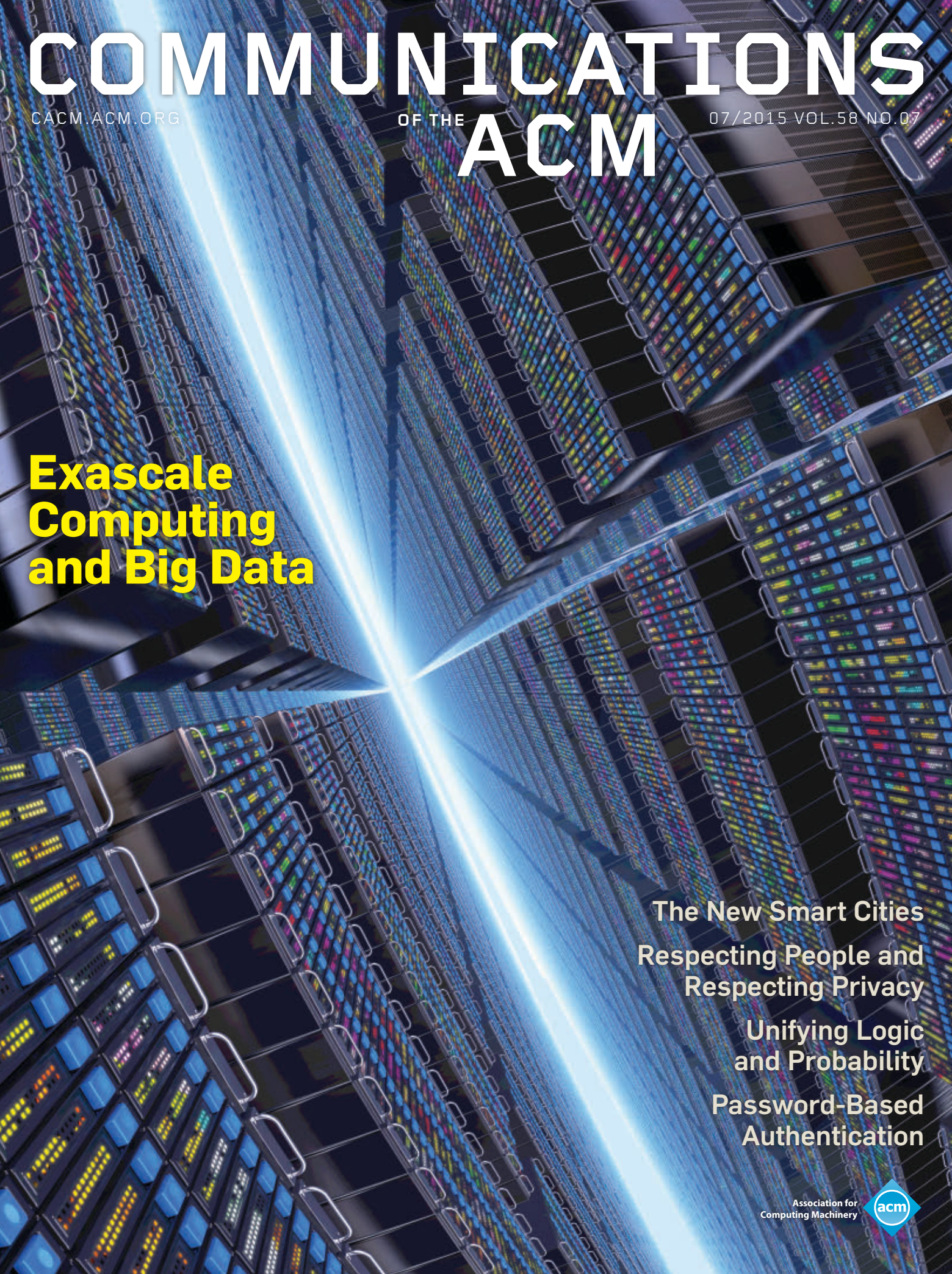
COMMUNICATIONS OF THE ACM
CACM.ACM.ORG
07/2015 VOL.58 NO.07



Exascale Computing and Big Data

The New Smart Cities

Respecting People and Respecting Privacy

Unifying Logic and Probability

Password-Based Authentication

Association for Computing Machinery

KDD2015

21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining

10 - 13 August 2015, Hilton, Sydney

http://www.kdd.org/kdd2015/

KDD 2015 is a premier conference that brings together researchers and practitioners from data mining, knowledge discovery, data analytics, and big data. KDD 2015 will be the first Australian edition of KDD, and is its second time in the Asia Pacific region.

- 4 Keynotes
- 8 Invited Industry Talks
- 14 Workshops
- 12 Tutorials
- Industry and Government Track
- Research Track
- Big Data Summit (collocated)
- Regional Forums
- Mentoring Program
- 27 Exhibition Booths

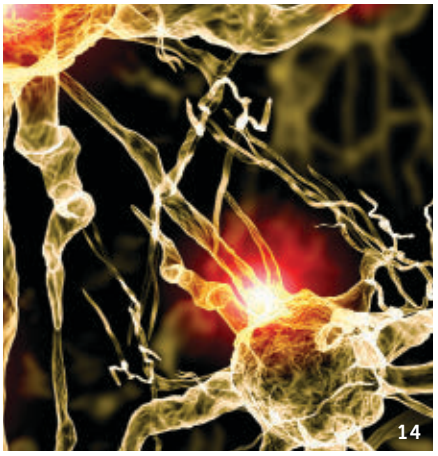
For enquiries, contact Prof. Longbing Cao at general2015@kdd.org.

COMMUNICATIONS OF THE ACM

Departments

Last Byte

News



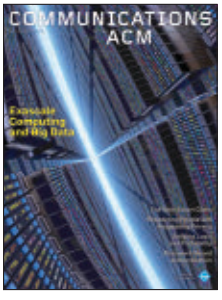
Viewpoints

Association for Computing Machinery
Advancing Computing as a Science & Profession

IMAGE BY SERGEY NIVENS

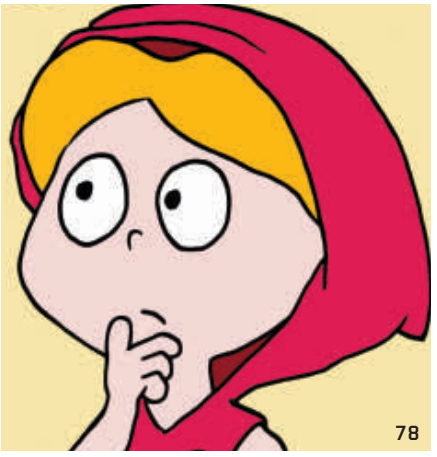
Practice

Articles' development led by acmqueue
queue.acm.org



About the Cover: Daniel Reed and Jack Dongarra explore the myriad of great opportunities and greater challenges that face the future of exascale computing and big data analysis (p. 56). Cover illustration by Peter Bollinger.

Contributed Articles



Watch the authors discuss their work in this exclusive *Communications* video. http://cacm.acm.org/videos/exascale-computing-and-big-data

Review Articles

Watch the author discuss their work in this exclusive *Communications* video. http://cacm.acm.org/videos/unifying-logic-and-probability

Research Highlights

IMAGES BY MARIO WAGNER. IWONA USAKIEWICZ/ANDRIJ BORYS ASSOCIATES. ALMAGAMI

> **Bobby Schnabel's contributions, to the computing community in general and ACM in particular, have been nothing short of stellar.**

DOI:10.1145/2788395

Alexander L. Wolf

A New Chief Executive Officer and Executive Director of ACM

I SOMETIMES REFLECT on what makes ACM so special, so unique as a professional society. I can easily come up with several good reasons, including the extent to which responsibility, authority, and budget control are devolved to our 37 special interest groups or the strong support we provide to our fellow volunteers in building a broad range of community service activities (the "good works" of ACM). But alongside or, perhaps, underlying these very visible special and unique features is the largely invisible special and unique relationship that exists between volunteer leaders and ACM headquarters staff. This is a relationship characterized by mutual respect and trust, and a deep understanding of each other's motivations, aspirations, and priorities. Not that there are no conflicts and sometimes even some tensions, but as in any healthy relationship, it is the manner in which you approach and overcome problems and challenges that matters, not the fact they may exist.

What is the secret to ACM's incredible success in building its volunteer/staff relationship? From where did it come and how do we ensure it continues? This is something that has been literally decades in the making (starting well before my time as an ACM volunteer) and something that as President I feel a great responsibility to preserve and nurture.

From my point of view, a watershed moment in the building of that relationship occurred in November 1998 when John White—holder of a Ph.D. in computer science, a former university professor, an industrial research lab director, and a past president of ACM, transitioned from his role as a long-standing volunteer to that of chief executive officer (the first person to hold that title) and executive director (the traditional title of someone who heads the staff of a non-profit organization). For the first time, the most senior member of staff would be someone from our technical community. What John did with that opportunity is, simply put, legendary. While previous ACM EDs contributed fundamentally and substantially to our success, John brought a different kind of experience and perspective to the position, one that in the context of computing's explosive growth in importance and ubiquity (John took up his position in the same year that Microsoft Windows 98 was released and Google was founded) could not have been more timely for our profession. Together with ACM COO Pat Ryan and the rest of the headquarters staff, John worked tirelessly with ACM volunteers to guide our association on a path toward its standing today as the world's leading international professional society of computing engineers, scientists, educators, and students.

With John's impending retirement after nearly 17 years, it fell to me (much to the relief of some recent past presidents!) to lead the process of finding a CEO who could ably sustain and even grow ACM's success, but more fundamentally perhaps, continue the excel-

lent working relationship between volunteers and staff. From the outset we aimed, once again, to attract someone from our community: someone with the right combination of experience leading a major organization, a history of deep knowledge of ACM, and an acknowledged technical standing in computing. I called upon a diverse set of highly respected computing professionals to serve on a search committee: Vicki Hanson (chair), Muffy Calder, Vint Cerf, Stu Feldman, Mary Jane Irwin, Matthias Kaiserswerth, and Peter Lee. I asked them to come back with a recommendation to the ACM Executive Committee, the leadership group charged in our bylaws with making the appointment decision. After receiving a remarkable number of truly outstanding applications and carrying out a thorough review and interview process, the committee forwarded a name that was unanimously accepted by the ACM EC, with the unanimous concurrence of the full ACM Council.

I am extremely pleased to welcome Robert (Bobby) Schnabel as the new CEO and ED of ACM. Bobby will take up his position on November 1, and, in the interim, I have named Pat Ryan acting ED. Bobby is without question an ideal person to serve in this role going forward. He comes to us from Indiana University, where he is Professor and Dean in the School of Informatics and Computing, leading the school through a remarkable period of growth in size, stature, and funding. Prior to this he spent 30 years at the University of Colorado at Boulder after receiving a Ph.D. in computer science from Cornell University. Among many other senior duties and positions in Boulder, Bobby founded and directed the Alliance for Technology, Learning and Society (ATLAS) Institute and served for nine years as campus CIO. Bobby's technical background is in numerical computation, having published over 100 scholarly articles in the area, earning him advancement to Fellow of both SIAM and the ACM.

Bobby's contributions, to the computing community in general and ACM in particular, have been nothing short of stellar. Early on he served as chair of an ACM special interest group (SIGNUM) and later editor-in-chief of the highly respected *SIAM Review*. He will step down as founding chair of the ACM Education Policy Committee, which under his leadership has played a pivotal role in convincing policymakers of the importance of an early and substantial computing education. Bobby was a co-founder of the National Center for Women & Information Technology (NCWIT) and was a board member of the Computer Research Association (CRA). Bobby is currently a board member of code.org, serves as chair of the advisory board of the Alliance for Hispanic Serving Institutions, and is a member of the U.S. National Science Foundation Computing and Information Science and Engineering advisory committee. These are but some of the many important activities that have seen him take a leadership role in our community, building a broad and secure foundation for his challenging new duties at ACM.

Although I am sad to see John leave, I am extremely excited to have Bobby take up the role at ACM. Bobby brings with him a special passion and bright new ideas that will help continue the great tradition of leadership, innovation, openness, and growth that have marked the history of ACM. C

Alexander L. Wolf is president of ACM and a professor in the Department of Computing at Imperial College London, U.K.

PHOTOGRAPH BY DAVID KATZIVE

Bobby Schnabel assumes the role of ACM's CEO and Executive Director on Nov. 1, 2015.

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

Printed in the U.S.A.

Association for Computing Machinery

DOI:10.1145/2786922

Vinton G. Cerf

Milestones

Last month I wrote about how much we owe to John White, not only for his 17 years of service to ACM as CEO, but also his earlier services in various ACM volunteer positions, including president. Now I want to draw attention to the fact that ACM will be 70 years old in 2017. It is not too early to being thinking about how we might usefully recognize this milestone. We had a remarkable celebration we called ACM 1 in 1997 as we reached the 50th year of our existence and while 70 is not the same as, say, 100, it is a significant moment and worthy of some serious thought. Such moments encourage all of us to think about what we want to achieve in the future, even as we look back on the many years of our existence and accomplishments.

One immediate thought is to try to compose a list of, say, 25 problems in computer science that remain unsolved—rather like David Hilbert's 23 problems in mathematics published in 1900.[a] What might be on our list? I am not sure how to formulate this, but one thing on my list is to ask whether we can ever get to the point where we can write bug-free code in a reliable fashion! There are other questions having to do with computability and computer design. Will new machine architectures permit some form of computation that goes beyond the reach of the Turing Machine? Are there limits to quantum computation? Scott Aaronson tackles that question in a 2008 *Scientific American* article.[b] What do we not know about the fundamentals of computation and programming, in their largest sense, that we should seek to understand? What other questions would you put on such a list?

Another thing we might consider is how we think computing (and communications) will change over the next 70 years. The Internet, the World Wide Web and the Internet of Things are already a part of daily life for about 40% of the world's population and smartphones for an even larger fraction. Can we try to imagine what the next major changes might be? Will nano devices, programmable and autonomous robots, machine learning and artificial intelligence have as large or even greater impact than what has gone before? Will programs learn to evolve themselves to pose and solve new problems? Do we face an optimistic future in which software artifacts are seen as partners or are the darker predictions of the replacement of humanity with smarter robotic software more credible?

What will be the state of affairs for computing in 2017 and how should we help to shape its future? What is the role of ACM and its members?

Should we be concerned about the increasingly pervasive role of software in our lives? Should we be learning about and teaching some form of "responsible programming?" Should there be incentives for software makers and providers to assure that credible measures have been taken to maximize safety and minimize risk to those using or dependent on software? What might those incentives look like? What should we do about preserving digital content including the software needed to correctly interpret it? What are the implications for intellectual property protection and for the interests of the general public? What is the future of ACM's Digital Library and the concept of open access to publications, data, and software?

The notion of "stock taking" seems very applicable here. What will be the state of affairs for computing in 2017 and how should we help to shape its future? What is the role of ACM and its members? What about the millions of professionals (and amateurs) with an interest in computing who may not be members but are affected by ACM's work? It is not too early to begin thinking about 2017 and, as we wish John White a fond farewell and his successor a warm welcome, let us begin the process of formulating a meaningful and substantive recognition of that milestone year. I await your suggestions with great anticipation! C

a http://en.wikipedia.org/wiki/Hilbert%27s_problems

b Scott Aaronson, The Limits of Quantum Computers, *Scientific American 298*, (2008) 62-69; doi:10.1038/scientificamerican0308-62.

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

DOI:10.1145/2785962

Quality vs. Quantity in Faculty Publications

As a former computer science department chair, I have long felt the system for evaluating and promoting faculty is far too dependent on quantity and fails to account adequately for quality. So I applaud the Computing Research Association best practices memo Moshe Y. Vardi mentioned in his Editor's Letter "Incentivizing Quality and Impact in Computing Research" (May 2015). However, more specific guidelines are needed for this initiative to be effective. I would urge all computing departments to inform candidates for assistant-professor positions their submitted CVs should list no more than three publications. Candidates must also be the primary author of at least two of them, meaning they were responsible for at least 50% of the intellectual content and at least 50% of the writing. On any other publication, the candidate would be considered a secondary author; yes, some papers may have no primary author.

I would go further. Faculty being considered for tenure should be permitted to list at most 15 publications and be a secondary author on at most five of them. Full-professor candidates should be permitted to list at most 25 and be a secondary author on at most 10. Also, in either case, candidates should be required to list three recent papers for which they are primary authors and on which they are to be explicitly evaluated.

These thresholds are meant as examples, not firm proposals. What is important is departments and research organizations send a clear message to candidates that CV-padding will not help their case for promotion and tenure.

To make evaluations of papers meaningful, they should be a written part of the hiring/promotion case. That is, for each paper to be evaluated, some expert in the candidate's field should be asked to study the subject paper and write a substantive review. Many departments will be able to handle these reviews internally. Those lacking faculty with the requisite expertise could commission reviews from outside experts, possibly in exchange for a nontrivial honorarium.

Jonathan Turner, Sarasota, FL

Author's Response:

The Editor's Letter in question indeed suggested leading computing-research organizations sign a statement committing to adopt the CRA memo as the basis for their own hiring and promotion practices. Such a statement can be more detailed than the CRA memo.

Moshe Y. Vardi, Editor-in-Chief

Python for Beginners Here to Stay

It was refreshing to see Python discussed as a primary teaching language by Esther Shein in her news article "Python for Beginners" (Mar. 2015). The Python experience for introductory students at my college—Washington and Lee University—has been nothing but positive. We knew when we switched from Java to Python a decade ago that the language would offer the résumé value of widespread use in industry, as well as the pedagogical value of stepwise, stress-free introduction of programming constructs, from simple algorithmic code to the use and definition of functions, objects, and classes. Minimal syntactic "pixie dust" and support for safe semantics are the most critical features of Python for novices. The language is well suited to the kinds of short programs assigned in beginning courses while appealing to students with widely different backgrounds.

Excellent APIs are available for creating easy GUI-based Python programs in introductory courses (http://home.wlu.edu/~lambertk/breezypythongui/index.html) and for teaching data structures and object-oriented programming in a systematic way (http://home.wlu.edu/~lambertk/classes/112/).

When students learn the basic principles of programming with Python in these courses, they are ready to move on to the wider world of software development involving more challenging languages like Java, C, and Erlang. But for beginners, Python is here to stay.

Kenneth Lambert, Lexington, VA

Esther Shein's news article (Mar. 2015) downplayed the robust nature of Python, perhaps in the interests of journalistic "balance." But the word count dedicated to critique seemed misplaced in an article that might otherwise have presented more positive examples of use. For example, the repeated reference to Python as a "scripting language" was dismissive. Python is not only useful for scripting applications or OS functions but for industrial applications with tens of thousands of LOC. I once led a development team that wrote a robust Web transaction B2B infrastructure in Python that could handle individual database files with millions of records. It is still running two decades later.

