is also correct to derive the judgment $P : (\mathcal{J} \setminus \{J_1, J_2\} \cup \{J_1 \sqcup J_2\})$. Note that this overapproximation may *overestimate* the Lipschitz constants for some of the control flow paths in P, but this is acceptable as we are not seeking the most precise Lipschitz constant for P anyway.

Figure 3 shows our rules for associating a set $\mathcal{J}$ of Lipschitz matrices with a program P. In the first rule SKIP, $\mathbf{I}$ is the identity matrix. The rule is correct because **skip** is 1-Lipschitz in input x_i and output x_i for all i, and 0-Lipschitz in input x_i and output x_j, where $i \neq j$.

To obtain Lipschitz constants for assignments to variables (rule ASSIGN), we must quantify the way the value of an arithmetic expression e changes when its inputs are changed. This is done by computing a vector ∇_e whose i-th element is an upper bound on $\left|\frac{\partial [\![e]\!]}{\partial x_i}\right|$. In more detail, we have

$$\nabla_e(i) = \begin{cases} 0 & \text{if } e \text{ is a constant} \\ 1 & \text{if } e \text{ is } x_i \text{ or } x_i[k], \text{ for some } k \\ 0 & \text{if } e \text{ is } x_j \text{ or } x_j[k], \text{ for } j \neq i \text{ and some } k \\ \nabla_{e_1}(i) + \nabla_{e_2}(i) & \text{if } e \text{ is } (e_1 + e_2) \\ \nabla_{e_1}(i) \cdot |e_2| + \nabla_{e_2}(i) \cdot |e_1| & \text{if } e \text{ is } (e_1 \cdot e_2) \text{ and one of } e_1 \text{ and } e_2 \text{ is a constant} \\ \infty & \textit{otherwise} \end{cases}$$

Assignments $x_i[m] := e$ to array locations are slightly trickier. The location $x_i[m]$ is affected by changes to variables appearing in e; if the variable x_i does not appear in e, then perturbations to the initial value of x_i have no effect on $x_i[m]$. However, the *remaining locations* in x_i are affected by, and only by, changes to the initial value of x_i. Thus, we can view x_i as being split into two "regions"—one consisting of $x_i[m]$ and the other of every other location—with possibly different Lipschitz constants. We track these constants using two different Lipschitz matrices J and J'. Here J is as in the rule ASSIGN, while J' is identical to the Lipschitz matrix for a hypothetical assignment $x_i := x_i$.

Sequential composition is handled by matrix multiplication (rule SEQUENCE)—the insight here is essentially the chain rule of differentiation. As mentioned earlier, the rule for conditional statements merges the Lipschitz matrices computed along the two branches. The WEAKEN rule allows us to overestimate a Lipschitz constant at any point.

The rule WHILE derives Lipschitz matrices for while-loops. Here $Bound^+(P, M)$ is a premise that states that the symbolic or numeric constant M is an upper bound on the number of iterations of P—it is assumed to be inferred via an auxiliary checker.[11] Finally, $\mathcal{J}^M$ is shorthand for the singleton set of matrix products $\{J_1 \ldots J_M : J_i \in \mathcal{J}\}$. In cases where M is a symbolic bound, we will not be able to compute this product explicitly. However, in many practical cases, one can reduce it to a simpler manageable form using algebraic identities.

The WHILE rule combines the rules for **if**-statements and sequential composition. Consider a loop P whose body R has Lipschitz matrix J. If the loop terminates in exactly M iterations, J^M is a correct Lipschitz matrix for it. However, if the loop may terminate after $M' < M$ iterations, we require an extra property for J^M to be a correct Lipschitz matrix: $J^i \leq J^{i+1}$ for all $i < M$. This property is ensured by the condition

Figure 3. Rules for deriving Lipschitz matrices.

$$\text{(SKIP)}\ \frac{}{\mathbf{skip} : \{\mathbf{I}\}}$$

$$\text{(ASSIGN)}\ \frac{}{(x_i := e) : \{J\}} \quad \text{Where } J(j,k) = \begin{cases} \nabla_e(k) & \text{if } j = i \\ 1 & \text{if } j = k \neq i \\ 0 & \text{otherwise} \end{cases}$$

$$\text{(ARRAY-ASSIGN)}\ \frac{}{(x_i[m] := e) : \{J, \mathbf{I}\}}\ J \text{ is as in ASSIGN}$$

$$\text{(WEAKEN)}\ \frac{P : \mathcal{J} \qquad J_1, J_2 \in \mathcal{J}}{P : \mathcal{J} \setminus \{J_1, J_2\} \cup \{J_1 \sqcup J_2\}}$$

$$\text{(SEQUENCE)}\ \frac{P_1 : \mathcal{J}_1 \qquad P_2 : \mathcal{J}_2}{(P_1 : P_2) : \{(J_2 J_1) : J_1 \in \mathcal{J}_1, J_2 \in \mathcal{J}_2\}}$$

$$\text{(ITE)}\ \frac{P_1 : \mathcal{J}_1 \qquad P_2 : \mathcal{J}_2}{\mathbf{if}\ b\ \mathbf{then}\ P_1\ \mathbf{else}\ P_2 : \mathcal{J}_1 \cup \mathcal{J}_2}$$

$$\text{(WHILE)}\ \frac{P = \mathbf{while}\ b\ \mathbf{do}\ R \quad R : \mathcal{J} \quad Bound^+(P, M) \quad \forall J \in \mathcal{J}, \forall i, j. (J(i,j) \geq 1 \vee J(i,j) = 0)}{P : \mathcal{J}^M}$$

$\forall i,j: J(i,j) = 0 \vee J(i,j) \geq 1$. Note that in the course of a proof, we can weaken *any* Lipschitz matrix for the loop body to a matrix J of this form.

We can prove the following soundness theorem:

THEOREM 2. *Let P be continuous in input x_i and output x_j. If the rules in Figure 3 derive the judgment $P : \{J\}$, then P is $J(j, i)$-Lipschitz in input x_i and output x_j.*

EXAMPLE 8 (WARMUP). *Recall the program "***if** $(x > 2)$ **then** $x := x/2$ **else** $x := -5x + 11$*" from Example 5 (x is a real). Our rules can associate the left branch with a single Lipschitz matrix containing a single entry $\frac{1}{2}$, and the right branch with a single matrix containing a single entry 5. Given the continuity of the program, we conclude that the program is 5-Lipschitz in input x and output x.*

EXAMPLE 9 (BUBBLE SORT). *Consider the Bubble Sort algorithm (Figure 2) once again, and as before, let $R_{\langle p,q\rangle}$ denote the code fragment from line p to line q. Let us set x_0 to be A and x_1 to be t.*

Now, let $J = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$. From the rules in Figure 3, we can derive $(t := A[i]) : \{J\}$, $(A[i] := A[i+1]) : \{\mathbf{I}\}$, and $(A[i+1] := t) : \{J, \mathbf{I}\}$.

Carrying out the requisite matrix multiplications, we get $R_{\langle 4,4\rangle} : \{J\}$. Using the rule ITE*, we have $R_{\langle 3,4\rangle} : \{\mathbf{I}, J\}$. Now, it is easy to show that $R_{\langle 3,4\rangle}$ gets executed N times, where N is the size of A. From this we have $R_{\langle 2,4\rangle} : \{\mathbf{I}, J\}^N$. Given that $J^2 = \mathbf{I}J = J\mathbf{I} = J$, this is equivalent to the judgment $R_{\langle 2,4\rangle} : \{\mathbf{I}, J\}$. From this, we derive $BubbleSort : \{J, \mathbf{I}\}$. Given the proof of continuity carried out in Example 1, Bubble Sort is 1-Lipschitz in input A and output A.*

Intuitively, the reason why Bubble Sort is so robust is that here, (1) there is no data flow from program points where arithmetic operations are carried out to points where values are assigned to the output variable and (2) continuity holds everywhere. In fact, one can formally prove that any program that meets the above two criteria is 1-Lipschitz. However, we do not develop this argument here.

EXAMPLE 10 (BELLMAN–FORD; DIJKSTRA). *Let us consider the Bellman–Ford algorithm (Figure 2) once again, and let x_0 be G and x_1 be d. Consider line 5 (i.e., the program $R_{\langle 5,5\rangle}$); our rules can assign to this program the Lipschitz matrix J, where $J = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. With a few more derivations, we obtain $R_{\langle 4,5\rangle} : \{J\}$. Using the rule for loops, we have $R_{\langle 3,5\rangle} : \{J^N\}$, where N is the number of edges in G, and then $BellmanFord : \{J^{N^2}\}$. But note that*

$$J^{N^2} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{N^2} = \begin{pmatrix} 1 & 0 \\ N^2 & 1 \end{pmatrix}.$$

Combing the above with the continuity proof in Example 7, we decide that the Bellman–Ford algorithm is N^2-Lipschitz in input G and output d.