Python is strongly dynamically typed, enabling it to handle flexible runtime situations efficiently. If a programmer wants the advantages of static typing, it can be bolted on using a preprocessor like "mypy," an experimental optional static type checker for Python. A programmer could also use Python to implement automated testing, structured documentation, and other safe-coding practices. But doing so would be useful no matter what the language, and in any case goes beyond the target audience of "beginners."

Shriram Krishnamurthi, the Brown University professor quoted in the article, complained Python has limited support for data structures. What it does have is built-in tuple, list, set, array, and dictionary types, and queues, b-trees, ordered dictionaries, named tuples; other functions are part of the standard library. Moreover, third-party libraries exist for every other imaginable situation. For custom-mixed data structures, Python lets programmers

use a class. Krishnamurthi bemoaned their "onerous syntax and tricky semantics," which might be a reasonable criticism when a programmer goes beyond trivial cases, but for simple structures a programmer needs only the "pass" command, as in the following example:

```
class Album:
    pass
favourite = Album()
favourite.name = 'Revolver'
favourite.artist = 'The
Beatles'
favourite.year = 1966
```

A novice can understand this code once the instantiation at line 3 is explained. It is hardly "onerous," as Krishnamurthi claimed. From here, `put` and `get` methods, type checking, validation, and other functions can be added in an incremental way, an approach eminently suitable for beginners.

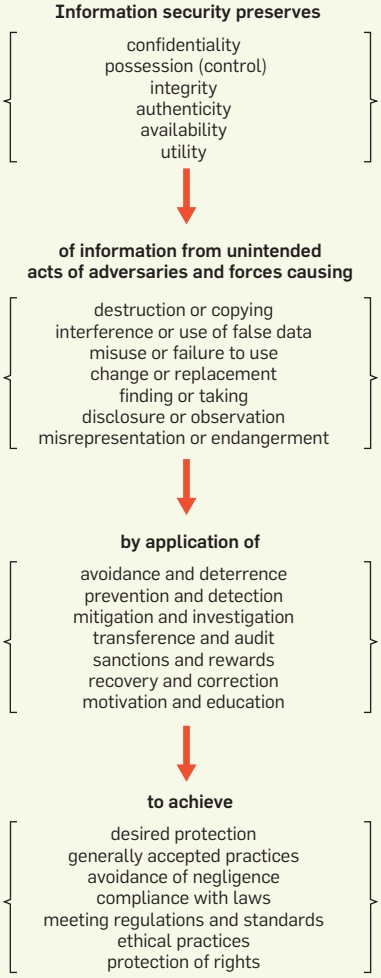
The problem of underestimating Python may involve bringing biases from one language into another. The lack of "struct," or static types, does not make Python any less capable. It could do the opposite. There are certainly domains where some other programming language would be preferable. But Python is the language for the 99%.

Robin Parmar, Limerick, Ireland

Comprehensive Diligence for Information Security

Security specialists must address information security as precisely and completely as possible because clever adversaries use attack methods and vulnerabilities defenders have overlooked. Security must be comprehensive, as it is no stronger than its weakest link. Building security of information should start with formal models that address all known material weaknesses and security controls and practices. Though Johannes Sametinger et al.'s review article "Security Challenges for Medical Devices" (Apr. 2015) was excellent describing many potentially mortal threats and calling for formal methods and models to achieve security, it was only one more article demonstrating the weak and deficient fundamentals of information security. It said, "Security is about protecting information and information systems from unauthorized access and use." Authorized but malicious access and use must also be addressed. Acts not mentioned include unintended representation, taking, interference, endangerment, failure to use, and false data entry. Privacy is indeed not a fundamental element of security but a human right achieved, in part, through protection of confidentiality, possession, integrity, authenticity, availability, and utility.

Information security fundamentals.



Sametinger et al. also said, "Confidentiality, integrity, and availability (CIA) of information are core design and operational goals." Possession and control, authenticity, and utility are also needed to achieve security. They said, incorrectly, confidentiality is threatened by disclosure, integrity by altering data, and availability of devices by inoperability. Confidentiality is threatened first by taking or observing. Disclosure means somebody in possession reveals information to another person. Integrity, meaning whole and sound, is not necessarily threatened by altering data. Altering data threatens information authenticity.

Moreover, the article switched from information to devices threatened by loss of availability. What about when information is unavailable or unusable? Sametinger et al. are not to blame for such oversights since these conundrums are abundant throughout the information-security literature and even in laws and regulations.

The figure here is a formal diligence model I devised for reviewing the information security of more than 200 enterprise clients (http://www.cbi.umn.edu/) that comes closer than any article I know to describing the basic elements of information security.

I challenge Sametinger et al., as well as all *Communications* readers, to identify any significant detail I overlooked in this model at the same level of abstraction and within normal dictionary meanings of its words.

Donn B. Parker, Los Altos, CA

Authors' Response:

Parker mentions the "... weak and deficient fundamentals of information security." It is not the fundamentals that are weak and deficient but their execution. Devices could be much more secure. Parker is right there are aspects of security like authorized but malicious access we have not considered. For now, though, we would be happy to avoid unauthorized access and trust our cardiologist. We agree privacy is a human right, but some security measures are needed to avoid disclosure.

Johannes Sametinger, Jerzy Rozenblit, Roman Lysecky, and **Peter Ott**

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.

ACM

ON A MISSION TO SOLVE TOMORROW.

Dear Colleague,

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

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

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

Enhance your professional career with these exclusive ACM Member benefits:

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

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

Sincerely,

Alexander Wolf
President
Association for Computing Machinery

Association for Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING.
JOIN ACM TODAY.

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

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

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

ACM STUDENT MEMBERSHIP:

- ❑ Student Membership: $19 USD
- ❑ Student Membership plus ACM Digital Library: $42 USD
- ❑ Student Membership plus Print *CACM* Magazine: $42 USD
- ❑ Student Membership with ACM Digital Library plus Print *CACM* Magazine: $62 USD

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

Priority Code: CAPP

Payment Information

Name

ACM Member #

Mailing Address

City/State/Province

ZIP/Postal Code/Country

Email

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

❑ AMEX ❑ VISA/MasterCard ❑ Check/money order

Total Amount Due

Credit Card #

Exp. Date

Signature

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

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

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

Satisfaction Guaranteed!

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.

Association for Computing Machinery

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

Hours: 8:30AM - 4:30PM (US EST)
Fax: 212-944-1318

acmhelp@acm.org
acm.org/join/CAPP

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:10.1145/2771281 http://cacm.acm.org/blogs/blog-cacm

The Dangers of Military Robots, the Risks of Online Voting

John Arquilla considers the evolution of defense drones, and why Duncan A. Buell thinks we are not ready for e-voting.

John Arquilla
"Meet A.I. Joe"
http://bit.ly/1HYARMi
May 1, 2015

Isaac Asimov's Three Laws of Robotics, first codified in his 1942 short story "Runaround" (http://bit.ly/1AAkKhW), sought to steer use of artificial intelligence (AI) in peaceful directions. Robots were never to harm humans, or by inaction allow them to come to harm. Within months of the elucidation of these laws, however, an extremely primitive robot, the Norden bombsight, was being put to lethal use by the U.S. Army Air Corps. The bombsight combined a computer that calculated various factors affecting an aircraft's arrival over a target with an autopilot. Touted for its accuracy from high altitude, the Norden nevertheless tended to miss the aim point by an average of a quarter-mile. Yet pilots routinely turned over control to the machine for the final run to target.

In the 70 years since the end of World War II, AI has advanced enormously, and the U.S. military has continued to show a steady appetite for acquiring lethal robots. Tomahawk cruise missiles, the great-grandchildren of the Norden, once launched find their own way to far-distant targets over even the most complicated terrain; yet the enemy sometimes eludes the Tomahawks, which end up killing the wrong people. The Phalanx ship-defense system is another important military robot—many missiles move too fast for human reflexes—but Phalanx, limited to close-in, Gatling-gun-like defensive fire, is unlikely to cause collateral damage to noncombatants.

For all the advances in AI, there remains a significant reluctance to embrace the notion of fully autonomous action by machines. Hence the popularity today of remotely controlled aircraft and ground combat systems. Indeed, the growth in the number of drone pilots has been explosive, and soldiers on the ground have become very attached to their machine buddies; there have been instances when drones have been given medals, and have been ceremonially buried when "killed in action." It is fascinating to see the great emotional appeal of the machines juxtaposed with the intellectual fear of what robots might do in the future, when they become more able to act independently.

That fear is more than just the residue of the dark tropes of the *Terminator* and *Matrix* film franchises—not to mention Ultron—or of the "Battlestar Galactica" TV series reboot; or, for that matter, of the worries about robots most recently expressed by luminaries like Stephen Hawking. There are real and practical concerns about autonomy and flawed machine judgment. How is a robot to determine the difference between enemy soldiers and noncombatants? This is a difficult-enough problem for human soldiers in the irregular wars of our time. Other questions are: When should a robot keep fighting or stop shooting if the foe is wounded, or is trying to surrender? In the case of a robot-piloted attack aircraft, can it discriminate between military and civilian targets adequately?

Even worse, can robots be hacked? The Iranians claim to have hacked an American drone and brought it down safely on their territory back in 2011. However it happened, they have it, and refused to return it when President Obama somewhat cheekily asked for it back. This incident should prompt us to consider the question: What if robots could be taken over and turned on their masters? A coup of this sort would require a high level of technological so-

phistication and an absolute mastery of the radio-electromagnetic spectrum, but the consequences of one side being able to do this would be catastrophic for the side whose robots were taken over.

Strategic concerns aside, on the political front, ethical and practical concerns about robots have been raised at the United Nations, which in an April 2013 report called for an immediate moratorium on the development and deployment of "lethal autonomous robots." The report is part of a valiant effort to stave off the onset of an AI arms race, but indicators from many places are that the race is already on. In Britain, for example, the Taranis combat aircraft (named for the Celtic god of thunder) has autonomous capabilities, unlike the Predators and other types of drones we have become familiar with over the past decade. The British are also moving toward fielding robot soldiers; their humanoid Porton Man reflects exceptional sophistication of design.

The Russians are not far behind, although their Strelok sharpshooter and Metalliste machine gun and grenade launcher systems, while apparently having some autonomous capability, seem to be under fairly close human control. But don't count on it staying that way.

The Chinese military is making swift advances in both remote-controlled and autonomous systems, on land, at sea, and in the air. Their advances in naval mine warfare include weapons that can sense the type of ship coming along and move stealthily to attack it. There is also evidence of Chinese sea mines with a capability to detect and then attack helicopters flying above them.

Against these threats, the U.S. Navy is developing autonomous mine-clearing technologies, giving the undersea fight an increasingly robot vs. robot flavor.

The same is true in cyberspace, where the sheer speed and complexity of offensive and defensive operations are driving a shift to reliance on robots for waging cyberwars. This is an area about which little can be said openly save that here is yet more evidence that the arms race the United Nations seeks to prevent is well under way. That suggests it is high time for an international conference, and an accompanying discourse, on the prospects for crafting robotic arms control agreements. In this way, we can keep alive the ideal of peaceful robotics that Asimov introduced so long ago.

There may be no way, ultimately, to stop the spread of killer robots, but at the very least they should be obligated to observe the laws of armed conflict, like their human counterparts.

Duncan A. Buell
"Computer Security and the Risks of Online Voting"
http://bit.ly/1CeFQ4A
April 2, 2015

One hears from the science community that scientists need to stand up and explain their science to the public. The usual topics—climate change, evolutionary biology, stem cell research—are outside the normal scope of most ACM members, but we in the computer sciences are the experts on one matter that has enormous impact.

In the wake of the 2000 U.S. presidential election, Congress passed the Help America Vote Act (HAVA). The nation largely turned away from older election technology and moved to computer-based systems, in some states relying entirely on the software and sometimes-arcane procedures that provided no secondary method against which to compare the tally produced by the software.

Now, as the HAVA machines age, many election officials around the country and around the world seem enchanted with the marketing hype of Internet voting software vendors and are buying in to the notion that we could—and should—vote online now and in the very near future.

Never mind the almost-daily reports of data breaches of financial organizations with deep pockets to spend on securing their computers. Never mind that governments, with shallower pockets, are routinely hacked, or that former FBI director Robert Mueller went on the record that the only two kinds of computers are those that have already been hacked and those that no one has yet bothered to hack. Election officials seem in awe of ill-defined vendor terms like "military-grade encryption."

In a small corner of the ACM world, scientists wonder why officials are not hearing the message of computer security experts. It is time for that small corner to expand and for the membership to find a voice in hopes that election officials can be made to listen.

ACM members are moderately aware that Alex Halderman of the University of Michigan and his students (legitimately) hacked the test Internet voting software of the District of Columbia. They are probably less aware that Halderman and Vanessa Teague of the University of Melbourne recently demonstrated that vendor software being used for an Australian statewide election was subject to a standard security flaw. Teague and Halderman responded in the best manner of professionals recognizing a flaw: they alerted the Australian CERT, which alerted the New South Wales Election Commission (NSWEC), which patched the software. Then Halderman and Teague went public, but the software patch to prevent votes from being intercepted and changed in flight to election central did not happen until one-fourth of the total online votes were cast.

The response from the NSWEC was both dismissive and frightening. On the one hand, it was admitted the NSWEC had factored the likelihood of corrupted votes into their analysis. On the other hand, they targeted Teague with a formal complaint to her university, accusing her of a breach of ethics and suggesting further that a DoS attack was coming from the University of Michigan. The NSWEC then had the audacity to publish with the vendor a puff piece (http://bit.ly/1GVp4Nr) on online voting without mentioning the security flaw.

The Australian election is not an anomaly. Many U.S. states are toying with the notion of online voting, contracting their elections to private companies whose code has never been given a public vetting. As scientists, we would all probably rather be doing science than trying to find ways to convince the public and election officials that security online today is not up to the task of voting online today. Yet the need is critical for disseminating the hard facts about computer security and the huge risks of online voting. We must take this on as a professional responsibility. The nation and the world deserve no less than our full involvement.

John Arquilla is professor and chair of defense analysis at the United States Naval Postgraduate School. **Duncan A. Buell** is professor of computer science and engineering at the University of South Carolina.

Science | DOI:10.1145/2771283 Chris Edwards

Growing Pains for Deep Learning

Neural networks, which support online image search and speech recognition, eventually will drive more advanced services.

Advances in theory and computer hardware have allowed neural networks to become a core part of online services such as Microsoft's Bing, driving their image-search and speech-recognition systems. The companies offering such capabilities are looking to the technology to drive more advanced services in the future, as they scale up the neural networks to deal with more sophisticated problems.

It has taken time for neural networks, initially conceived 50 years ago, to become accepted parts of information technology applications. After a flurry of interest in the 1990s, supported in part by the development of highly specialized integrated circuits designed to overcome their poor performance on conventional computers, neural networks were outperformed by other algorithms, such as support vector machines in image processing and Gaussian models in speech recognition.

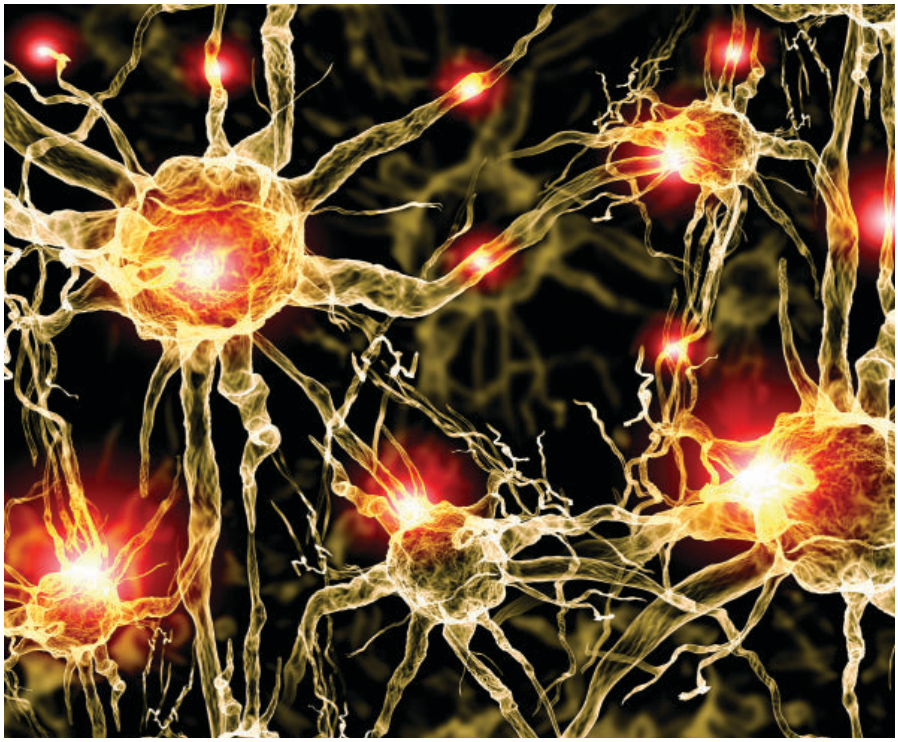


IMAGE BY SERGEY NIVENS

Older simple neural networks use only up to three layers, split into an input layer, a middle 'hidden' layer, and an output layer. The neurons are highly interconnected across layers. Each neuron feeds its output to each of the neurons in the following layer. The networks are trained by iteratively adjusting the weights that each neuron applies to its input data to try to minimize the error between the output of the entire network and the desired result.

Although neuroscience suggested the human brain has a deeper architecture involving a number of hidden layers, the results from early experiments on these types of systems were worse than for shallow networks. In 2006, work on deep architectures received a significant boost from work by

Geoffrey Hinton and Ruslan Salakhutdinov at the University of Toronto. They developed training techniques that were more effective for training networks with multiple hidden layers. One of the techniques was 'pre-training' to adjust the output of each layer independently before moving on to trying to optimize the network's output as a whole. The approach made it possible for the upper layers to extract high-level features that could be used more efficiently to classify data by the lower, hidden layers.

Even with improvements in training, scale presents a problem for deep learning. The need to fully interconnect neurons, particularly in the upper layers, requires immense compute power. The first layer for an image-processing application may need to analyze a million pixels. The number of connections in the multiple layers of a deep network will be orders of magnitude greater. "There are billions and even hundreds of billions of connections that have to be processed for every image," says Dan Cireşan, researcher at the Manno, Switzerland-based Dalle Molle Institute for Artificial Intelligence Research (IDSIA). Training such a large network requires quadrillions of floating-point operations, he adds.

Researchers such as Cireşan found it was possible to use alternative computer architectures to massively speed up processing. Graphics processing units (GPUs) made by companies such as AMD and nVidia provide the ability to perform hundreds of floating-point operations in parallel. Previous attempts to speed up neural-network training revolved around clusters of workstations that are slower, but which were easier to program. In one experiment in which a deep neural network was trained to look for characteristic visual features of biological cell division, Cireşan says the training phase could have taken five months on a conventional CPU; "it took three days on a GPU."