Note that the Lipschitz constant obtained in the above proof is not the optimal one—that would be N. This is an instance of the gap between truth and provability that is the norm in program analysis. Interestingly, our rules can derive the optimal Lipschitz constant for Dijkstra's algorithm. Using the same reasoning as above, we assign to the main loop of the algorithm the single Lipschitz matrix J. Applying the LOOP *rule, we derive*

$$Dijkstra : \left\{ \begin{pmatrix} 1 & 0 \\ N & 1 \end{pmatrix} \right\}.$$

Given that the algorithm is continuous in input G and output d, it is N-Lipschitz in input G and output d.

Let us now briefly consider the case when the input and output variables in our program are of discrete type. As a program is trivially continuous in every discrete input, continuity is not a meaningful notion in such a setting. Therefore, we focus on the problem of verifying Lipschitz continuity—for example, showing that the Bubble Sort algorithm is 1-Lipschitz even when the input array A is an array of integers.

An easy solution to this problem is to *cast* the array A into an array $A^\star$ of reals, and then to prove 1-Lipschitz continuity of the resultant program in input $A^\star$ and output $A^\star$. As any integer is also a real, the above implies that the original algorithm is 1-Lipschitz in input A and output A. Thus, reals are used here as an *abstraction* of integers, just as (unbounded) integers are often used in program verification as abstractions of bounded-length machine numbers.

Unsurprisingly, this strategy does not always work. Consider the program "**if** $(x > 0)$ **then** $x := x + 1$ **else skip**," where x is an integer. This program is 2-Lipschitz. Its "slope" is the highest around initial states where $x = 0$: if the initial value of x changes from 0 to 1, the final value of x changes from 0 to 2. At the same time, if we cast x into a real, the resultant program is discontinuous and thus not K-Lipschitz for any K.

It is possible to give an analysis of Lipschitz continuity that does not suffer from the above issue. This analysis casts the integers into reals as mentioned above, then calculates a Lipschitz matrix of the resultant program; however, it checks a property that is slightly weaker than continuity. For lack of space, we do not go into the details of the analysis here.

We have extended our implementation of continuity analysis with the verification method for Lipschitz continuity presented above, and applied the resulting system to the suite of 13 algorithms mentioned at the end of Section 3.1. All these algorithms were either 1-Lipschitz or N-Lipschitz. Our system was able to compute the optimal Lipschitz constant for 9 of the 11 algorithms where continuity could be verified. In one case (Bellman-Ford), it certified an N-Lipschitz computation as N^2-Lipschitz. The one example on which it fared poorly was the Floyd-Warshall shortest path algorithm, where the best Lipschitz constant that it could compute was exponential in N^3.

4. RELATED WORK

So far as we know, we were the first[2] to propose a framework for continuity analysis of programs. Before us, Hamlet[12] advocated notions of continuity of software; however, he concluded that "it is not possible in practice to mechanically test for continuity" in the presence of

loops. Soon after our first paper on this topic (and before our subsequent work on Lipschitz continuity of programs), Reed and Pierce[18] gave a type system that can verify the Lipschitz continuity of functional programs. This system can seamlessly handle functional data structures such as lists and maps; however, unlike our method, it cannot reason about discontinuous control flow, and would consider any program with a conditional branch to have a Lipschitz constant of ∞.

More recently, Jha and Raskhodnikova have taken a *property testing* approach to estimating the Lipschitz constant of a program. Given a program, this method determines, with a level of probabilistic certainty, whether it is either 1-Lipschitz or ϵ-far (defined in a suitable way) from being 1-Lipschitz. While the class of programs allowed by the method is significantly more restricted than what is investigated here or by Reed and Pierce [13], the appeal of the method lies in its crisp completeness guarantees, and also in that it only requires blackbox access to the program.

Robustness is a standard correctness property in control theory,[16, 17] and there is an entire subfield of control studying the design and analysis of robust controllers. However, the systems studied by this literature are abstractly defined using differential equations and hybrid automata rather than programs. The systematic modeling and analysis of robustness of *programs* was first proposed by us in the context of general software, and by Majumdar and Saha[14] in the context of control software.

In addition, there are many efforts in the abstract interpretation literature that, while not verifying continuity or robustness explicitly, reason about the uncertainty in a program's behavior due to floating-point rounding and sensor errors.[6, 7, 10] Other related literature includes work on automatic differentiation (AD),[1] where the goal is to transform a program P into a program that returns the derivative of P where it exists. Unlike the work described here, AD does not attempt verification—no attempt is made to certify a program as differentiable or Lipschitz.

5. CONCLUSION

In this paper, we have argued for the adoption of analytical properties like continuity and Lipschitz continuity as correctness properties of programs. These properties are relevant as they can serve as useful definitions of *robustness* of programs to uncertainty. Also, they raise some fascinating technical issues. Perhaps counterintuitively, some of the classic algorithms of computer science satisfy continuity or Lipschitz continuity, and the problem of systematic reasoning about these properties demands a nontrivial combination of analytical and logical insights.

We believe that the work described here is a first step toward an extension of the classical calculus to a symbolic mathematics where programs form a first-class representation of functions and dynamical systems. From a practical perspective, this is important as physical systems are increasingly controlled by software, and as even applied mathematicians increasingly reason about functions that are not written in the mathematical notation of textbooks, but as *code*. Speaking more philosophically, the classical calculus focuses on the computational aspects of real analysis, and the notation of calculus texts has evolved primarily to facilitate symbolic computation by hand. However, in our era, most mathematical computations are carried out by computers, and a calculus for our age should not ignore the notation that computers can process most easily: programs. This statement has serious implications—it opens the door not only to the study of continuity or derivatives but also to, say, Fourier transforms, differential equations, and mathematical optimization of code. Some efforts in these directions[4, 5, 9] are already under way; others will no doubt appear in the years to come.

Acknowledgments

This research was supported by NSF CAREER Award #1156059 ("Robustness Analysis of Uncertain Programs: Theory, Algorithms, and Tools").

References

1. Bucker, M., Corliss, G., Hovland, P., Naumann, U., Norris, B. Automatic Differentiation: Applications, Theory and Implementations, Birkhauser, 2006.
2. Chaudhuri, S., Gulwani, S., Lublinerman, R. Continuity analysis of programs. In *POPL* (2010), 57–70.
3. Chaudhuri, S., Gulwani, S., Lublinerman, R., Navidpour, S. Proving programs robust. In *FSE* (2011), 102–112.
4. Chaudhuri, S., Solar-Lezama, A. Smooth interpretation. In *PLDI* (2010), 279–291.
5. Chaudhuri, S., Solar-Lezama, A. Smoothing a program soundly and robustly. In *CAV* (2011), 277–292.
6. Chen, L., Miné, A., Wang, J., Cousot, P. Interval polyhedra: An abstract domain to infer interval linear relationships. In *SAS* (2009), 309–325.
7. Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., Rival, X. The ASTREÉ analyzer. In *ESOP* (2005), 21–30.
8. de Moura, L. M. Bjørner, N. Z3: An effcient smt solver. In *TACAS* (2008), 337–340.
9. Girard, A., Pappas, G. Approximate bisimulation: A bridge between computer science and control theory. *Eur. J. Contr. 17*, 5 (2011), 568.
10. Goubault, E. Static analyses of the precision of floating-point operations. In *SAS* (2001).
11. Gulwani, S., Zuleger, F. The reachability-bound problem. In *PLDI* (2010), 292–304.
12. Hamlet, D. Continuity in software systems. In *ISSTA* (2002).
13. Jha, M., Raskhodnikova, S. Testing and reconstruction of lipschitz functions with applications to data privacy. In *FOCS* (2011), 433–442.
14. Majumdar, R., Saha, I. Symbolic robustness analysis. In *RTSS* (2009), 355–363.
15. Parnas, D. Software aspects of strategic defense systems. *Commun. ACM 28*, 12 (1985), 1326–1335.
16. Pettersson, S., Lennartson, B. Stability and robustness for hybrid systems. In *Decision and Control* (Dec 1996), 1202–1207.
17. Podelski, A., Wagner, S. Model checking of hybrid systems: From reachability towards stability. In *HSCC* (2006), 507–521.
18. Reed, J., Pierce, B. Distance makes the types grow stronger: A calculus for differential privacy. In *ICFP* (2010).
19. Strichman, O. Regression verification: Proving the equivalence of similar programs. In *CAV* (2009).
20. Zhu, Z., Misailovic, S., Kelner, J., Rinard, M. Randomized accuracy-aware program transformations for efficient approximate computations. In *POPL* (2012).