Yann LeCun, director of artificial intelligence research at Facebook and founding director of New York University's Center for Data Science, says, "Before, neural networks were not breaking records for recognizing continuous speech; they were not big enough. When people replaced Gaussian models with deep neural nets, the error rates went way down."

Deep neural nets showed an improvement of more than a third, cutting error rates on speech recognition with little background noise from 35% to less than 25%, with optimizations allowing further improvements since their introduction.

There are limitations to this form of learning. London-based DeepMind—which was bought by Google in early 2014 for $400 million—used computer games to evaluate the performance of deep neural networks on different types of problems. Google researcher Volodymyr Mnih says the system cannot deal with situations such as traversing a maze, where the rewards only come after successfully completing a number of stages. In these cases, the network has very little to learn from when it tries various random initial maneuvers but fails. The deep neural network fares much better at games such as Breakout and Virtual Pinball, where success may be delayed, but it can learn from random responses.

When it comes to deploying deep networks in commercial applications, teams have turned to custom computer designs using field-programmable gate arrays (FPGAs). These implement custom electronic circuits using a combination of programmable logic lookup tables, hard-wired arithmetic logic units optimized for digital signal processing, and a matrix of memory cells to define how all of these elements are connected.

Chinese search-engine and web-services company Baidu, which uses deep neural networks to provide speech recognition, image searches, and to serve contextual advertisements, decided to use FPGAs rather than GPUs in production servers. According to Jian Ouyang, senior architect at Baidu, although individual GPUs provide peak floating-point performance, in the deep neural network applications used by Baidu, the FPGA consumes less power for the same level of performance and could be mounted on a server blade, powered solely from the PCI Express bus connections available on the motherboard. A key advantage of the FPGA is that because the results from one calculation can be fed directly to the next

ACM Member News

ELLIOTT CONSIDERS ONE MORE BIG PROJECT

Chip Elliott has a triple-threat career: he is chief scientist at Raytheon BBN Technologies, adjunct professor of computer science at Dartmouth College, and Futures Director at the National Science Foundation's Global Environment for Network Innovations (GENI). Elliott is also an active inventor, with over 90 issued patents.

Elliott was born in Connecticut, raised in Kansas, and "escaped back to New England" to earn a B.A. in mathematics from Dartmouth College in New Hampshire. He began his computer science career as a programmer, teaching and founding the startup True Basic. His passion is research "seven to 10 years beyond current economic drivers, to explore [my vision]."

One such project involved collaborating with quantum physicists and photonics experts to build the world's first quantum network, linking Raytheon BBN with Harvard and Boston universities.

His research at GENI centers on large-scale cloud infrastructures. GENI provides 3,000 researchers at 50 U.S.-based universities with a virtual laboratory to promote innovation. "Most clouds are pre-baked; we bake our own," Elliott says, adding, "Our researchers are programming new types of clouds that are better at processing big data."

Among the initiatives he is keen to pursue at BBN are "futuristic cellular Internets"; fully homomorphic encryption—"done correctly, you can perform computation on encrypted data without unencrypting it"; synthetic biology, "to make cells into engineered systems, performing advanced tasks like producing diesel fuel inside plant cells," and producing machines that "sense interactions with chemicals, light, and pressure, to incorporate in the human bloodstream" for the potential medical benefits.

—Laura DiDio

without needing to be held temporarily in main memory, the memory bandwidth requirement is far lower than with GPU or CPU implementations.

"With the FPGA, we don't have to modify the server design and environment, so it is easy to deploy on a large scale. We need many functions to be supported that are impossible to deploy at the same time in FPGA. But we can use their reconfigurability to move functions in and out of the FPGA as needed. The reconfiguration time is less than 10µs," says Ouyang.

The Baidu team made further space savings by using a simplified floating-point engine. "Standard floating-point implementations provided by processors can handle all possible exceptions. But in our situation we don't need to handle all of the exceptions of the IEEE [754] standard."

As well as finding ways to use more effective processors, researchers are trying to use distributed processing to build more extensive deep-learning networks that can cope with much larger datasets. The latency of transfers over a network badly affects the speed of training. However, rearranging the training algorithms together with a shift from Ethernet networking to Infiniband, which offers lower latency, allowed a team from Stanford University in 2013 to achieve almost linear speed-ups for multiple parallel GPUs. In more recent work using clusters of CPUs rather than GPUs, Microsoft developed a way to relax the synchronization requirements of training to allow execution across thousands of machines.

More scalable networks have made it possible for Baidu to implement an "end to end" speech recognition system called Deep Speech. The system does not rely on the output of traditional speech-processing algorithms, such as the use of hidden Markov models to boost its performance on noisy inputs. It reduced errors on word recognition to just over 19% on a noise-prone dataset, compared to 30.5% for the best commercial systems available at the end of 2014.

However, pre-processing data and combining results from multiple smaller networks can be more effective than relying purely on neural networks. Cireşan has used a combination of image distortions and

Researchers are trying to use distributed processing to build more extensive deep-learning networks that can cope with much larger datasets.

"committees" of smaller networks to reduce error rates compared to larger single deep-learning networks. In one test of traffic-sign recognition, the combination of techniques resulted in better performance than human observers.

Deciding on the distortions to use for a given class of patterns takes human intervention. Cireşan says it would be very difficult to have networks self-learn the best combination of distortions, but that it is typically an easy decision for humans to make when setting up the system.

One potential issue with conventional deep learning is access to data, says Neil Lawrence, a professor of machine learning in the computer science department of the University of Sheffield. He says deep models tend to perform well in situations where the datasets are well characterized and can be trained on a large amount of appropriately labeled data. "However, one of the domains that inspires me is clinical data, where this isn't the case. In clinical data, most people haven't had most clinical tests applied to them most of the time. Also, clinical tests evolve, as do the diseases that affect patients. This is an example of 'massively missing data.'"

Lawrence and others have suggested the use of layers of Gaussian processes, which use probability theory, in place of neural networks, to provide effective learning on smaller datasets, and for applications in which the neural networks do not perform well, such as data that is interconnected across many different databases, which is the case in healthcare. Because data may not be present in certain databases for a given candidate, a probabilistic model can deal with the situation better than traditional machine-learning techniques. The work lags behind that on neural networks, but researchers have started work on effective training techniques, as well as scaling up processing to work on platforms such as multi-GPU machines.

"We carry an additional algorithmic burden, that of propagating the uncertainty around the network," Lawrence says. "This is where the algorithmic problems begin, but is also where we've had most of the breakthroughs."

According to Lawrence, deep-learning systems based on Gaussian processes are likely to demand greater compute performance, but the systems are able to automatically determine how many layers are needed within the network, which is not currently possible with systems based on neural networks. "This type of structural learning is very exciting, and was one of the original motivations for considering these models."

In currently more widespread neural-network systems, Cireşan says work is in progress to remove further limitations to building larger, more effective models, "But I would say that what we would like mostly is to have a better understanding of why deep learning works." C

Further Reading

Hinton, G.E., and Salakhutdinov, R.R.
Reducing the dimensionality of data with neural networks, *Science* (2006), Vol 313, p 504.

Schmidhuber, J.
Deep learning in neural networks: an overview, *Neural Networks* (2015), Volume 61, pp85-117 (ArXiv preprint: http://arxiv.org/pdf/1404.7828.pdf)

Mnih, V., et al
Human-level control through deep reinforcement learning, *Nature* (2015), 518, pp529-533

Damianou A.C. and Lawrence N.D.
Deep Gaussian processes, *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS) 2013.* (ArXiv preprint: http://arxiv.org/pdf/1211.0358.pdf)

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

Technology | DOI:10.1145/2771299 Gregory Goth

Bringing Big Data to the Big Tent

Open source tools assist data science.

Some experienced hands in the field of data analysis feel the differences between investigational data scientists, who work on the leading edge of concepts using statistical tools such as the R programming environment, and operational data scientists, who have traditionally used general-purpose programming languages like C++ and Java to scale analytics to real-time enterprise-level computational assets, need to become less relevant.

That sentiment is growing, especially as tools emerge that enable individual scientists to analyze ever-larger amounts of data. Though many, if not most, of these data practitioners are not considered software developers in the traditional sense, the data their analysis creates becomes itself an increasingly valuable resource for many others.

"It is increasingly important to bridge that gap, for a few reasons," says Michael Franklin, director of the AMPLab, an open source innovation lab at the University of California at Berkeley. Franklin's reasons for closing the gap include:

- Having a unified system for both speeds up discovery by "closing the loop" between exploration and operation.
- Having different systems introduces the potential for error—a model created in the "investigational" tool set may be translated incorrectly to the (different) operational system—sometimes these errors can be quite subtle and go unnoticed.
- Increasingly, data scientists are responsible for both exploration and production.

The tools around many of the most innovative and promising data science initiatives are often open source resources that range from the R statistical programming language, widely used by individual researchers in the life sciences, physical sciences, and social sciences alike, to the enterprise-friendly Apache Spark cluster computing platform and its companion machine learning library developed at the AMPLab.

The Scientific Software Network Map allows interactive exploration of data showing software tools used, usage trends, which packages were used together to produce a result, and publications resulting from that use.

The attraction of open source tools might be ascribed to the academic culture in which much cutting-edge data science is being done, according to Stanford University marine biologist Luke Miller. Miller started using R when he began working at an institution that did not have a free site license for Matlab; his observations are shared by Scott Chamberlain, co-founder of rOpenSci, which creates open source scientific R packages, collections of R functions, data, and compiled code that help reduce redundant coding.

"Academics have this culture of not wanting to have to pay for things," Chamberlain says. "They want to use things that either the university has a license for, or they want it to be free. That's the culture; love it or hate it, that's the way it is."

Maturing the R Ecosystem

While the free culture around R has allowed it to spread to many disciplines—it has narrowly outpaced Python in the past two O'Reilly data science tool surveys—it has also become rather unwieldy, according to Jim Herbsleb, a professor of computer science at Carnegie Mellon University.

"There's a lot duplication of effort out there, a lot of missed opportunities, where one scientist has developed a tool for him or herself, and with a few tweaks, or if it conformed to a particular standard or used a particular data format, it could be useful to a much wider community," he says. "R is all over the world, on everybody's laptops, so it's really hard to get a sense of what's happening out in the field. That's what we're trying to address."

Herbsleb and post-doctoral researcher Chris Bogart have created the Scientific Software Network Map (http://scisoft-net-map.isri.cmu.edu/), essentially a meta-tool that allows interactive exploration of data showing which software tools have been used, how use is trending, which packages are used together to produce a result, and the publications that resulted from that use.

Herbsleb views the newly launched project, funded by a National Science

IMAGE COURTESY OF THE SCIENTIFIC SOFTWARE NETWORK MAP PROJECT

Foundation grant, more as a reference point for R package developers who wish to see how end users are utilizing their work than for those seeking tools. "If developers know what other people need or want from a tool they are working on, they are often willing to do some extra work to make sure it works for everybody," he says. "That's another thing one could hope to get from the map: a better understanding of what community needs are by seeing how the tools are actually being used. Another use case is for communities of scientists who either already do, or would like to, start managing their software assets as a community."

However, Larry Pace, author of several books on R, thinks Herbsleb's network map could be exactly the reference neophyte R users, as well as developers, would find very useful.

"I can see the various R packages I use on this map; it's a really, really cool way to show this," Pace says. "There are, even in R right now, tools that do what psychologists do, psychometric analyses and things like that. They're just not well publicized, so people don't know about them."

For experienced end users like Miller, there appears to be a critical mass of mailing list exchanges, an index of domain-specific "task views" on the Comprehensive R Archive Network (CRAN) website (http://cran.r-project.org/), and dialogue within scientific communities that facilitate reuse of code and overall expansion of bodies of knowledge codified in R; what one may term the "operational" side of the worldwide availability of reproducible data.

"There are enough R users who have written very useful scripts that I can piggy-back onto and open these same data files and plunk through them," he says. "Python in some ways may be a better way to do that, certainly in terms of time invested; a good Python programmer probably would have gotten it done faster than I did in R. We're not doing wildly fancy programming here; we're just opening data files and parsing them. But in terms of integrating into the rest of my data visualization and analysis workflow, there's very little reason to step outside of R for that sort of task."

User-Friendly ML Emerging

Developers of machine learning tools are also embracing non-traditional communities of data scientists, touting speed and ease of use rather than programming environment as selling points, though R interfaces are increasingly viewed as important features.

For some of these projects, survival seems assured, through ample funding and acceptance by the open source infrastructure. Spark, for example, which boasts in-memory processing capabilities that can be 100 times faster than Hadoop MapReduce, was initially developed as part of a $10-million National Science Foundation award to the Berkeley AMPLab and granted top-level status by the Apache Software Foundation in 2014. The developers of Spark (which includes the MLbase machine learning stack) cite rapid growth in the number of people trained on it; AMPLab co-director Ion Stoica estimates 2,000 received training in 2014 and estimates 5,000 will receive training in 2015. The first Spark Summit, held in December 2013, had 450 attendees and a second, in June 2014, had 1,100.

"Today, Spark is the most active big data open source project in terms of number of contributors, in number of

Milestones

Oxford Celebrates Lovelace's 200th Birthday

This year, the University of Oxford will celebrate the 200th anniversary of the birth of computer visionary Ada Lovelace.

The centerpiece of the celebration will be a display at the University of Oxford's Bodleian Library (to run Oct. 29 through Dec. 18) and a symposium on Dec. 9–10 presenting Lovelace's life and work, and contemporary thinking on computing and artificial intelligence.

Ada, Countess of Lovelace (1815–1852), is best known for an article she wrote about Charles Babbage's unbuilt computer, the Analytical Engine, that presented the first documented computer program, designed to calculate Bernoulli numbers, and explained the ideas underlying Babbage's machine–which similarly underlie every computer and computer program in use today. Lovelace also wrote about computers' creative possibilities and limits; her contribution was highlighted in one of Alan Turing's most famous papers, "Computing Machinery and Intelligence," as "can a machine think?"

The Symposium will be aimed at a broad audience interested in the history and culture of mathematics and computer science, and will present current scholarship on Lovelace's life and work, and link her ideas to contemporary thinking about computing, artificial intelligence, and the brain. Confirmed speakers include Lovelace biographer Betty Toole, computer historian Doron Swade, mathematician Marcus du Sautoy, and graphic novelist Sydney Padua.

Other activities will include a workshop for early career researchers, a "Music and Machines" event, and a dinner at Oxford's Balliol College on Dec. 9, the eve of Lovelace's 200th birthday.

Oxford's celebration will be led by the Bodleian Libraries and the University of Oxford's Department of Computer Science, working with colleagues in the Mathematics Institute, Oxford e-Research Centre, Somerville College, the Department of English and TORCH.

For more information or to register your interest, please visit http://blogs.bodleian.ox.ac.uk/adalovelace/.

FURTHER CELEBRATION

On Oct. 16, Oxford's Sommerville College will host the Ada Lovelace Bicentenary: Celebrating Women in Computer Science, a celebration of the life of the mathematician and scientific visionary and of women in science.

The event will take place in the Flora Anderson Hall of Somerville College. Speakers will include:

- **Cecilia Mascolo, professor of Mobile Systems at University of Cambridge**
- **Jennifer Widom, Fletcher Jones Professor and chair of the Computer Science Department, Stanford University**
- **Ulrike Sattler, professor of Computer Science, University of Manchester**
- **Ursula Martin CBE, professor of Computer Science, University of Oxford**
- **James Essinger, author of Ada Lovelace, A Female Genius: How Ada Lovelace started the Computer Age (2013)**

Somerville College also will exhibit materials from Ada Lovelace's literary life.

Somerville College's Ada Lovelace Bicentenary Celebration will be chaired by Mason Porter, tutor in Applied Mathematics at Somerville College.

For more information about attending Sommerville College's Ada Lovelace Bicentenary Celebration, please contact Barbara.raleigh@some.ox.ac.uk.

—Lawrence M. Fisher

commits, number of lines of code—in whatever metric you are using, it's several times more active than other open source projects, including Hadoop," Stoica says.

To help attract data scientists who have not traditionally been users of machine learning at scale, the Spark team has developed features including:

- Spark's MLlib machine learning library already ships with application programming interfaces (APIs) such as a pipelines API, which streamlines the sequence of data pre-processing, feature extraction, model fitting, and validation stages by a sequence of dataset transformations. Each transformation takes an input dataset and outputs the transformed dataset, which becomes the input to the next stage.
- Spark's developers have also been perfecting its R interface. Introduced in January 2014, early versions of SparkR supported a distributed list-like API that maps to Spark's Resilient Distributed Dataset (RDD) API. More recently, says SparkR developer Shivaram Venkataraman, the team is close to releasing a DataFrame API that will allow R users to use a familiar data frame concept, but now on large amounts of data. Venkataraman says the developers also have some work in progress in SparkR that will allow R users to call Spark's machine learning algorithms by using the pipelines. At a high level, Venkataraman says, the interface will allow R users to use specific columns of the DataFrame for machine learning and parameter tuning.
- Franklin also says the MLbase project, still under development, is going one step further to democratize machine learning over big data by allowing users to specify what they want to predict and then having the underlying system determine the best way to accomplish that using MLlib and the rest of the scalable Spark infrastructure.

A much smaller project than Spark, called mlpack, also targets those not experienced with machine learning. The project's "primary maintainer," Ryan Curtin, a Georgia Institute of Technology doctoral student, says the project, which launched its 1.0 version in December 2011, now includes five or six consistent developers, 10 to 12 occasional contributors, and perhaps 20 who may send in one contribution and then move on.

Spark boasts in-memory processing capabilities that can be 100 times faster than Hadoop MapReduce.

Curtin says mlpack appeals to non-experts through a consistent API that provides default parameters that can be left unspecified. Additionally, a user could move from one method to another while expecting to interact with the new method in the same way. Expert users may use its native C++ environment to customize their work.

Curtin also says that, though mlpack and Spark's target audiences may differ, the tool can be very useful for operational tasks in areas such as the burgeoning field of data-driven population health management. A data analyst for a public health department or healthcare delivery system, for example, could use one of mlpack's clustering algorithms on a set of numerical values that include fields such as a person's name and location (both mapped to a granular numerical format) and vital health data such as blood glucose and blood pressure, to predict where clinicians will have to deliver more intense care.

"There are a couple of clustering algorithms in mlpack you could use out of the box, like k-means and Gaussian Mixture Models, and there are enough flexible tools inside mlpack that if you really want to dig deep into it and write some C++, you can really fine-tune the clustering algorithm or write a new or modified one."

While mlpack research shows benchmarks significantly faster than those of other open source platforms such as Shogun (which originally focused on large-scale kernel methods and bioinformatics) and Weka (a popular general-purpose machine learning platform with a graphical user interface that may appeal to those uncomfortable with command line interfaces) in a test of k-nearest neighbors, Curtin says ultimately, the prevailing vibe among the community is cooperative more than competitive.

"We all sort of have the same goal," he says, "but there's an understanding the Shogun guys built their own thing. Shogun focuses on kernel machines and supports vector machines and classifiers of that ilk. mlpack focuses more on problems where you're comparing points in some space, like nearest neighbor."