Swarat Chaudhuri (swarat@rice.edu), Department of Computer Science, Rice University, Houston, TX.

Sumit Gulwani (sumitg@microsoft.com), Microsoft Research, Redmond, WA.

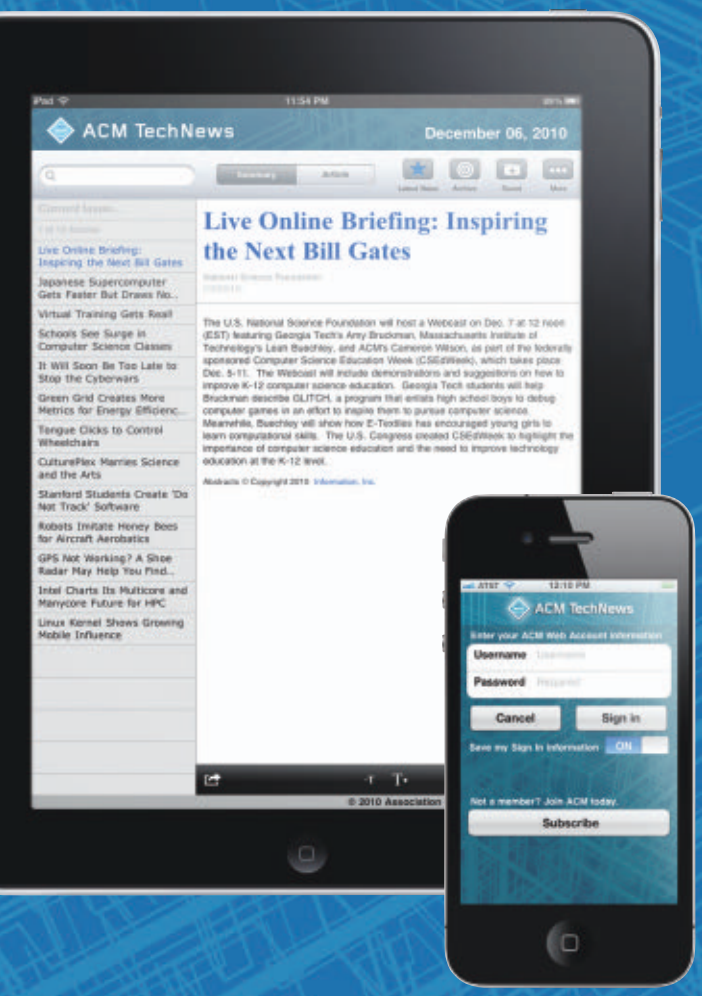
Roberto Lublinerman (rluble@psu.edu), Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA.

ACM TechNews Goes Mobile

iPhone & iPad Apps Now Available in the iTunes Store

ACM TechNews—ACM's popular thrice-weekly news briefing service—is now available as an easy to use mobile apps downloadable from the Apple iTunes Store.

These new apps allow nearly 100,000 ACM members to keep current with news, trends, and timely information impacting the global IT and Computing communities each day.



TechNews mobile app users will enjoy:

- ***Latest News:*** Concise summaries of the most relevant news impacting the computing world
- ***Original Sources:*** Links to the full-length articles published in over 3,000 news sources
- ***Archive access:*** Access to the complete archive of TechNews issues dating back to the first issue published in December 1999
- ***Article Sharing:*** The ability to share news with friends and colleagues via email, text messaging, and popular social networking sites
- ***Touch Screen Navigation:*** Find news articles quickly and easily with a streamlined, fingertip scroll bar
- ***Search:*** Simple search the entire TechNews archive by keyword, author, or title
- ***Save:*** One-click saving of latest news or archived summaries in a personal binder for easy access
- ***Automatic Updates:*** By entering and saving your ACM Web Account login information, the apps will automatically update with the latest issues of TechNews published every Monday, Wednesday, and Friday

The Apps are freely available to download from the Apple iTunes Store, but users must be registered individual members of ACM with valid Web Accounts to receive regularly updated content.

http://www.apple.com/iphone/apps-for-iphone/ **http://www.apple.com/ipad/apps-for-ipad/**

ACM TechNews

Princeton University
Computer Science
Lecturer

The Department of Computer Science seeks applications from outstanding teachers to assist the faculty in teaching our introductory course sequence or some of our upper-level courses starting September 1, 2012. Depending on the qualifications and interests of the applicant, the job responsibilities will include such activities as teaching recitation sections and supervising graduate-student teaching assistants; grading problem sets and programming assignments, and supervising students in the grading of problem sets and programming assignments; developing and maintaining online curricular material, classroom demonstrations, and laboratory exercises; and supervising undergraduate research projects. An advanced degree in computer science, or related field, is required (PhD preferred). The position is renewable for 1-year terms, up to six years, depending upon departmental need.

Princeton University is an equal opportunity employer and complies with applicable EEO and affirmative action regulations. You may apply online, by submitting a letter of application, resume and names of three references at http://jobs.cs.princeton.edu/lecturer. Requisition number: 1200313

U.S. Naval Academy
Computer Science Department
Assistant Professor

The U.S. Naval Academy's Computer Science Department invites applications for one or more tenure track positions. Appointments at all ranks will be considered, but preference is for junior faculty at the rank of Assistant Professor. These positions may begin as early as the Fall of 2012. A Ph.D. in Computer Science or closely related field is required.

Applicants with backgrounds, experience and research interests in cyber security, and information assurance are especially encouraged to apply, however all backgrounds of computer science will be considered. Applicants must have a dedication to teaching, an ability to teach a broad range of computer science courses, and the ability to initiate and maintain a strong research program.

The Computer Science Department offers majors in Computer Science and Information Technology, and is developing a new major in Cyber Security. We currently have 85 CS majors, 100 IT majors and a faculty of 21. The department is housed in a state of the art building overlooking the scenic Severn River, and discussions have begun regarding a new academic building to support the department, including the new cyber security program. Our spaces provide outstanding office, laboratory, and research facilities for both students and faculty, including specialized labs for information assurance, networking, and robotics, as well as three micro-computing labs and two high performance computing labs. In addition to computer science and information technology courses for the majors, we also teach a required course on cyber security to the entire freshman class.

The Naval Academy is an undergraduate institution located in historic downtown Annapolis, Maryland on the Chesapeake Bay. Roughly half the faculty are tenured or tenure track civilian professors with Ph.D.s who balance teaching excellence with internationally recognized research programs. The remaining faculty are active duty military officers with Masters or Doctoral degrees. Each year the academy graduates roughly 1000 undergraduate students with majors in the sciences, engineering, and humanities. More information about the department and the Academy can be found at http://www.usna.edu/cs/ and http://www.usna.edu/.

Applicants should send a cover letter, teaching and research statements, curriculum vitae, and arrange for three letters of recommendation that address both teaching and research abilities to be sent to cssearch@usna.edu.

Review of applications will begin immediately, continuing until the positions are filled.

The United States Naval Academy is an Equal Opportunity Employer. This agency provides reasonable accommodations to applicants with disabilities. This position is subject to the availability of funds.

University of Calgary
Department of Computer Science
Assistant Professor Position

The **Department of Computer Science** at the University of Calgary seeks outstanding candidates for a tenure-track position at the Assistant Professor level. Applicants from the area of Database Management are of primary interest. Details for this position appear at: http://www.cpsc.ucalgary.ca/. Applicants must possess a doctorate in Computer Science at the time of appointment, and have a strong potential to develop an excellent research record.

The Department is one of Canada's leaders as evidenced by our commitment to excellence in research and teaching. It has large undergraduate and graduate programs and extensive state-of-the-art computing facilities. Calgary is a multicultural city that is the fastest growing city in Canada. Calgary enjoys a moderate climate located beside the natural beauty of the Rocky Mountains.

Further information about the Department is available at http://www.cpsc.ucalgary.ca/.

Interested applicants should send a CV, a concise description of their research area and program, a statement of teaching philosophy, and arrange to have at least three reference letters sent to: **Dr. Carey Williamson, Head,** Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, T2N 1N4 or via email to: search@cpsc.ucalgary.ca.

Completed applications received by October 15, 2012 will receive full consideration, though the review process will continue until the position is filled. Hiring decisions will be finalized as soon as possible, with the successful candidate joining the U of C on either January 1, 2013 or July 1, 2013.

All qualified candidates are encouraged to apply; however, Canadians and permanent residents will be given priority.

University of Oregon
Department of Computer and Information Science
Professor/Department Head

The Computer and Information Science (CIS) Department at the University of Oregon invites applications for Professor/Department Head. We seek an outstanding scholar who will be excited to head a computer science department in a strong public research university. A number of senior faculty have recently or will soon retire, creating an opportunity for regeneration through multiple hires over the next several years. The department is currently developing and implementing a strategic plan for increased research prominence in the context of a college-wide plan for excellence.