That sense of community may prove paramount for small projects such as mlpack. While Curtin says the growth curve of mlpack's downloads is becoming exponential, he has limited time to develop features that might spur growth even further among non-expert users, such as designing a GUI and automatic bindings for other languages.

Those things, in fact, may be for somebody else to do, says Curtin.

"My interest is less in seeing mlpack as itself grow, although that's nice," he says. "I want to see the code get used more than anything else. That's what brought me to open source in the first place; building things that are useful for people." C

Further Reading

Matloff, N.
The Art of R Programming: A Tour of Statistical Software Design, *No Starch Press*, San Francisco, CA, 2011.

Pace, L.
Beginning R: An Introduction to Statistical Programming, *Apress*, New York, NY, 2012.

Wickham, H.
The Split-Apply-Combine Strategy for Data Analysis, *Journal of Statistical Software 40 (1)*, April 2011

Sparks, E., Talwalkar, A., Smith, V., Kottalam, J., Pan, X., Gonzalez, J., Franklin, M., Jordan, I., and Kraska, T.
MLI: An API for Distributed Machine Learning, *International Conference on Data Mining*, Dallas, TX, Dec. 2013

Curtin, R., Cline, J., Slagle, N.P., March, W., Parikshit, R., Mehta, N., and Gray, A.
MLPACK: A Scalable C++ Machine Learning Library, *Journal of Machine Learning Research 14*, March 2013

Gregory Goth is an Oakville, CT-based writer who specializes in science and technology.

Society | DOI:10.1145/2771297 Gregory Mone

The New Smart Cities

How urban information systems are slowly revamping the modern metropolis.

More than half of the world's population currently lives in or around a city. By the year 2050, the United Nations projects another 2.5 billion people could be moving to metropolises. As urban populations increase, the number of data-generating sensors and Internet-connected devices will grow even faster. Experts say cities that capitalize on all the new urban data could become more efficient and more enjoyable places to live. The big question now is how to make that happen.

Until recently, this seemed to be as simple as choosing the right out-of-the-box smart city solution from a big multinational corporation. "The overhyped promises from the big technology companies have kind of blown away," says Anthony Townsend, Senior Research Scientist at New York University's Rudin Center for Transportation Policy and Management. "Now you have city governments regrouping and developing comprehensive long-range visions of the role information technology will play in making their city better."

The leading urban centers are not placing their technological futures in the hands of a company or a single university research group. Instead, they are relying on a combination of academics, civic leaders, businesses, and individual citizens working together to create urban information systems that could benefit all these groups. The challenges are diverse and demanding, but a handful of new projects around the world are providing a glimpse of what a truly smart city could offer.

Chicago will be home to the Array of Things, a planned citywide installation of at least 500 sensor-packed boxes.

IMAGE BY ALEX DIBROVA

The Sensing City

In Chicago, Argonne National Laboratory computer scientist Charlie Catlett is leading a collaboration called the Array of Things, a planned citywide installation of at least 500 sensor-packed boxes. Catlett, who is also director of the Urban Center for Computation and Data, says the project began as an effort to better understand air quality downtown, but after conversations with city officials and other scientists, it expanded in scope.

The boxes will be approximately the height and width of a parking sign and attached to poles or other fixtures. Each unit will have 17 or 18 sensors, and although the final implementation has not been determined, the boxes will likely measure temperature, precipitation, humidity, air quality, and pedestrian flow. Microphones, for instance, could detect noise pollution or trucks idling in one spot for too long.

The data captured will be valuable to a variety of groups. Atmospheric scientists hoping to understand urban climate will have access to detailed information on air quality, temperature, and precipitation. Social scientists will be able to study pedestrian counts and flows to analyze how people move through urban spaces. City planners will be able to make more informed decisions about where to place new bus stops or how much road salt to apply to certain areas after a heavy snow. There is also the potential to use the data to fight gridlock. "The traffic controllers would like these on every corner," Catlett says.

Average citizens will benefit as well. Today, a runner equipped with a smartphone or a wearable fitness device can easily track her route, mileage, average heart rate, and total steps; now, imagine that gadget talking to the city itself. The Array of Things units could exchange data each time a wearable device communicating via a Bluetooth or Wi-Fi signal comes within range. "Our phones can keep track of so much," says Catlett, "but I'm not going to find out my exposure to particulate matter or carbon monoxide or ozone. This new device will tell me my exposure to various emissions; then, maybe I can use that to change my route or manage my exposure."

The city of Glasgow, as part of a project called Future City Glasgow,

is testing a similar technology: an installation of intelligent street lamps that will measure air quality, traffic, and pedestrian flow. Yet Glasgow is also depending on a broader range of sources to drive its transformation into a smarter metropolis. Recently, Glasgow announced the launch of its City Data Hub, a cloud-based, publicly accessible information store drawn from more than 400 datasets. The Hub provides data on traffic, pedestrian footfall in retail areas, parking spots at public garages, and more. Eventually, it could tap into other sources as well, such as smartcard readers at subway turnstiles. "We're creating an ecosystem of data providers throughout the city," says Colin Birchenall, the technology architect for Future City Glasgow.

Citizens First

In all these projects, organizers hope this ecosystem will include the citizens themselves. A city can only install so many smartlights or lamppost-mounted units, but almost every resident will be carrying a sensor-packed smartphone. Tapping into that data will provide a whole new layer of information. For citizens to volunteer this information, though, there has to be a clear value proposition.

As an example, Birchenall cites Glasgow's new cycling app, which helps riders plan routes, easily locate bicycle racks, and more; the app requests the right to gather data on a user's trips. There is no immediate benefit, but Birchenall says cyclists have opted in because of the potential long-term benefits. The idea is that if users do volunteer their data, then city planners will be better able to analyze cycling traffic, and potentially improve urban infrastructure by adding racks or widening lanes.

This approach is also behind certain aspects of the Quantified Community project, a collaboration between real estate developers, technology and design firms, and researchers at New York University's Center for Urban Science and Progress (CUSP). As the developers revamp and rebuild several derelict city blocks in New York, CUSP researchers are working with them to install a range of smart systems. There will be sensors that operate on the large scale, tracking waste disposal and recycling, but the researchers also hope residents will play an active role as willing sources of data.

CUSP deputy director Constantine Kontokosta points to the installation of sensors that track energy and water consumption down to individual outlets and faucets. The data will be anonymized, and while residents will still be able to opt out, Kontokosta suspects this information will prove appealing. "The value is clear," he says. "We're saying that if you let us collect information about how much energy and water you're consuming, then we will analyze the data, and we'll tell you about others so you can gauge how you're doing." The residents might be inspired to change their behavior as a result, and perhaps use less water or energy.

The Quantified Community and the Array of Things projects hope to track pedestrian movement as well, but in both cases, the researchers have designed the systems to preserve privacy. If an Array of Things unit was measuring pedestrian flow, and you were to walk past with a Bluetooth-enabled smartphone, the device might register that signal, but it would not store the unique signature from your phone; instead, it would simply interpret that signal as the presence of a person and delete the private data, so there would be no record that you, specifically, were there. "Our plan is to count the number of unique addresses that we can see, then throw those addresses away and report only the count," Catlett explains. "This is meant to be a public data utility, so we have to think hard about how to protect people's privacy."

Open Data, Endless Possibilities

This public utility approach is one of the common threads linking the different smart city projects. The Quantified Community project will publicly release all of its data. Similarly, the Array of Things units are designed to dispatch information to Argonne National Laboratory for quality control and calibration, then on to the City of Chicago for mass dissemination through its open data portal.

In Glasgow, the City Data Hub makes all the information gathered easily accessible to local residents. The city is counting on them to make use of it, too, by working closely with urban informatics groups at local universities, encouraging businesses to use the data on pedestrian density to determine where to establish new shops, and calling for enterprising citizens to come up with new solutions on their own. "It's an opportunity to stimulate innovation within the city," says Birchenall. "You don't have to rely on the government. If the data is open, anyone could try to solve the city's problems through technology."

The flood of information could also create new challenges, according to Pete Beckman, director of the Exascale Technology and Computing Institute at Argonne and a leader in the Array of Things project. "We'll have so much data about our smart cities, but will we have the operating system?" asks Beckman.

Overall, the changes to these smart cities will be incremental, whether they are driven by researchers like Beckman and Catlett, urban planners, private innovators, or some combination. "You're not going to wake up one day and live in the city of the future," says Townsend. "Cities are systems of systems. It takes time to bring about large-scale change, but in the future I think we will be living in cities that fundamentally operate differently." C

Further Reading

Townsend, A.
Smart Cities: Big Data, Civic Hackers, and the Quest for a New Utopia, W.W. Norton, 2013.

Li, T., Keahey, K., Sankaran, R., Beckman, P., and Raicu, I.
A Cloud-based Interactive Data Infrastructure for Sensor Networks, *IEEE/ACM Supercomputing/SC*, 2014.

Shin, D.H.
Ubiquitous City: Urban Technologies, Urban Infrastructure, and Urban Informatics, *The Journal of Information Science*, 2009.

Carvalho, L.
Smart Cities from Scratch? A Socio-Technical Perspective, *The Cambridge Journal of Regions, Economy and Society*, 2014.

Future City Glasgow: A Day in the Life http://bit.ly/1GMuHwP

Gregory Mone is a Boston, MA-based science writer and the author of the children's novel *Fish*.

Awards | DOI:10.1145/2773558 Lawrence M. Fisher

ACM Announces 2014 Award Recipients

Recognizing excellence in technical and professional achievements and contributions in computer science and information technology.

ACM recently announced the recipients of its 2014 Software System Award, Distinguished Service Award, Outstanding Contribution to ACM Award, and Doctoral Dissertation Awards.

Software System Award

Mach, a pioneering operating system used as the basis for later operating systems, received the 2014 ACM Software System Award. Lead developers **Rick Rashid** of Microsoft and **Avadis (Avie) Tevanian** created an operating system, whose innovative approaches to virtual memory management and microkernel architecture established a foundation for later operating systems on personal computers, tablets, and mobile phones.

The Mach operating system, a research project funded by the U.S. Defense Advanced Research Projects Agency at Carnegie Mellon University from 1983 through 1992, was based on innovative approaches to virtual memory management and microkernel architecture. Mach's influence reflects both significant commercial acceptance and substantial contributions to the concept of operating systems.

The Mach kernel forms the heart of the Apple iOS and OS X systems. Mach was at the core of NeXT's operating system, which Apple acquired and subsequently used as the basis of OS X and iOS. Mach's influence can also be traced to operating systems such as GNU Hurd, and UNIX systems OSF/1, Digital Unix, and Tru64 Unix.

Rashid, who graduated from Stanford University with degrees in mathematics and comparative literature and earned M.S. and Ph.D. degrees in computer science from the University of Rochester, founded Microsoft Research in 1991. Today, he is vice president and chief research officer of Microsoft's Applications and Services Group.

Tevanian, who holds a B.A. in mathematics from the University of Rochester and earned M.S. and Ph.D. degrees in computer science from Carnegie Mellon University, spent nearly 10 years at Apple and was a member of the company's leadership team, before becoming managing director of Elevation Partners, a private equity firm.

The ACM Software System Award honors an institution or individual for developing a software system that has had a lasting influence, reflected in contributions to concepts, in commercial acceptance, or both. The award is accompanied by a prize of $35,000. Financial support for the award is provided by IBM.

Distinguished Service, Outstanding Contribution to ACM

Jeannette Wing of Microsoft Research has been named recipient of the 2014 ACM Distinguished Service Award, and **Dame Wendy Hall** of the University of Southampton has been selected to receive the 2014 Outstanding Contribution to ACM Award.

Wing was cited for her advocacy of "computational thinking," which can be used to algorithmically solve complicated problems of scale; her leadership of the Computer and Information Science and Engineering (CISE) Directorate of the U.S. National Science Foundation (NSF), "and for drawing new and diverse audiences to the field of computer science."

As corporate vice president of Microsoft Research, Wing oversees its core research laboratories around the world. Previously, she led the CISE Directorate of the NSF, and was the President's Professor of Computer Science at Carnegie Mellon University.

Wing earned S.B. and S.M. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), where she also received a Ph.D. in computer science.

The Distinguished Service Award is presented based on the value and degree of services to the computing community, including consideration of activities outside ACM and emphasizing contributions to the computing community at large.

The first ACM president from outside North America, Hall was cited "for guiding ACM to become a truly international organization, helping improve diversity within ACM, and working to increase ACM's visibility in scientific venues worldwide."

Hall initiated the establishment of ACM Councils in Europe, India, and China, and has focused on the education of upcoming computer science generations, promoting gender diversity and nurturing talent in computing from all corners of the world. A professor of computer science at the University of Southampton, U.K., Hall was a founding director of the Web Science Research Initiative to promote the discipline of Web Science and foster research collaboration between the University of Southampton and MIT.

A native of London, Hall received B.Sc. and Ph.D. degrees in mathematics from the University of Southamp-

PHOTOS COURTESY OF MICROSOFT RESEARCH (RICK RASHID, JEANNETTE WING), APPLE (AVIE TEVANIAN), BBC (WENDY HALL)

ton, and a Master of Science degree in computing from City College London. She has served as president of the British Computer Society, and since 2014, has served as a commissioner for the Global Commission on Internet Governance. In 2009, she was appointed Dame Commander of the Order of the British Empire.

Outstanding Contribution to ACM Award recipients are selected based on the value and degree of their service to ACM.

"The computing field has benefited immensely from the passion and energy of these two world-leading computer scientists," said ACM President Alexander L. Wolf. "Wing recognized early on that the ever-increasing impact of computing in modern society must be accompanied by a language in which we can discuss computing's intellectual underpinnings with those not versed in its technical complexities.

"Hall provided leadership and inspiration at a time when the computing discipline exploded onto the international scene, promoting ACM as the foremost association of computing professionals worldwide."

Doctoral Dissertation

Matei Zaharia has been selected to receive ACM's 2014 Doctoral Dissertation Award for his innovative solution to tackling the surge in data processing workloads and accommodating the speed and sophistication of complex multistage applications and more interactive ad hoc queries. His work proposed a new architecture for cluster computing systems, achieving best-in-class performance in a variety of workloads while providing a simple programming model that lets users easily and efficiently combine them.

To address the limited processing capabilities of single machines in an age of growing data volumes and stalling process speeds, Zaharia developed Resilient Distributed Datasets (RDDs). As described in his dissertation "An Architecture for Fast and General Data Processing on Large Clusters" (http://bit.ly/1ENKA9e), RDDs are a distributed memory abstraction that lets programmers perform computations on large clusters in a fault-tolerant manner. Zaharia implements RDDs in the open source Apache Spark system, which matches or exceeds the performance of specialized systems in many application domains, achieving speeds up to 100 times faster for certain applications. It also offers stronger fault tolerance guarantees and allows these workloads to be combined.

An assistant professor in the Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, Zaharia completed his dissertation at the University of California, Berkeley, which nominated him. Zaharia received a Bachelor of Mathematics degree from the University of Waterloo, where he won a gold medal at the ACM International Collegiate Programming Contest in 2005. He is a co-founder and chief technology officer of Databricks, the company commercializing Apache Spark.

ACM's annual Doctoral Dissertation Award comes with a $20,000 prize. Financial sponsorship of the award is provided by Google Inc.

Honorable Mention for the 2014 ACM Doctoral Dissertation Award went to **John Criswell** of the University of Rochester, and **John C. Duchi** of Stanford University, who share a $10,000 prize, with financial sponsorship provided by Google Inc.

Criswell's dissertation, "Secure Virtual Architecture: Security for Commodity Software Systems" (http://bit.ly/1IjTPxn), describes a compiler-based infrastructure designed to address the challenges of securing systems that use commodity operating systems like UNIX or Linux.

Duchi's dissertation, "Multiple Optimality Guarantees in Statistical Learning" (http://bit.ly/1DI4L2C), explores trade-offs that occur in modern statistical and machine learning applications. C

Lawrence M. Fisher is Senior Editor/News for ACM Magazines.

PHOTOS COURTESY OF MASSACHUSETTS INSTITUTE OF TECHNOLOGY (MATEI ZAHARIA); STANFORD UNIVERSITY (JOHN DUCHI); JOHN CRISWELL

Milestones

NAS Elects 5 Computer Scientists

Five computer scientists were among those recently elected to the National Academy of Sciences (NAS) in recognition of distinguished, continuing achievements in original research.

The new members are:

- **Robert E. Kahn, president/CEO of the Corporation for National Research Initiatives, Reston, VA;**
- **Jitendra Malik, Arthur J. Chick Professor of Electrical Engineering and Computer Sciences at the University of California, Berkeley; and**
- **Moshe Y. Vardi, Karen Ostrum George Distinguished Service Professor in Computational Engineering at Rice University, Houston, TX, and editor-in-chief of *Communications*.**

Joining the ranks of NAS foreign associates were:

- **Manindra Agrawal, N. Rama Rao Chair Professor in the department of computer science and engineering of the Indian Institute of Technology, Kanpur, India; and**
- **Kurt Mehlhorn, director of the Max Planck Institute for Informatics at Saarbrücken, Germany.**

IBM'S GENTRY NAMED MACARTHUR FELLOW

Among the newest Fellows named by the MacArthur Foundation is Craig Gentry, a research scientist in the Cryptography Research Group at IBM Thomas J. Watson Research Center in Yorktown Heights, NY, recognized for "fueling a revolution in cryptography and theoretical computer science through his elegant solutions to some of the discipline's most challenging open problems."

Gentry published a plausible candidate construction of a fully homomorphic encryption scheme, making it possible to perform arbitrary computations on data while encrypted and without needing a secret key.

In addition, Gentry and his colleagues published a plausible candidate multilinear map, and used those findings to present the first example of cryptographic software obfuscation.

DOI:10.1145/2770890

Pamela Samuelson

Legally Speaking
Anti-Circumvention Rules Limit Reverse Engineering

Considering some of the requested exceptions to technical protection mechanisms.

Until the U.S. Congress passed the Digital Millennium Copyright Act (DMCA) in 1998, reverse engineering of computer programs and other digital works was widely regarded as lawful in the U.S. The DMCA changed the law because the entertainment industry feared clever hackers could and would bypass technical protection measures (TPMs) that the industry planned to use to protect their copyrighted works from unauthorized copying and dissemination. The industry persuaded Congress to make it illegal to circumvent TPMs and to make or offer circumvention tools to the public.

Circumvention of TPMs is, of course, a form of reverse engineering. This activity is now illegal not only in the U.S., but also in most of the rest of the world unless there is a special exception that permits circumvention-reverse engineering for specific purposes under specific conditions. The DMCA rules, for instance, include exceptions for law enforcement, intelligence, and national security purposes, for making software interoperable, and for encryption and computer security research under certain conditions.