We seek an individual with strategic vision and leadership abilities. The ideal candidate will be a prominent scholar with a sustained record of publication and research funding in an area of software or intelligent systems that compliments existing research strengths of the department. We are looking for an innovative thinker who is eager to advance interdisciplinary scholarship, build bridges between academia and industry, mentor faculty to achieve excellence, teach at the undergraduate and graduate levels, and communicate the intellectual excitement of computer science and its broad, evolving role in contemporary society. A PhD in Computer Science or a closely related field is required.

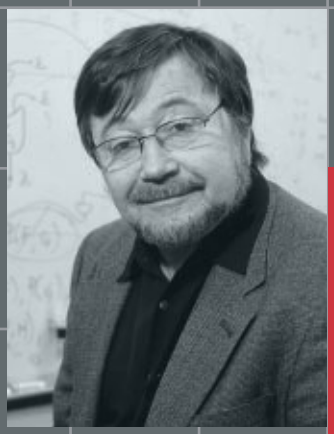
The University of Oregon is an AAU comprehensive research university with many nationally and internationally renowned programs. It is located in Eugene, two hours south of Portland, and within one hour's drive of both the Pacific Ocean and the snow-capped Cascade Mountains. The CIS Department is part of the College of Arts and Sciences and is housed within the integrated Lorry Lokey Science Complex. The department offers B.S., M.S. and Ph.D. degrees. More information about the department, its programs and faculty can be found at http://www.cs.uoregon.edu, or by contacting the search committee at faculty.search@cs.uoregon.edu.

Applications will be accepted electronically through the department's web site (only). Application information can be found at http://www.cs.uoregon.edu/Employment/. Review of applications has been extended through October 1, 2012 and will continue until the position is filled. Please address questions to faculty.search@cs.uoregon.edu.

The University of Oregon is an equal opportunity/affirmative action institution committed to cultural diversity and is compliant with the Americans with Disabilities Act. We are committed to creating a more inclusive and diverse institution and seek candidates with demonstrated potential to contribute positively to its diverse community.

THE ACM A. M. TURING AWARD

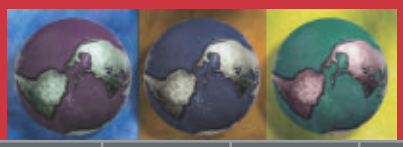
by the community ◆ from the community ◆ for the community



ACM, Intel, and Google congratulate

JUDEA PEARL

for fundamental contributions to artificial intelligence through the development of a calculus for probabilistic and causal reasoning.



"Dr. Pearl's work provided the original paradigm case for how to do statistical AI. By placing structured knowledge representations at the heart of his work, and emphasizing how these representations enabled efficient inference and learning, he showed the field of AI how to build statistical reasoning systems that were actually telling us something about intelligence, not just statistics."

Limor Fix
Director, University Collaborative Research Group
Intel Labs

For more information see www.intel.com/research.

"Judea Pearl is the most prominent advocate for probabilistic models in artificial intelligence. He developed mathematical tools to tackle complex problems that handle uncertainty. Before Pearl, AI systems had more success in black and white domains like chess. But robotics, self-driving cars, and speech recognition deal with uncertainty. Pearl enabled these applications to flourish, and convinced the field to adopt these techniques."

Alfred Spector
Vice President, Research and Special Initiatives
Google Inc.

For more information, see http://www.google.com/corporate/index.html and http://research.google.com/.

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

DOI:10.1145/2240236.2240263

Peter Winkler

Puzzled
Find the Magic Set

Welcome to three new puzzles. Each involves a collection of items, and your job is to find a subset of them that is characterized by a particular property. Since solving the puzzles is not easy, here are a couple of hints: For the first, think about averages; for the other two, try constructing your sets sequentially, bearing in mind that if two partial sums are equal, the terms between them must add up to zero.

1. A balance scale sits on a teacher's desk, currently tipped to the right. A set of weights is on the scales, and on each weight is the name of at least one student. Class is about to begin, and on entering the classroom, each student moves each weight carrying his or her name to the opposite side of the scale. Now show there is a set of pupils that you, the teacher, can let in the classroom that will tip the scales to the left.

2. You have two sets (blue and red) of n n-sided dice, with each die labeled with the numbers 1 to n. You roll all $2n$ dice simultaneously. Now find a nonempty subset of the red dice and a nonempty subset of the blue dice with the same sum.