In response to expressions of concern that the anti-circumvention rules might have detrimental effects on the ability to make fair and otherwise lawful uses of technically protected digital content, Congress created a triennial rulemaking process that enables affected persons to request special exceptions to the anti-circumvention rules to engage in specified legitimate activities that TPMs are thwarting.

In November 2014, the Copyright Office received more than 40 proposals for special exceptions to the DMCA anti-circumvention rules. In February 2015, the Office received detailed comments explaining the rationales for the proposed exceptions. In March, opponents had the opportunity to express their objections to the proposed exceptions. In late May, the Office held hearings to allow proponents to offer further arguments in support of proposed exemptions and opponents to rebut those arguments. Thereafter the Copyright Office will review the record, hold some hearings, and ultimately issue rules that will either grant or deny the requested exceptions. This column provides an overview of the requested exceptions and delves into some proposals that may be of interest to computing professionals.

Overview of Submissions

Approximately half of the proposed exceptions aim to enable interoperability with devices or software that the anti-circumvention rules arguably makes illegal. Some submissions argue for exceptions to allow bypassing TPMs for purposes of repair and modification of software in vehicles. A few ask for broader exceptions for computer security research purposes.

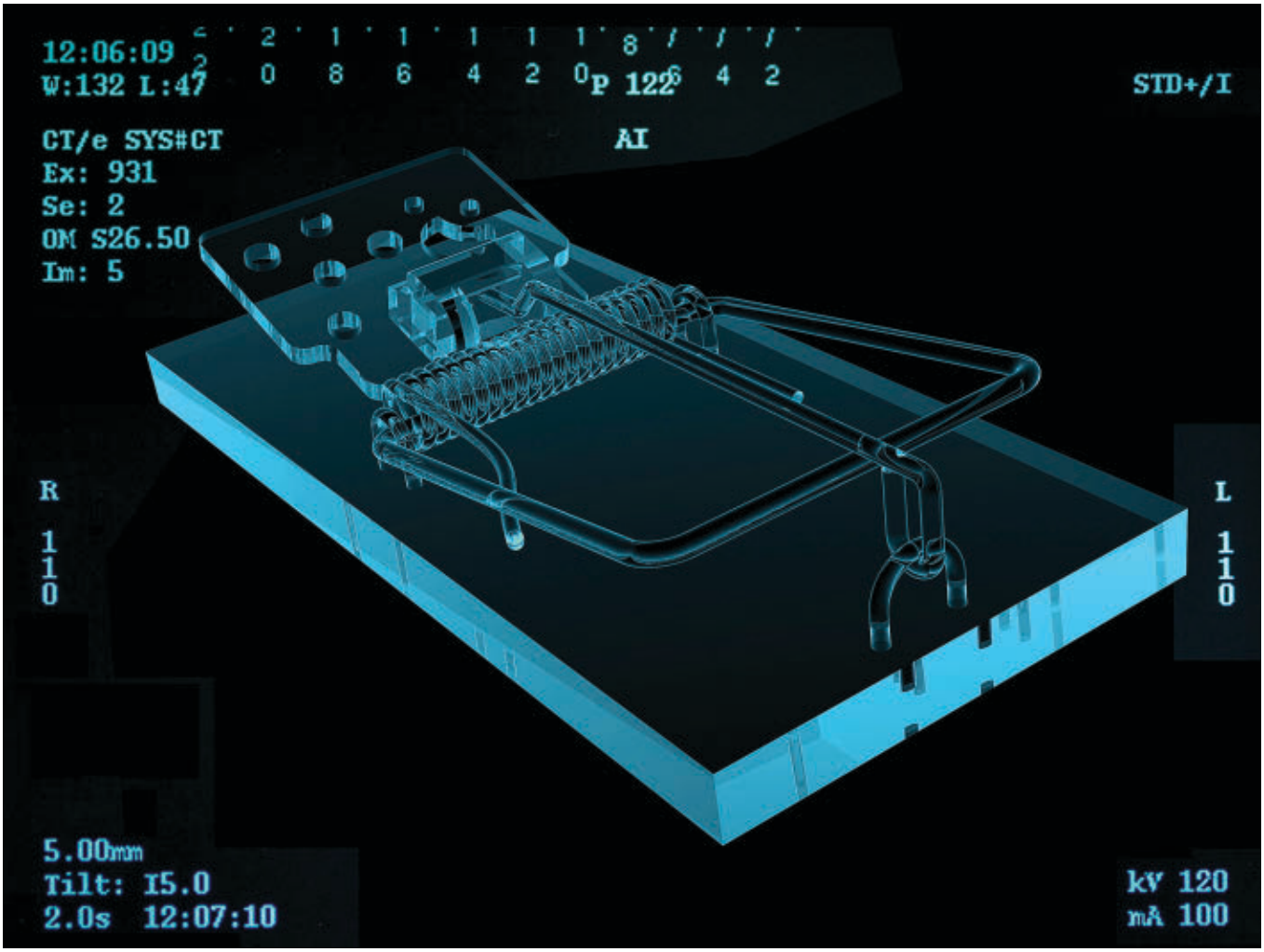


IMAGE BY X-RAY PICTURES

Several proposed exceptions aim to overcome impediments the anti-circumvention rules pose for creating multimedia e-books, other educational materials, documentary films, and remixes of technically protected works. Two submissions request exceptions for bypassing TPMs to provide assistive technologies for print-disabled persons so they can, for example, have access to digital books in alternative formats.

One submission asks for an exception to enable consumers to be able to continue to use videogames they have purchased after the games' makers have stopped providing support for the games. Another submission seeks to enable space-shifting of DVD movies. Two others want to make broader personal uses of technically protected works. All submissions for this year's triennial review can be found at http://copyright.gov/1201.

Missing from the triennial review in 2014–2015 is a proposed exception to allow bypassing of TPMs to "unlock" cellphones so their owners can access alternative wireless networks. Even though the Copyright Office denied a requested exception to enable this activity in the last triennial review, Congress passed a special law in 2014 that granted an exception for this legitimate activity, a sensible result given that cellphone unlocking poses no threat of copyright infringement.

Interoperability

Because the DMCA rules have an interoperability exception, it may seem puzzling that so many of the proposed exceptions to the anti-circumvention rules address interoperability issues. The existing exception permits reverse engineering of technically protected software "for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs." The information obtained thereby can only be used or disseminated to others for interoperability purposes.

Does that exception permit circumvention for purposes of enabling consumers to use computer tablets or wearable computing devices to access alternative wireless networks or to access mobile hotspots? The Rural Wireless Association fears it does not, so it is seeking exemptions for these kinds of activities. Unfortunately, the cellphone unlocking exception passed by Congress does not extend to these devices. Yet, these uses would seem to pose no threat of copyright infringement to justify outlawing this type of circumvention of TPMs.

Another interoperability exception being sought is Public Knowledge's effort to enable bypassing of TPMs that makers of 3D-printing devices have embedded in their software to stop unauthorized firms from competing in the supply of feedstock to owners of their 3D printers. Competition policy

would seem to support the grant of this exception, which also poses no threat of copyright infringement.

Two other interoperability exceptions focus on bypassing TPMs to enable consumers to have more choices on the applications that can run on their devices. One submission asks for an exception so owners of Linux operating system computers can watch lawfully purchased DVD movies. Another submission requests an exception so owners of videogame consoles can bypass TPMs that limit the applications that can run on those consoles.

Computer Security

Computer researchers Steve Bellovin, Matt Blaze, Ed Felten, Alex Halderman, and Nadia Heninger submitted a request for a computer security testing exception that would permit bypassing TPMs to access computer software and databases embodied in various technologies to test for vulnerabilities, malfunctions, and flaws.

Among the types of software systems in devices these researchers envision testing are: insulin pumps, pacemakers, car components (including braking and acceleration systems), controls for nuclear power plants, smart grids, and transit systems, as well as smart technologies for the home. These researchers argue that such systems are very important for the health and safety of their users and of the public at large. Malfunctions, flaws, and vulnerabilities may cause considerable harms to individuals and to the public, so good faith testing is a public good. It too poses no threat of infringement, which was the principal justification for adoption of the anti-circumvention rules in the first place.

There is an existing computer security exception in the DMCA, but it requires advance permission of the owner of the computing system being tested and seems to limit the dissemination of results of security testing to the owner of that computing system.

The Bellovin submission wants computer security researchers to be able to test vulnerabilities without getting advance permission. The researchers also want to be able to disseminate their research results in responsible ways, such as by presentation of research results at conferences and in journal publications. The DMCA rules now contemplate that a copyright owner in technically protected software could enjoin dissemination of research results. This has had a chilling effect on the research that can be done to test the security of a wide variety of computing systems.

Congress should have adopted narrower anti-circumvention rules in the first place.

What Will the Office Do?

If the past is any predictor of the future, chances are quite high the Copyright Office will eventually deny the overwhelming majority of the requested anti-circumvention exceptions, no matter how harmless they might seem.

Some proposals will likely be rejected because the Office believes proponents failed to prove that TPMs are actually an impediment to lawful uses of copyrighted works; it is not enough to assert that TPMs might impede legitimate activities.

Some proposals may be denied because the Office perceives the requested exception will enable infringing uses. Eldridge Alexander, for instance, is unlikely to get an exception so he can bypass CSS to create a software library of his DVD movies because bypassing CSS would also enable infringing uses of the movies.

The Office may dismiss some requested exceptions as unnecessary because it perceives there are other ways to achieve the stated objective (for example, video capture of images from movies for educational or critical uses rather than bypassing the TPMs).

Even those exceptions the Office grants may be more restrictive as granted than as requested. During the last triennial review, for example, the Office was willing to grant an exception for film studies professors to bypass CSS to show clips from movies to illustrate filmmaking techniques. However, the Office did not recognize that many other types of instructors could benefit from an exception that enabled them to make fair use clips of movies to illustrate other types of lessons.

During the current triennial review, the Authors Alliance (of which I am a co-founder) has proposed exception for multimedia e-books that would, for instance, enable me to show clips from various James Bond movies so that my students could consider whether James Bond is an "idea" or an "expression" under copyright law, an issue that has been litigated in some U.S. cases.

Will the Office recognize the validity of the interoperability and computer security testing exceptions being sought? One can certainly hope so. However, without a team of technologists to analyze the submissions and advise the Office about the exception proposals, there is reason to worry the Office will regard these exceptions skeptically, especially if entertainment industry groups oppose them as they have in the past.

Conclusion

Congress should have adopted narrower anti-circumvention rules in the first place. Only circumventions that facilitate copyright infringement should be illegal. This would obviate the need for a triennial review process, and make reverse engineering of digital works far less risky than it is today.

Over time, the anti-circumvention rules may perhaps be amended so that computer security and interoperability interests are better protected than they are now. Yet until that day comes, we should be grateful the triennial review process exists to provide a mechanism by which computing professionals, among others, can make the case for reverse engineering as a legitimate activity that serves the public interests in competition, ongoing innovation, and public health and safety. C

Pamela Samuelson (pam@law.berkeley.edu) is the Richard M. Sherman Distinguished Professor of Law and Information at the University of California, Berkeley.

DOI:10.1145/2770892

L. Jean Camp

Computing Ethics
Respecting People and Respecting Privacy

Minimizing data collection to protect user privacy and increase security.

People conflate privacy and security. For many there are no trade-offs, choices, or decisions: guarantee security and you guarantee privacy. But computer designers know it is more complicated than that. This column argues that starting with respect for people who desire privacy will help guide good security design. For example, to help mitigate the security threat of identity theft one wants to consider the loss of private information.

Good design practice is a responsibility. The ACM Code of Ethics requires that designers "respect the privacy of others" and provides two paragraphs of best practice. Many users are both busy and insufficiently proficient technically to watch out for themselves. They understand neither technical minutiae nor the basics of privacy. A recent survey asked about Internet public key certificates (PKI) certificates. Most people said they did not know what such certificates are, or that PKI certificates provide more protection than they do. Many thought PKI certificates ensure privacy, prevent tracking, provide legal accountability, or certify that a site is protected from "hackers."[2] They confuse security goals and privacy, and seldom understand related risks well.

The survey included a range of respondents: shoppers at the Bloomington Farmers Market, people who attend Indiana University's "Mini University" (participants are active retirees who can be vulnerable and need to protect financial assets); attendees at the Dashcon convention for Tumblr enthusiasts (predominantly young people familiar with the Internet); and college students from psychology and computer science who are typically "digital natives." Privacy and security practices were not well understood by intelligent, educated, and cognitively flexible people, although those with significant expertise came closer to understanding the purpose of certificates.

Mobile device permissions can be even more confusing.[3,4] Few people understand the routine level of geographic tracking (pinpointing the device's location). Intensive tracking might betray a pathological desire for data collection, but it might also be a result of the easiest default to set. Some designers have difficulty mastering permissions, and extensive data collection can be the consequence. I argue for data minimization, meaning collecting the least data needed to help ensure privacy and security. Failure to minimize data collection can be an ethical issue, but many developers

IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

fail to grasp its importance. Here, I offer three reasons to seek data minimization to protect privacy, and thereby security.

First, collecting and transmitting data exacerbates the risk of data exfiltration, a leading risk in security attacks. Attackers can subvert a computer or mobile phone to acquire credentials. An outward-facing mobile phone camera can allow reconstruction of sensitive office space. Attackers can examine an account before deciding how to exploit it.[1] Traditional email attacks (for example, the stranded traveler[a]) can be used on phone-based clients, and URLs for malware diffusion can be sent by short text messages (SMS), email, and other phone-based mechanisms. SMS is already used to obtain funds directly through per-charge messages. Reducing the amount of data that can be exfiltrated reduces both privacy and security risks.

Second, data minimization can reduce business risk. Flashlight apps that exfiltrate phone information to support advertisers can exfiltrate more data than needed. Too much exfiltration led to the removal of flashlight apps as iPhones and Android included free integrated flashlights. Exfiltration can lead to short-term profit but long-term losses to the companies involved. Apps can destroy markets when they weaken privacy controls. The app Girls Around Me used Foursquare data to display Facebook profiles of women near a location, including pictures. The app was making money but Foursquare blocked it because violating privacy lost users for Foursquare. An immensely popular and valuable application became a liability. Superfish on Lenovo computers hurt Lenovo; what Lenovo considered advertising others considered malware.

Data minimization can also reduce legal problems. Assistant U.S. Attorney General Leslie Caldwell of the Justice Department's Criminal Division said, "Selling spyware is not just reprehensible, it's a crime." Superfish on Lenovo computers is being investigated as wiretapping. The successful app StealthGenie has been considered a kind of spyware, and its CEO is now under indictment. Thinking in advance about privacy can help both designers and users.

Thinking in advance about privacy can help both designers and users.

Why do people accept privacy-violating products in the first place? Perhaps they do not care much about privacy or do not understand the implications. Or they care but accept the convenience in exchange for the risk of privacy violation. Perhaps they cannot use the tools intended to provide protection, so designers think users are protected but in fact they are not. There are good economic reasons to support those people who care about privacy. A market with *perfect information* enables price discrimination in which each person pays the amount he or she is willing to pay, no less and no more. Some customers will pay money for privacy, while some will pay in possible loss of privacy to avoid spending money. But the presence or lack of protections for privacy should be clear.

When people do not understand the privacy risks of their own choices, there is not only a business process failure, but also an ethical one. Usability analysis should indicate whether tools are effective for people trying to protect themselves. People can be bad at risk analysis, so care must be taken to help them understand risks. People might choose to take risks online for the same reasons they choose risks offline. Of course, designs cannot force people to make careful risk decisions. But designs often stop far short of that point. Lack of information about privacy risk can lead to consumer regret and unwillingness to reinvest. Or users might refuse products that carry too much risk to privacy, resulting in an untapped market. Users expect computer designers to follow the code of ethics laid down about privacy. To the extent that participation in the network is a form of enfranchisement, poor design for privacy is a kind of disenfranchisement.

Privacy risks and loss should not be invisible. If someone trades privacy for pricing, that trade should be abundantly clear. People do not expect their televisions to listen to every word in the house. They want transparency and risk communication. Some users will ignore a manufacturer that includes hidden surveillance capability while others will be furious. Transparency is expected by the U.S. Federal Trade Commission and is common in many markets. The market for computing devices, from computers to mobile phones, should be one such market. This does not mean everyone has the same expectations of privacy. It merely means that customers should know about privacy risks, and be able to handle those risks as they see fit. Choices about privacy and security are important. The choices that designers provide should help surgeons, air traffic controllers, and other highly skilled individuals who have responsibilities for the security and privacy of others—and those less highly skilled (and less empowered as well), make sensible decisions about their privacy and security needs. When security and privacy technologies fail, those with the knowledge, role, and skills put them in the best position to prevent the failures bear much of the responsibility. C

a The "stranded traveler" scam uses a subverted account to send a message to all account contacts asking for emergency help. The attacker pretends to be the account owner, claims to be desperately stranded, and requests money be sent immediately to a specific (attacker-owned) bank account.

References

1. Bursztein, E. et al. Handcrafted fraud and extortion: Manual account hijacking in the wild. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, ACM, 2014.
2. Camp, L.J., Kelley, T., and Rajivan, P. Instrument for Measuring Computing and Security Expertise–TR715. Indiana University, Department of Computer Science Technical Report Series; http://www.cs.indiana.edu/cgi-bin/techreports/TRNNN.cgi?trnum=TR715.
3. Felt, A.P. et al. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, ACM, 2012.
4. Kelley, P.G. et al. A conundrum of permissions: Installing applications on an Android smartphone. *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2012, 68–79.

L. Jean Camp (ljeanc@gmail.com) is a professor of informatics at Indiana University Bloomington.

DOI:10.1145/2770894

David Anderson

Historical Reflections

Preserving the Digital Record of Computing History

Reflecting on the complexities associated with maintaining rapidly changing information technology.

IMAGE BY WIKTORIA PAWLAK

Increasingly, the source materials on which historians of computing work are either born digital or have been digitized. How these digital treasures are preserved, and how continuing access to them may be secured over the long term are therefore matters of concern to us. One of the key determinants of whether this material will remain accessible is the file format in which the source materials are stored. Choosing the "wrong" format will have significant implications for the extent to which files might be supported by systems, automated tools, or workflow associated with the digital content life cycle processes in libraries, archives, and other digital repositories. For this reason, digital preservationists have always taken a keen interest in file formats, and are understandably motivated to maintain up-to-date information on the characteristics of different formats, and the effect these have on their preservability over time.

A considerable amount of digital material is now subject to mandatory deposit to national libraries, meaning even data creators who have no interest in preservation or long-term access need to develop some understanding of the formats preferred, or deemed acceptable by repositories and why.

The principal annual gathering of digital preservationists takes place at the iPRES conference, the most recent of which was held in Melbourne, Australia last year; the next conference will be held this November in Chapel Hill, N.C.[a] The iPRES conference always provides an opportunity to gauge the zeitgeist of the field, and last year it was difficult not to be struck by the amount of attention being paid to the difficulties posed by file format issues, and the extent to which a viable format registry could be created, which both responds to the needs of the preservation community and involves practitioners meaningfully going forward. Format was prominent both in the pa-

a See http://www.digitalmeetsculture.net/article/international-conference-on-digital-preservation-ipres-2015/.

pers presented, and in the break-time conversations, some of which were quite animated. The problems caused by format obsolescence and interoperability affect all of us. Whether we are scientists wanting to return to data generated by a specialized scientific instrument that no longer sits in the lab, or animators working on Hollywood blockbusters whose work must be done from scratch many times in a single movie because the studio wants to make use of features in the latest piece of bleeding-edge software. At a more mundane level, files created in version 1.0 of our favorite word processor are seldom fully compatible when we upgrade to later versions. Format obsolescence has the capacity to cut us off from the fruits of research results whether publicly funded, or developed by companies. Files, even when preserved perfectly, are useless if we do not have software that can make sense of them. The pace at which technology drives forward, and the lure of the new, means it is ever more difficult to ensure legacy files are kept fully accessible.

One of the complaints that came up was that "format," despite being a term in common use, is, in fact, not well understood or agreed upon even within the "format" community. Specific problems arising as the result of particular "formats" include:

- Not having publicly available specifications;
- Having implementations that do not comply with published specifications;
- Having format specifications, or "usual" implementations changed over time, and between different vendors, users, and others; and
- Producing a comprehensive mapping of possible formats for files (of any reasonable length) so as to be able to identify files of unknown format.

These, and similar considerations give rise to a certain degree of consternation, and a general lack of confidence concerning the likelihood of succeeding in 'imposing' much by way of a single 'understanding' of format in general, or even a given individual format.

A common approach to identifying files, the format of which is unknown, is to examine the individual bytes (the bitstream) looking for characteristic patterns or 'signatures' associated with known formats. Tools such as DROID (Digital Record and Object Identification), have been developed to make this process easier, and form an important part of the digital forensics toolkit. Perhaps, therefore, we might avoid confusion over "formats" by concentrating our attention on the identifying characteristics of bitstreams. Indeed, there appeared to be no dissent from the view that file formats can be thought of as (usually nested) interpretations (or encoding schemes) of bitstreams. However tempting such linguistic legerdemain may be, there are at least two grounds for thinking this is not the right approach.

Files, even when preserved perfectly, are useless if we do not have software that can make sense of them.

Abstraction is the process of establishing the level of complexity on which people interact with systems. For example, when we are writing code involving numerical operations we normally ignore the way in which numbers are represented in the underlying computer hardware (for example, 16-bit or 32-bit), concentrating entirely on "number." On other occasions, these implementation (or lower-level) details are critical and we pay close attention to each "bit."

Changing our level of abstraction down (for example, from "format" to "bitstream") is a certainly recognized way of getting around talking about difficult, disputed, or otherwise problematic concepts, similarly changing a level of abstraction (up) is often very helpful in introducing illuminating concepts or organizational principles that not only are not apparent at a lower level but simply do not apply. The restricted applicability of concepts (they are tied to the level of abstraction—and in some sense define it) is often easier for us to notice in one direction of travel than the other.

For example, when speaking of "color depth" we are talking how finely levels of color can be expressed in a given "format" (encoding scheme). It is not possible to talk meaningfully about color depth at the level of individual bits, or at bitstream level. Color depth is a "higher"-level concept than it is possible to express at the level of bits, it requires an abstraction to a level where encoding schemes exist. This does not preclude us talking at the higher level *about* bits of course, and indeed, that is how color depth is normally discussed—that is, how many bits a given encoding scheme devotes to representing the color of a particular pixel. "Color," of course, is a higher-level concept again. There are no red bits, or green bitstreams. Talk of "color" does not belong there. It is important to have a clear sense of what concepts apply at each level of description if we are to avoid making "category mistakes."[b] It is difficult to see how, if we were to restrict ourselves to discussing bitstreams alone we would continue to have at our disposal the set of concepts, such as "color depth," that we require routinely.

A second set of difficulties flow from the disturbingly large problem space that results from understanding formats to be identifiable ways of encoding computer files. For example, a file of bit length L, where each bit may take 1 of V values may be potentially encoded in VL ways. So, a binary file, 8-bits long, is capable of being represented in 256 distinguishable ways and could, therefore give rise to 256 different formats. A 64-bit file can support 18,446,744,073,709,600,000 different representations. Even if we exclude the notion of files that are all format and no "payload" the problem space is not significantly reduced. The smallest "payload" the system I have just described can support is a single "payload" bit, the remainder of the file being taken up with encoding the format. Thus, a binary file 8 bits long (with a 1-bit "payload") can be represented in 128 distinguishable ways and could, therefore give rise to 128 different formats and there could be exactly two different "payloads" representable in this scheme for each of the possible formats. Our 64-bit file (with a

b http://en.wikipedia.org/wiki/Category_mistake

1-bit "payload") can similarly support 9,223,372,036,854,780,000 different representations/formats, any of which may represent one of two payloads. Suffice it to say the numbers involved are very large, so large in fact, that if we insist on trying to wrestle with the whole theoretical problem space, expressed purely in terms of bitstreams, we are not likely to make much progress.

Some other limitations of bitstreams are also apparent. Files having exactly similarly bitmap sequences (syntax) need not have the same format (semantics). Formats are encodings, and different vendors can impose different "meaning" on the same patterns. Syntax is not the same as semantics. It is simply not possible, in the abstract, to distinguish between a given 6-bit pattern followed by 2 bits of "payload" and a 7-bit pattern (coincidentally sharing the same initial 6 bits) followed by a 1-bit "payload." How we interpret (encode) these 8 bits is a matter of choice, which is apt to change over time, place, and circumstance.

It is clear that if, for example, we aim to have a comprehensive mapping of possible formats for files (of any reasonable length), basing our work on bitstreams is not going to succeed. Abandoning talk of formats in favor of bitstream language, for all its intellectual/mathematical attractiveness, will not solve the problems with which we are faced or with which we should be attempting to grapple. I am therefore disinclined to go down the bitstream route.

Some of the problems we need to be addressing include:

- Categorization of formats;
- Modeling format; and
- Identifying the format of files whose format is currently unknown.

There is every reason to be hopeful that substantial progress can be made against each of these problems.

In the biological sciences (which represents a much more complex and multifaceted domain than computer file formats), broadly similar kinds of problems:

- Categorization of life-forms;
- Modeling life-forms; and
- Identifying the genus and species (and other characteristics) of life-forms whose identity is currently unknown.

These problems have been substantially solved and a movement now exists within the biodiversity informatics community to provide globally unique identifiers in the form of Life Science Identifiers (LSID) for all biological names. Three large nomenclatural databases have already begun this process: Index Fungorum, International Plant Names Index (IPNI), and ZooBank. Other databases, which publish taxonomic rather than nomenclatural data, have also started using LSIDs to identify taxa.

The solution did not, in the biological sciences, involve abandoning talk of genus or species in favor of DNA sequences (for example) but was built on a broadly Linnaean taxonomic approach. This is the sort of approach that might be employed in our more modest and easily tractable domain.

Having decided which attributes of files are of interest (and of course there will be many, and the list will change over time) we can begin to group these into different categories. This is broadly analogous to the Life, Domain, Kingdom, Phylum, Class, Order, Family, Genus, Species taxonomy that is familiar in biology. The biological approach has been perfectly able to withstand heated scientific dispute of the correct classification of individual life-forms, and it is by no means uncommon to see, for example, an insect being classified differently over time, as scientific understanding and technique have changed or developed. It is not necessary for us to get classification schemes right the first time, or for them to be frozen for all time, in order for them to gain widespread acceptance, and to confer significant benefits on the community.

By declining to speak of formats, in favor of speaking of bitstreams, we will not only fail to improve matters but may actually make intractable the format problems we need to tackle. Insofar as we do want to employ multiple levels of abstraction in our discussions of the overall problem space, we must be careful which concepts we deploy where, or we are likely to make category mistakes. C

David Anderson (cdpa@btinternet.com) is the CiTECH Research Centre Director at the School of Creative Technologies, University of Portsmouth, U.K.

DOI:10.1145/2770896

Phillip G. Armour

The Business of Software
An Updated Software Almanac

Research into what makes software projects succeed.

Software projects can be so complicated and so different from each other that predicting whether they will succeed or fail can be as difficult as forecasting the weather or picking winning stocks. Will the project entirely fulfill its goals? Will it deliver some value at a higher cost or later than desired? Or will it just crash and burn leaving the exhausted survivors to lick their wounds, bury the dead bodies, and shred the evidence?

Courageous efforts are being made to collect and codify the data that is available, to try to spot what trends are occurring in the industry, and to provide some useful guidelines for managing the business of software. The recently published *QSM Software Almanac*, dubbed the "2014 Research Edition," is an example of this.

QSM Software Almanac

Quantitative Software Management (QSM) published the IT Metrics Edition of its Software Almanac in 2006.[1] This highly detailed analysis of thousands of project data points was interesting reading and reached a few quite jaw-dropping conclusions. The 2014 version confirms much of the 2006 analysis and provides further insights. It begins by suggesting how software projects and organizations should be measured and what we can infer from these measurements.

Five Core Metrics

In software projects and organizations, the list of things-that-could-be-measured is daunting. The intricacy of projects, the differences in the types of systems being built, in project life cycles, in toolsets, in expertise and development environments all add to the variables operating in our business. But some simple core metrics, available from all projects and enterprises, can be as important as the measurement of temperature and blood pressure are to medical diagnosis. They may not diagnose all ailments, but they can give provide insights into the health of projects and organizations.

These metrics[2] are:

Schedule—the elapsed calendar time from a project start until completion of its mandate;

Effort or cost—which is usually driven by the number of staff employed during the project duration;

Functionality delivered or "system size"—this is the *value component* of the project, what is obtained for the cost and schedule expended;

Quality or defect level—the "anti-

IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

size" or that portion of the system that does not work properly on delivery; and

"Productivity"—the rate at which the resources of time, effort, and staff are turned into delivered functionality (minus defects).

The *QSM Software Almanac* uses these relatively simple metrics, backed up with more sophisticated measurements, to make some important points about modern systems development:

Projects seldom measure *performance*. Most projects in QSM's database measure and report schedule, effort, and size, but only one-third of the projects actually use this data to assess their performance. This is a missed opportunity for improvement.

Sacrifice schedule first. Projects are willing to overrun schedules more than budget and are more willing to overrun budget than deliver less functionality to customers.

Big teams are bad. In this study we see repeatedly that larger teams cost a lot more, deliver lower-quality systems, but seldom save time. In one quoted example, cutting project staff from 100 people to 52 people saved $12 million but resulted in only a modest increase in schedule. Worst-in-class projects, as measured by poor performance and low quality, are strongly correlated with large team size and best-in-class projects are strongly correlated with small team sizes. In addition, large-staffed projects tend to try to fix any problems they encounter by adding even more staff whereas small-staffed projects usually try other, more effective, approaches. QSM's data also showed a wide range of staffing for similarly sized systems; this may be driven by the common practice of throwing people at projects in trouble. Smaller-staffed projects showed between three and eight times greater productivity than larger staffed projects delivering equivalent value(!).

Quality is getting better over time. This is especially true for engineering systems, which are getting larger while IT systems are getting smaller and taking longer. This was shown for best-in-class projects; worst-in-class projects seldom collect quality data, but probably have worse quality.

Reuse does not usually work. "Minor enhancement" projects with high levels of expected reuse typically cost twice that of equivalent new development.

Agile advantage tops out around 30K line-of-code equivalents. The data shows that, at their present stage of evolution, Agile methods are best suited to smaller projects. This finding seems to support a general industry perception of Agile. But QSM also found that large Agile projects have

Figure 1. Typical engineering project duration.

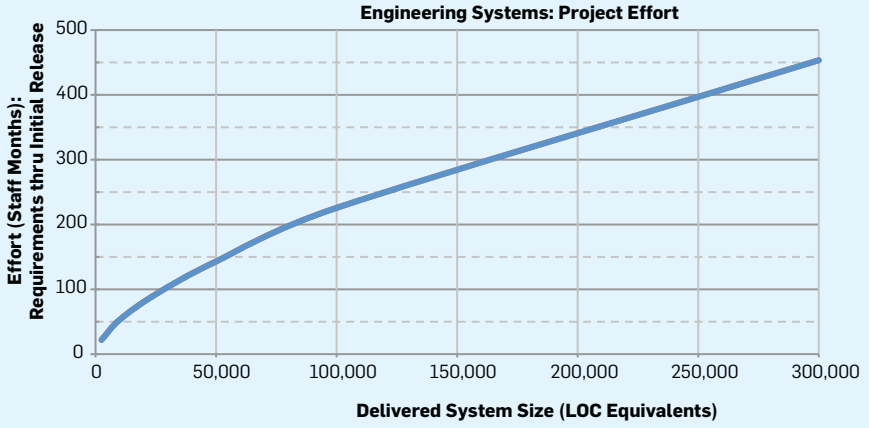


Figure 2. Typical engineering project effort.

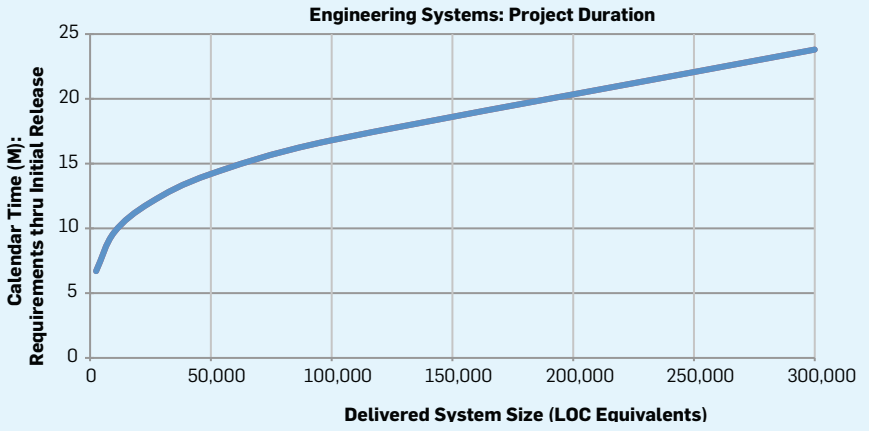
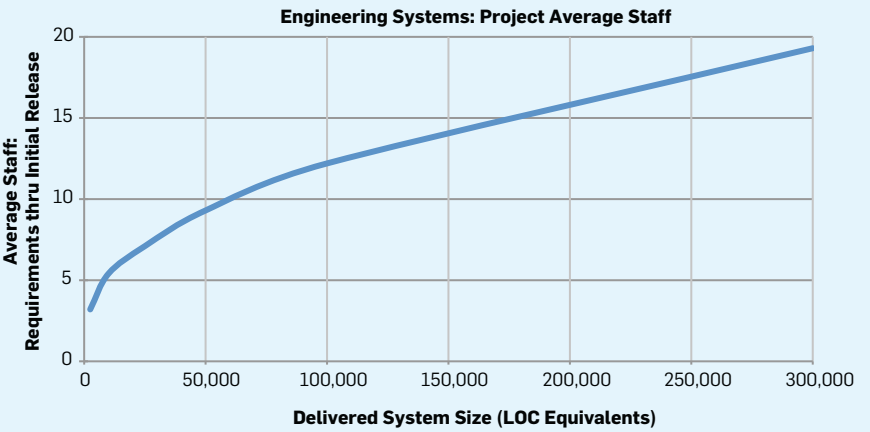


Figure 3. Typical engineering project average staff.



In software projects and organizations, the list of things-that-could-be-measured is daunting.

much more variable results, so it is likely the benefits of larger Agile projects depend a lot on how they are implemented. Also, larger Agile projects that experience problems are better off adding time than dropping functionality or adding staff.

Projects grow. Average functionality-to-be-delivered increases 15% over the life of a project, typically creating schedule overruns of 8% and cost overruns of 16%.

Some Industry Trends Over Time

The Almanac shows:

- The median project duration contraction that occurred during the 1990s and 2000s has leveled off to 10–11 months.
- Project effort is declining along with the delivered size of systems, but delivered size is varying more as time goes by.
- Median project team size is relatively stable.
- There is a strong trend toward more financial systems development in IT.
- Mixed-language development is the norm, but Java has become the most common development language.

Measurement Ideas

Some of the articles in the Almanac describe useful approaches for the measurement of projects. One article describes the use of Shewart control charts (modified for Rayleigh mathematics) to calibrate defect discovery trends and use them for predictive purposes. Another covers four models for data mining that can be used for process improvement. A third article describes an empirical study to see what project and environmental factors most correlate with schedule performance on projects. For business applications these factors are overall technical system complexity and team communication complexity—an important finding for distributed or offshore teams.

Performance Benchmark Tables

The Almanac's final set of tables can be used to determine if your project fits close to typical measured norms for business, engineering, or real-time systems development. Given a reasonable assessment of delivered system size, these tables would allow a project manager to determine typical values for project duration, effort/cost, and average staff in a minute or two, and avoid serious overcommitments in time, money, staff, or delivered functions (see figures 1–3 for typical engineering system ranges).

Lessons Learned

This study indicates some simple lessons:

- Collect the basic and practical core metrics on projects and use them to figure out what you can do, what you are doing, and how well you are doing it.
- Commit projects at levels around measured industry benchmarks in duration, effort, and staff.
- Avoid using large teams—they will probably be ineffective.
- Do not add people to a project that hits snags—it will likely make things worse; instead consider giving the project more time.
- Do not assume reuse will save you anything in time or effort.
- Manage larger Agile projects carefully so the benefits are not nullified by size and complexity.

It's Free

The *QSM Software Almanac 2014 Research Edition* contains a great many more insights and detailed analyses of the copious amount of data collected. It is free for download from QSM at: http://bit.ly/1y5MahV. C

References

1. Armour, P.G. Software, hard data. *Commun. ACM 49*, 9 (Sept. 2006), 15–17.
2. Putnam, L.H. and Myers, W. *Five Core Metrics*. Dorset House Publishing Co., 2003.

Phillip G. Armour (armour@corvusintl.com) is a vice president at Applied Pathways LLC, in Schaumburg, IL, and a senior consultant at Corvus International Inc., in Deer Park, IL.

DOI:10.1145/2770929

Juan E. Gilbert, Jerlando F.L. Jackson, Edward C. Dillon, Jr., and LaVar J. Charleston

Broadening Participation
African Americans in the U.S. Computing Sciences Workforce

An exploration of the education-to-work pipeline.

Broadening participation in computing has received a great deal of media coverage recently on diversity challenges in Silicon Valley.[2,8] Major Silicon Valley technology companies, including Dell and Intel, have released employment data and the lack of diversity has caused many to question their commitment.[2] For example, Intel has committed $300 million over the next five years to improve the company's workforce diversity.[8] As employment data is released for the technology workforce in Silicon Valley and other technology hubs, an important question emerges: How are Ph.D.-granting computing departments doing regarding the representation of African Americans? In this column, we examine efforts to increase Ph.D. and faculty production for African Americans in computing sciences through the work of a new project funded by the National Science Foundation—Institute for African-American Mentoring in Computing Sciences (iAAMCS, pronounced "i am c s"). The data presented in this column is from the Computing Research Association (CRA) Taulbee Survey (see http://www.cra.org/resources/taulbee/)[a] and the NSF Survey of Earned Doctorates (SED) Tabulation Engine (http://ncses.norc.org).