3. You begin with a list of all 1,024 possible vectors of length 10 with entries +1 or −1. Your crayon-wielding three-year-old child has got hold of the list and unfortunately changed some of the entries in some of the vectors to zeroes. Now find a non-empty subset of the altered vectors that add up to the all-zero vector (0,0,0,0,0,0,0,0,0,0).

Readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is William Morrill Professor of Mathematics and Computer Science at Dartmouth College, Hanover, NH.

THIRD ANNUAL ACM CONFERENCE ON

SYSTEMS, PROGRAMMING, LANGUAGES, APPLICATIONS: SOFTWARE FOR HUMANITY

SPLASH

TUCSON, ARIZONA

OCTOBER 19-26, 2012

LOCATION
Loews Ventana Canyon Resort
Tucson, Arizona, USA

EVENTS
- OOPSLA
- Onward!
- Dynamic Languages Symposium,
- Wavefront
- Workshops
- ACM Student Research Competition
- Pattern Languages of Programming
- ...and more!

GENERAL CHAIR
Gary T. Leavens
University of Central Florida
chair@splashcon.org

OOPSLA PROGRAM CHAIR
Matthew B. Dwyer
University of Nebraska
oopsla@splashcon.org

ONWARD! PROGRAM CHAIRS
Jonathan Edwards/Julia Steele
MIT/O'Reilly
onward@splashcon.org
essays@splashcon.org

DLS PROGRAM CHAIR
Alessandro Warth
Google
dls@splashcon.org

FOR MORE INFORMATION
info@splashcon.org
http://splashcon.org/

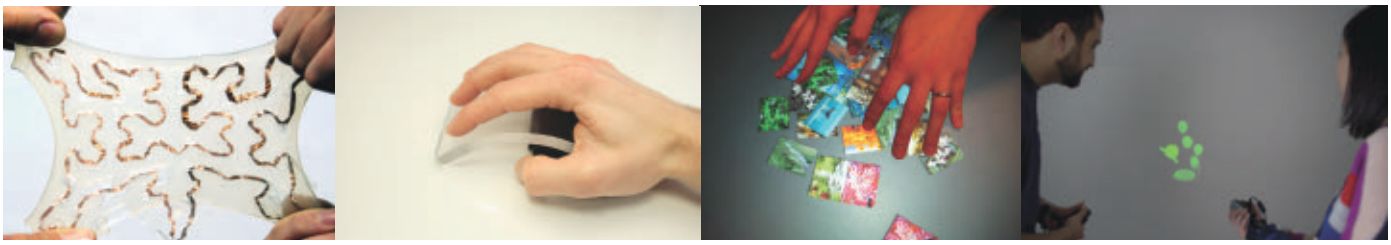
SPLASH is sponsored by
ACM SIGPLAN

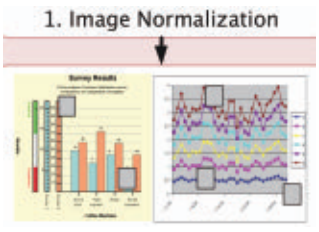
UIST2012

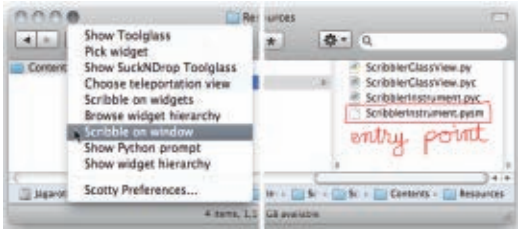
25th ACM SYMPOSIUM ON USER INTERFACE SOFTWARE & TECHNOLOGY

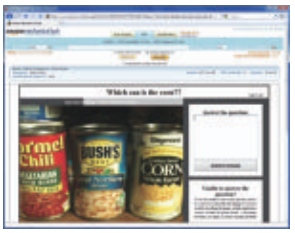
OCTOBER 7th - 12th, 2012
CAMBRIDGE, MASSACHUSETTS

GRAPHICAL + WEB USER INTERFACES
FOR INNOVATIVE APPLICATIONS
TANGIBLE + UBIQUITOUS COMPUTING
VIRTUAL + AUGMENTED REALITY
NEW INPUT + OUTPUT DEVICES
COMPUTER SUPPORTED COOPERATIVE WORK









http://acm.org/uist