IMAGE COURTESY OF INTEL

A panel discussion at the 2015 International Consumer Electronics Show on the importance of accelerating diversity in the technology industry.

African American Ph.D. and Faculty Production Landscape

Table 1 contains data from 2003 to 2013 on computer sciences (CS) Ph.D. production for African Americans from the CRA Taulbee Survey and the NSF Survey of Earned Doctorates that uses data from the National Center for Science and Engineering Statistics (NCSES). Taylor and Ladner[7] first reported differences in the CRA Taulbee Survey data and the WebCASPAR data. Therefore, this column will use data from both sources with respect to Ph.D. production to show the contrast. Both datasets show an increase in the raw number of African American CS Ph.D.'s produced; however, the total percentage distribution among Ph.D.'s has not

a The CRA Taulbee Survey contains information on the enrollment, production, and employment of Ph.D.'s in computer science and computer engineering (CS and CE) and demographic data for faculty in CS and CE in North America. It also includes data on gender and ethnicity breakdowns. Known limitations of Taulbee Survey data include: reporting universities can vary from year to year and reporting universities are only a subset of all Ph.D.-granting institutions, but generally cover all the top producing institutions.

changed much. There is at least a 50% increase in the raw number of Ph.D.'s produced, but the total percentage distribution remains relatively flat. Clearly, the overall CS Ph.D. production has increased at a rate that limits increases in the percentage. However, does this increase in overall CS Ph.D. production among African Americans result in an increase in African American faculty?

Table 2 captures the African American CS tenure-track faculty from 2003 to 2013 from the CRA Taulbee Survey. Table 2 only counts faculty in Ph.D.-granting departments that participate in the Taulbee survey. Broader surveys of African American faculty and their ranks do not seem to exist. This data shows more improvement compared to the Ph.D. production data. At all ranks, there is more than a 100% increase for African Americans in CS tenure-track faculty positions from 2003 to 2013. These numbers are promising, but there is still much work to be done. The fact that the highest concentration levels for assistant, associate, and full professors has been respectively 3.50%, 1.80%, and 0.80% are very low. There is opportunity for improvement within the academy. So, why are these numbers so low with respect to African American CS tenure-track faculty and Ph.D. production? How can these numbers be improved?

Gibbs et al.[1] conducted a study of 1,500 recent American BMS (biomedical sciences) Ph.D. graduates (including 276 underrepresented minorities) that examined career preferences over the course of their graduate training experiences. Their findings showed a disproportionately low interest among underrepresented minorities and women in pursuing an academic career at a research university upon completing graduate school compared to their White and Asian male counterparts. The article also stated that scientists from underrepresented backgrounds, which included American Indian/Alaska Native, Black/African American, Hispanic/Latino, or Native Hawaiian/Pacific Islander, earn 10% of life science Ph.D.'s and that number had remained unchanged since 1980, which is better than CS Ph.D. production for African Americans and all underrepresented groups combined.

Considering these numbers, it would appear there are many African Americans and other underrepresented minorities pursuing science and engineering Ph.D.'s; however, so few chose CS. Data from many decades ago suggests African Americans do not pursue science, technology, engineering, and mathematics (STEM) degrees because of cultural stigmas (including sentiments such as it's not cool to be a scientist or scientists are nerds). Hager and Elton[3] surveyed college freshmen and Sewell and Martin[6] surveyed high school juniors. In these two studies, they found African American men expressed a greater interest in social service fields compared to White men, who prefer STEM disciplines. In general, African American college students are highly represented in disciplines such as education, humanities, and social sciences.[5] Hall and Post-Kammer[4] reported African Americans choose these disciplines because they have a cultural orientation and expectation to help others. Historically, STEM disciplines are generally not seen as disciplines that can be used to help others. Do these cultural stigmas still apply and are they the reason for the low representation of African Americans in CS Ph.D. programs? If so, one possible solution could be to show examples of how computing research can have an impact on society. That is, demonstrate how computing can change lives and improve the human condition, especially for African Americans. Connecting computing to solving human conditions that will likely impact communities of color is one of the primary goals of the iAAMCS.

Table 1. African American Ph.D. production 2003–2013 CRA Taulbee Survey and 2006–2012 NSF SED.

Taulbee Survey			NSF SED NCSES		
Year	Reported	Percent	Year	Reported	Percent
2003	10	1.30%			
2004	12	1.50%			
2005	9	0.50%			
2006	18	1.40%	2006	20	1.38%
2007	19	1.20%	2007	37	2.24%
2008	22	1.50%	2008	36	2.01%
2009	17	1.30%	2009	33	2.05%
2010	17	1.30%	2010	37	2.22%
2011	16	1.20%	2011	39	2.28%
2012	27	1.80%	2012	40	2.17%
2013	22	1.50%			

Table 2. African American CS faculty 2003–2013 CRA Taulbee Survey.

Year	Full	Percent	Associate	Percent	Assistant	Percent
2003	6	0.40%	10	0.90%	16	1.40%
2004	9	0.60%	8	0.70%	24	2.00%
2005	7	0.40%	12	1.10%	23	1.90%
2006	8	0.50%	11	0.90%	26	2.30%
2007	8	0.50%	11	0.90%	21	2.10%
2008	14	0.70%	20	1.40%	21	2.00%
2009	10	0.50%	16	1.20%	22	2.50%
2010	11	0.60%	17	1.20%	24	2.90%
2011	12	0.60%	21	1.40%	23	3.00%
2012	16	0.80%	25	1.60%	26	3.30%
2013	16	0.80%	25	1.80%	25	3.50%

Emerging National Resource for Diversifying Computing Sciences

The iAAMCS is a NSF Broadening Par-

ticipation in Computing (BPC) Alliance (see http://www.iaamcs.org). The mission of iAAMCS is to: increase the number of African Americans receiving Ph.D. degrees in computing sciences; promote and engage students in teaching and training opportunities; and add more diverse researchers into the advanced technology workforce. iAAMCS provides various activities that serve as interventions for increasing interest in computing sciences among underrepresented minorities with a particular focus on African Americans, as described here.

Faculty and student training. This activity consists of face-to-face workshops and webinars for African American (and other underrepresented) students and their advisors to provide a shared context for the completion of the research experience. The students received training in time management, managing expectations, and research processes (for example, literature review, source control, and so forth). The advisor's training is based on documented differences between mentoring by effective and non-effective teachers of African American (and other underrepresented) students.

Academic year undergraduate research (AYUR). The purpose of AYUR is to increase the African American Ph.D. pipeline by introducing second-semester freshmen and sophomore African American students in computing to research. Specifically, the objectives of AYUR are to: develop undergraduate research competence in a computing area such as robotics, game programming, mobile applications, or other computing research area; increase critical inquiry and critical thinking associated with conducting research; increase active engagement through a research experience (presentations, competitions, and so forth); develop a repository of models of effective undergraduate research mentoring; and ultimately, increase the number of African American computing students with an interest in pursuing graduate degrees in CS. AYUR also encourages students to extend their research beyond this activity and pursue other research opportunities (for example, REUs).

Distributed research experiences for undergraduates (DREU). Similar to AYUR, DREU is an activity that provides research opportunities to African American (and other underrepresented) students in preparation for graduate school. This activity accepts applications from students and mentors who are then matched based on interests and backgrounds. During DREU, students complete a 10-week research experience that consists of several checkpoints in the process to ensure uniform expectations and outcomes. The DREU program is a collaboration between the CRA-W (Computing Research Association's Committee on the Status of Women in Computing Research) and the CDC (Coalition to Diversity Computing).

These numbers are promising, but there is still much work to be done.

Technical webinars and distinguished lecture series (DLS). DLS is an activity that provides targeted presentation interventions for African American students (and other underrepresented groups) in computing sciences. Topics include: computing research, academic faculty employment, research scientist positions, and other topics related to the benefits of getting a Ph.D. in computing sciences. Each presentation is an hour long with a period dedicated for the mentor leading the session to answer questions from attendees. These lectures are held onsite at the host institution that places the request for the lecture.

Distinguished fellows writing workshop (DFWW). DFWW serves as a platform for African American undergraduate and graduate students (and other underrepresented groups) to learn the process of writing a competitive application for summer internships, graduate school, and/or external funding. The targeted audience for this activity is junior and senior-level undergraduates and first- and second-year students as well as faculty that advise or mentor these students.

Calendar of Events

July 4–8
Sponsored: Innovation and Technology in Computer Science Education Conference 2015,
Vilnius, Lithuania,
Sponsored: ACM/SIG,
Contact: Valentina Dagiene
Email: valentina.dagiene@mii.vu.lt

July 6–9
ISSAC'15: International Symposium on Symbolic and Algebraic Computation,
Bath, UK,
Sponsored: ACM/SIG,
Contact: Stephen Alexander Linton
Email: steve.linton@st-andrews.ac.uk

July 6–July 10
LICS '15: 2015 ACM/IEEE Symposium on Logic in Computer Science,
Kyoto, Japan
Co-Sponsored: Other Societies,
Contact: Masahito Hasegawa
Email: hassei@kurims.kyoto-u.ac.jp

July 14–17
e-Energy'15: The Sixth International Conference on Future Energy Systems,
Bangalore, India,
Sponsored: ACM/SIG,
Contact: Deva P. Seetharam,
Email: deva.seetharam@gmail.com

July 16–17
SIGDOC '15: The 33rd ACM International Conference on the Design of Communication,
Limerick Ireland,
Sponsored: ACM/SIG,
Contact: Kathie Gossett,
Email: kgossett@iastate.edu

July 20–24
ISSTA '15: International Symposium on Software Testing and Analysis,
Baltimore, MD,
Sponsored: ACM/SIG,
Contact: Michal T. Young,
Email: michal@cs.uoregon.edu

July 21–23
PODC '15: ACM Symposium on Principles of Distributed Computing,
Donostia-San Sebastián Spain
Co-Sponsored: ACM/SIG,
Contact: Chryssis Georgiou,
Email: chryssis@cs.ucy.ac.cy

Troy Hill, from Winston Salem State University, won first place in the Robotics Simulation Competition at ARTSI Robotics Competition at the 2014 Tapia Conference.

K–12 outreach. This activity provides opportunities for African American undergraduate and graduate students (and other underrepresented groups) to work with middle and high school students through computing awareness and exposure events, afterschool programs, and summer camps. The goal of these programs is to provide opportunities for low-income and middle-class African American, Latino, and female students to explore computer science and develop programming and computational thinking skills. Through graduate and undergraduate participation in these programs, we aim to provide middle school and high school students with role models with similar racial and gender backgrounds to help them develop identities as computer scientists. For the graduate and undergraduate facilitators we aim to provide opportunities for them to give back to communities with similar demographics as the ones they came from and to help bolster their skills and identities as computer scientists.

ARTSI robotics competition. iAAMCS is the sponsor of (and provides some funding for) the ARTSI robotics competition. The objective of this competition is to develop an active robotics community for African American (and other underrepresented) students and recruit them to pursue graduate training and careers in research. This competition is held annually at the ACM Richard Tapia Celebration of Diversity in Computing Conference.

Tapia Celebration of Computing in Diversity Conference. iAAMCS sponsors students to attend this event. The premise of this activity is that there are professionals across the U.S. who are interested in supporting African American (and other diverse) students. Members of iAAMCS are both highly visible and deeply aware of who many of those individuals are. Underrepresented minority and female students are much less aware of those individuals. So the vision is that identifying and engaging this group of individuals could serve as a strong resource, both as role models and mentors for these students. When a broad group of diverse students interact with each other, there are benefits given their common bond of being underrepresented.

Conclusion

Currently, iAAMCS is in its second year as an organization. It is developing a model to engage, support, and sustain students via a national network. Building upon the prior relationships and networks developed by related organizations such as ELA and XSEDE, iAAMCS participants are connected to a larger network of peers, faculty, and computing professionals to meet the student's individual interests and needs. iAAMCS participants also join a network of individuals across the country, building a virtual (and sometimes in-person) network. This allows for any individual student, faculty member, or computing professional across the U.S. to become connected to others as a part of iAAMCS, including (in fact targeting) those that are not a member of any partnering institution. If iAAMCS meets its objectives, there should be shifts in the numbers of African Americans pursuing and receiving Ph.D.'s in CS. There should also be a continued increase in their representation in CS tenure-track faculty positions at all ranks. The iAAMCS projects are designed to provide mentoring, support, and a different perspective of computing for all underrepresented minorities with a focus on African Americans. C

References

1. Gibbs, K.D., Jr., McGready, J., Bennett, J.C., Griffin, K. Biomedical science Ph.D. career interest patterns by race/ethnicity and gender. *PLoS ONE 9*, 12 (Dec. 2014), e114736: DOI: 10.1371/journal.pone.0114736.
2. Guynn, J. Intel pledges diversity by 2020, invests $300 million. *USA Today* (2015); http://www.usatoday.com.
3. Hager, P.C. and Elton, C.F. The vocational interests of Black males. *Journal of Vocational Behavior 1* (1971), 153–158.
4. Hall, E.R. and Post-Kammer, P. Black mathematics and science majors: Why so few? *Career Development Quarterly 35* (1987), 206–219.
5. Powell, L. Factors associated with the underrepresentation of African-Americans in mathematics and science. *The Journal of Negro Education 59*, 3 (1990), 292–298.
6. Sewell, T.E. and Martin, R.P. Racial differences in patterns of occupational choice in adolescents. *Psychology in the Schools 13*, (1976), 326–333.
7. Taylor, V. and Ladner, R. Data trends on minorities and people with disabilities in computing. *Commun. ACM 54*, 12 (Dec. 2011), 34–37.
8. Vara, V. Can Intel make silicon valley more diverse? *The New Yorker* (2015); http://www.newyorker.com.

Juan E. Gilbert (juan@juangilbert.com) holds the Andrew Banks Family Preeminence Endowed Chair and is the associate chair of research in the Computer and Information Science and Engineering Department at the University of Florida where he leads the Human-Experience Research Lab.

Jerlando F.L. Jackson (jjackson@education.wisc.edu) is the Vilas Distinguished Professor of Higher Education and the director and chief research scientist of Wisconsin's Equity and Inclusion Laboratory (Wei LAB) at the University of Wisconsin-Madison.

Edward C. Dillon, Jr. (ecdillon@cise.ufl.edu) the iAAMCS project manager and a postdoctoral associate in the Computer and Information Science and Engineering Department at the University of Florida.

LaVar J. Charleston (charleston@wisc.edu) is the assistant director and a senior research associate at Wisconsin's Equity and Inclusion Laboratory (Wei LAB) within the Wisconsin Center for Education Research at the University of Wisconsin-Madison.

This material is based in part upon work supported by the National Science Foundation under Grant Number CNS-1457855. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

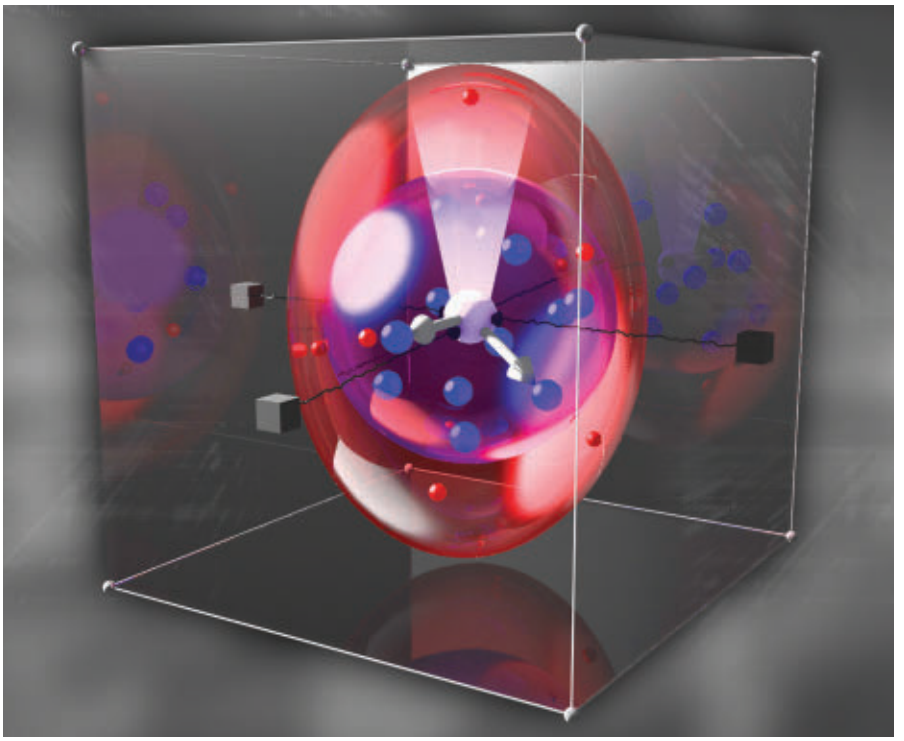
IMAGE COURTESY OF IAAMCS

DOI:10.1145/2668022

Vijay Kumar and Thomas A. Kalil

Viewpoint
The Future of Computer Science and Engineering Is in Your Hands

How government service can profoundly influence computer science research and education.



A conceptualized software system inspired by a DARPA project investigating software systems that will remain functional in excess of 100 years.

IMAGE COURTESY OF DARPA

MANY RESEARCHERS ARE concerned about flat or declining research budgets in computer science and engineering. Although the research community is right to be concerned about the shortage of funds, they should also be concerned about the shortage of scientists and engineers that are willing to serve in governments to address this challenge. Computer scientists and engineers can have a huge impact on the future of the field and the future of the U.S. By serving in the government, they can design and launch new research initiatives, inform IT-related policy decisions, and serve as a catalyst for public-private partnerships involving government, industry, and academia.

In the U.S., one way of advancing computer science and shaping public policy in computer science research and education is by serving in federal agencies that oversee investments in education and research in science and technology. Program and division directors at the National Science Foundation (NSF), the National Institutes of Health (NIH), National Aeronautics and Space Administration (NASA), the Defense Advanced Research Projects Agency (DARPA), the National Institute for Standards and Technology (NIST), the United States Department of Agriculture (USDA), the Department of Transportation (DOT), the Department of Homeland Security (DHS), and many other federal agencies help create national research and development initiatives that can create and/or transform disciplines.

The White House Office of Science and Technology Policy (OSTP) is an

agency that works with all other federal agencies. OSTP's overarching mission[a] is to "ensure that federal investments in science and technology are making the greatest possible contribution to economic prosperity, public health, environmental quality, and national security." OSTP also works to formulate policies that can advance science and technology and to make sure in-depth scientific knowledge and expertise in technology are used to inform policymaking. Over the last three years, OSTP has championed a number of new research initiatives that support computer science and engineering, or that harness information technology to advance national priorities and accelerate the pace of discovery in science and engineering.

In recent years, the U.S. government has created two national education and research initiatives that are particularly relevant to computer science and engineering. The National Robotics Initiative (NRI) is a multi-agency initiative designed by OSTP to support research and development in robotics science and technology and promote applications of robotics in manufacturing, healthcare, agriculture, space exploration, national defense, homeland security, civil infrastructure, and education. In June 2011, President Obama unveiled the NRI, with an initial focus on robots that can work with humans to extend and augment human skills. NSF, NIH, NASA, and USDA issued a joint NRI solicitation[b] and committed over $50 million to develop the science and technology for robots that can safely coexist and operate in close proximity to humans. The NRI now, in its third year, includes a team of over 50 program managers from nine different funding agencies and many of the best U.S. laboratories including the Army Research Laboratory, the Naval Research Laboratory, and the NASA Johnson Space Center. The NRI is broadening in scope to include a wider range of applications and will help strengthen U.S. leadership in robotics science and technology.

a Office of Science and Technology Policy; http://www.whitehouse.gov/ostp

b National Robotics Initiative; http://www.nsf.gov/pubs/2011/nsf11553/nsf11553.htm

OSTP has also designed a closely related interagency initiative in cyber physical systems (CPS) with active participation from NSF, NIST, DOT, and DHS. CPS are smart networked systems with embedded sensors, processors, and actuators that are designed to sense and interact with the physical world (including human users). CPS technology is pervasive and will transform entire industrial sectors, including transportation, healthcare, energy, manufacturing, and agriculture. This initiative addresses the urgent need for research and development for the design and verification of software to provide guaranteed performance, the around-the-clock reliability needed for safety-critical applications; education and training required for CPS applications; and development of standards and reference architectures for CPS in different industry sectors.

NSF has already invested in CPS by committing over $140M for CPS research and education over the last four years.[c] In addition, NSF has created a CPS Virtual Organization (VO),[d] a network of federal agencies, private industry, and academic research labs actively engaged in CPS research, development, and education. Two White House Presidential Innovation Fellows supported by the National Institute for Standards and Technology (NIST) are

c Cyber-Physical Systems; http://www.nsf.gov/pubs/2013/nsf13502/nsf13502.htm.

d Cyber-Physical Systems Virtual Organization; http://cps-vo.org/

The last two years have been particularly exciting for robotics, cyber physical systems, and the Internet of Things.

addressing sector-specific open testbeds for research and development, and for creating standards and reference architectures for CPS. Industry leaders such as GE, AT&T, Intel, IBM, and Cisco are exploring mechanisms for collaborating with each other and with government agencies and university researchers.

It is important to note that leaders from academia and industry have played a very significant role in shaping both the NRI and CPS initiatives. In 2008, the community pressed to create a National Robotics Week. In May 2009 and then again in March 2013, roadmaps[e] for robotics research, development and education, were presented to the Congressional Robotics Caucus.[f] OSTP, NSF, NIST, and other agencies held many workshops to help refine the vision for the future of robotics and CPS. In both these initiatives, individuals who served as rotators in the positions of program or division director at NSF, NASA, USDA, and NIH invested resources under their control to create new programs.

The last two years have been particularly exciting for robotics, CPS, and the Internet of Things. For example, on December 2, 2013, Amazon CEO Jeff Bezos announced plans for Amazon Prime Air, which will deliver small packages weighing less than five pounds (which constitute 86% of products shipped by Amazon) using aerial robots in the next five years. Google has acquired robotics companies with expertise in legged robots, low-cost arms, perception, virtual reality, and omnidirectional wheels. Recent reports on the Internet of Things predict economic impact in the tens of trillions of dollars[1] and over 75% of business leaders surveyed predicted a direct impact of this technology on their business.[2]

While there are a number of technology demonstrations in robotics and CPS that suggest that these fields are becoming mature, many of these solutions only work under tightly constrained conditions. The recent DARPA Robotics Challenge serves to highlight many of the open problems in robotics in addition to underscoring the tremendous potential of this field. This is why it is important for all of us to be collectively and continuously engaged in science and technology policy. We need experts that can help identify the most important challenges in robotics and CPS, and that can design high-impact research initiatives to address these challenges.

Federal agencies benefit from leadership in computer science and information technologies.

It is also important to realize that behind the recent avalanche of success stories involving start-ups and investments by major companies is the "snowball" of decades of federal funding in these fields. With targeted investments we can create the scientific and technology breakthroughs that will serve as a catalyst for the industries and jobs of the future. For example, the U.S. federal government should make investments that promote the infrastructure needed for R&D; reduce the "design, build, test" cycle; and attract the next generation of scientists, engineers, and entrepreneurs to advance robotics and CPS, and their applications in domains such as manufacturing, transportation, healthcare, and energy.

Given the importance of these and many other opportunities, it is time for more members of the computer science community to spend time in Washington, D.C. Whether you are in industry or in academia, a student or a professional, we urge you to reach out and connect with federal government agencies whose missions are aligned with your interests. OSTP offers semester-long internships for students of all majors. The Presidential Innovation Fellows (PIF) program[g] offers six- or 12-month appointments for innovators in the private sector who want to work with the government on developing high-impact technological solutions that can save lives, create new jobs, or make government more efficient. At OSTP, appointments of a year or more are possible through the Intergovernmental Personnel Act (IPA) and this can offer a great opportunity for those who have a sabbatical leave. There are also critical opportunities in other agencies such as NSF and DARPA. Indeed, the creation of the NRI and CPS initiatives was in large part because of individuals that took advantage of these opportunities at NSF.

Agencies like these clearly play a critical role in funding computer science and engineering research and education. However, federal agencies benefit from leadership in computer science and information technology. For example, NIH recently recruited a chief data scientist and many agencies have recruited chief technology officers.

In the U.S., the Office of Science and Technology Policy and many other federal agencies are committed to investing in computer science research and to harnessing information technology to address some of the nation's most important challenges. In other countries, the agencies responsible for R&D investment and their charters are obviously different. However, as in any country, we cannot do this without the active participation and engagement of creative members of the research community. Democracy is not a spectator sport, and government is only as effective as the people that chose to embrace public service. C

e Robotics Virtual Organization; http://www.robotics-vo.us/

f Robotics Caucus; http://www.roboticscaucus.org/

g Presidential Innovation Fellows; http://www.whitehouse.gov/innovationfellows/

References

1. Disruptive technologies: Advances that will transform life, business, and the global economy. McKinsey Global Institute Report, May 2013; http://www.mckinsey.com/insights/business_technology/disruptive_technologies.
2. The Internet of Things Business Index, A report from The Economist Intelligence Unit, 2013; http://www.economistinsights.com/analysis/internet-things-business-index.

Vijay Kumar (kumar@seas.upenn.edu) is UPS Foundation Professor at the University of Pennsylvania and served as the assistant director of Robotics and Cyber Physical Systems at the White House Office of Science and Technology Policy from 2012–2014.

Thomas A. Kalil (Thomas_A._Kalil@ostp.eop.gov) is Deputy Director for Technology and Innovation at the White House Office of Science and Technology Policy.

DOI:10.1145/2747303

Article development led by acmqueue
queue.acm.org

The finance industry has unique demands for low-latency distributed systems.

BY ANDREW BROOK

Low-Latency Distributed Applications in Finance

VIRTUALLY ALL SYSTEMS have some requirements for latency, defined here as the time required for a system to respond to input. (Non-halting computations exist, but they have few practical applications). Latency requirements appear in problem domains as diverse as aircraft flight controls,[a] voice communications,[b] multiplayer gaming,[c] online advertising,[d] and scientific experiments.[e]

Distributed systems—in which computation occurs on multiple networked computers that communicate and coordinate their actions by passing messages—present special latency considerations. In recent years the automation of financial trading has driven requirements for distributed systems with challenging latency requirements (often measured in microseconds or even nanoseconds; see Table 1) and global geographic distribution. Automated trading provides a window into the engineering challenges of ever-shrinking latency requirements and therefore may be useful to software engineers in other fields.

This article focuses on applications where latency (as opposed to throughput, efficiency, or some other metric) is one of the primary design considerations. Phrased differently, "low-latency systems" are those for which latency is the main measure of success and is usually the toughest constraint to design around. The article presents examples of low-latency systems that illustrate the external factors that drive latency and then discusses some practical engineering approaches to building systems that operate at low latency.

Why Is Everyone in Such a Hurry?

To understand the impact of latency on an application, it is important first to understand the external, real-world factors that drive the requirement. The following examples from the finance industry illustrate the impact of some real-world factors.

Request for quote trading. In 2003 I worked at a large bank that had just deployed a new Web-based institutional foreign-currency trading system. The quote and trade engine, a J2EE (Java 2 Platform, Enterprise Edition) application running in a WebLogic server on top of an Oracle database, had response times that were reliably under two seconds—fast enough to ensure good user experience.

Around the same time the bank's website went live, a multibank online trading platform was launched. On this new platform, a client would submit an RFQ (request for quote) that would be forwarded to multiple participating banks. Each bank would respond with a quote, and the client would choose which one to accept.

My bank initiated a project to connect to the new multibank platform. The reasoning was that since a two-second response time was good enough for a user on the website, it should be good enough for the new platform, and

ILLUSTRATION BY MARIO WAGNER

Table 1. Units of time.

Millisecond = 10^{-3} seconds	1,000 milliseconds = 1 second
Microsecond = 10^{-6} seconds	1,000 microseconds = 1 millisecond
Nanosecond = 10^{-9} seconds	1,000 nanoseconds = 1 microsecond

Table 2. Sequence of events.

100	Banks A and B receive messages indicating that the interbank rate increased to 1.3563 / 1.3566. Both start the process of computing updated quotes.
150	Bank A publishes an updated quote of 1.3562 / 1.3567.
200	Client C receives the updated quote from bank A but still has the old quote from bank B of 1.3557 / 1.3561.
210	Client C recognizes an arbitrage opportunity and sends a request to sell 1 million euros to bank A (receiving $1.3562 million) and at the same time sends a request to buy 1 million euros from bank B (paying $1.3561 million).
260	Bank A receives the request from C in which A receives 1 million euros from C and pays $1.3562 million in return.
260	Bank B receives the request from C in which B pays 1 million euros to C and receives $1.3561 million.
270	Since the quote is valid, A sends back an acknowledgment to C and at the same time sends a request to the interbank market to sell 1 million euros.
270	B likewise finds the request to be for a valid quote (it has not yet sent out the update that was started at 100ms) so acknowledges to C and sends a request to buy 1 million euros from the interbank market.
320	C receives acknowledgments from A and B. Its net profit is $100 (it received $1.3562 million from A and paid $1.3561 to B).
350	Bank B publishes its updated quote of 1.3562 / 1.3567. This does not change the processing of the order that was already received from client C at the old rate.
380	Bank A receives acknowledgment from the interbank market. Bank A makes a net profit of $100 (it paid $1.3562 million to C and received $1.3563 million from the interbank market).
380	Bank B receives acknowledgment from the interbank market. Unfortunately, the current interbank rate is 1.3563 / 1.3566, so it its order is filled at 1.3566, which means it pays $1.3566 million to buy the 1 million euros needed to offset the amount sold to C. Bank B thus suffers a net loss of $500.

Table 3. Examples of approximate latencies.

0 (1 second)	Acceptable response time to Web page loads or other query-/response-style interactions.
-1 (100 milliseconds)	Human perception of delay in interactive media (for example, voice communications). Real-time advertising auctions. Network latency across continents or oceans.
-2 (10 milliseconds)	Hard-disk seek times and related I/O operations on spinning disks when the data is not cached.
-3 (1 millisecond)	Real-time analytics in competitive environments (for example, natural language processing, topic classification).
-4 (100 microseconds)	Limits of metropolitan-area communications: microwaves travel about 30km in air; light travels about 20km in fiber.
-5 (10 microseconds)	Trade execution logic for relatively simple strategies running on CPU.
-6 (1 microsecond)	Sending a packet from one server to another over a single-hop local-area network.
-7 (100 nanoseconds)	Main memory access on modern Intel microprocessors. Thread context switch (best case; it can be much worse). Switching latency in 10Gbps cut-through switch.
-8 (10 nanoseconds)	Accuracy of good commercial GPS-based time sources.
-9 (1 nanosecond)	Time to execute a single instruction on a modern CPU, assuming that operands are in registers or L1 cache.

so the same quote and trade engine could be reused. Within weeks of going live, however, the bank was winning a surprisingly small percentage of RFQs. The root cause was latency. When two banks responded with the same price (which happened quite often), the first response was displayed at the top of the list. Most clients waited to see a few different quotes and then clicked on the one at the top of the list. The result was the fastest bank often won the client's business—and my bank was not the fastest one.

The slowest part of the quote-generation process occurred in the database queries loading customer-pricing parameters. Adding a cache to the quote engine and optimizing a few other "hot spots" in the code brought quote latency down to the range of approximately 100 milliseconds. With a faster engine, the bank was able to capture significant market share on the competitive quotation platform—but the market continued to evolve.

Streaming quotes. By 2006 a new style of currency trading was becoming popular. Instead of a customer sending a specific request and the bank responding with a quote, customers wanted the banks to send a continuous *stream* of quotes. This streaming-quotes style of trading was especially popular with certain hedge funds that were developing automated trading strategies—applications that would receive streams of quotes from multiple banks and automatically decide when to trade. In many cases, humans were now out of the loop on both sides of the trade.

To understand this new competitive dynamic, it is important to know how banks compute the rates they charge their clients for foreign-exchange transactions. The largest banks trade currencies with each other in the so-called interbank market. The exchange rates set in that market are the most competitive and form the basis for the rates (plus some markup) that are offered to clients. Every time the interbank rate changes, each bank recomputes and republishes the corresponding client rate quotes. If a client accepts a quote (that is, requests to trade against a quoted exchange rate), then the bank can immediately execute an offsetting trade with the interbank market, minimizing risk and locking in a small profit. There are, however, risks

to banks that are slow to update their quotes. A simple example can illustrate:

Imagine the interbank spot market for EUR/USD has rates of 1.3558/1.3560. (The term *spot* means the agreed-upon currencies are to be exchanged within two business days. Currencies can be traded for delivery at any mutually agreed-upon date in the future, but the spot market is the most active in terms of number of trades.) Two rates are quoted: one for buying (the *bid* rate), and one for selling (the *offered* or *ask* rate). In this case, a participant in the interbank market could sell one euro and receive 1.3558 U.S. dollars in return. Conversely, one could buy one euro for a price of 1.3560 U.S. dollars.

Say that two banks, A and B, are participants in the interbank market and are publishing quotes to the same hedge-fund client, C. Both banks add a margin of 0.0001 to the exchange rates they quote to their clients—so both publish quotes of 1.3557/1.3561 to client C. Bank A, however, is faster at updating its quotes than bank B, taking about 50 milliseconds while bank B takes about 250 milliseconds. There are approximately 50 milliseconds of network latency between banks A and B and their mutual client C. Both banks A and B take about 10 milliseconds to acknowledge an order, while the hedge fund C takes about 10 milliseconds to evaluate new quotes and submit orders. Table 2 breaks down the sequence of events.

The net effect of this new streaming-quote style of trading was that any bank that was significantly slower than its rivals was likely to suffer losses when market prices changed and its quotes were not updated quickly enough. At the same time, those banks that could update their quotes fastest made significant profits. Latency was no longer just a factor in operational efficiency or market share—it directly impacted the profit and loss of the trading desk. As the volume and speed of trading increased throughout the mid-2000s, these profits and losses grew to be quite large. (How low can you go? Table 3 shows some examples of approximate latencies of systems and applications across nine orders of magnitude.)

To improve its latency, my bank split its quote and trading engine into distinct applications and rewrote the quote engine in C++. The small delays added by each hop in the network from the interbank market to the bank and onward to its clients were now significant, so the bank upgraded firewalls and procured dedicated telecom circuits. Network upgrades combined with the faster quote engine brought end-to-end quote latency down below 10 milliseconds for clients who were physically located close to our facilities in New York, London, or Hong Kong. Trading performance and profits rose accordingly—but, of course, the market kept evolving.

Engineering Systems for Low Latency

The latency requirements of a given application can be addressed in many ways, and each problem requires a different solution. There are some common themes, though. First, it is usually necessary to measure latency before it can be improved. Second, optimization often requires looking below abstraction layers and adapting to the reality of the physical infrastructure. Finally, it is sometimes possible to restructure the algorithms (or even the problem definition itself) to achieve low latency.

Lies, damn lies, and statistics. The first step to solving most optimization problems (not just those that involve software) is to measure the current system's performance. Start from the highest level and measure the end-to-end latency. Then measure the latency of each component or processing stage. If any stage is taking an unusually large portion of the latency, then break it down further and measure the latency of its substages. The goal is to find the parts of the system that contribute the most to the total latency and focus optimization efforts there. This is not always straightforward in practice, however.

For example, imagine an application that responds to customer quote requests received over a network. The client sends 100 quote requests in quick succession (the next request is sent as soon as the prior response is received) and reports total elapsed time

Example quote engine and test harness.

```
(In the client application)

for (int i = 0; i < 100; i++){
  RequestMessage requestMessage = new RequestMessage(quoteRequest);
  long sentTime = getSystemTime();
  sendMessage(requestMessage);
  ResponseMessage responseMessage = receiveMessage();
  long quoteLatency = getSystemTime() - sentTime;
  logStats(quoteLatency);
}

(In the quote server application)

while (true){
  RequestMessage requestMessage = receive();
  long receivedTime = getSystemTime();
  QuoteRequest quoteRequest = parseRequest(requestMessage);
  long parseTime = getSystemTime();
  long parseLatency = parseTime - receivedTime;
  ClientProfile profile = lookupClientProfile(quoteRequest.client);
  long profileTime = getSystemTime();
  long profileLatency = profileTime - parseTime;
  Quote quote = computeQuote(profile);
  long computeTime = getSystemTime();
  long computeLatency = computeTime - profileTime;
  logQuote(quote);
  long logTime = getSystemTime();
  long logLatency = logTime - computeTime;
  QuoteMessage quoteMessage = new QuoteMessage(quote);
  long serializeTime = getSystemTime();
  long serializationLatency = serializeTime - logTime;
  send(quoteMessage);
  long sentTime = getSystemTime();
  long sendLatency = sentTime - serializeTime;
  logStats(parseLatency, profileLatency, computeLatency,
           logLatency, serializationLatency, sendLatency);
}
```

of 360 milliseconds—or 3.6 milliseconds on average to service a request. The internals of the application are broken down and measured using the same 100-quote test set:

- Read input message from network and parse: 5 microseconds
- Look up client profile: 3.2 milliseconds (3,200 microseconds)
- Compute client quote: 15 microseconds
- Log quote: 20 microseconds
- Serialize quote to a response message: 5 microseconds
- Write to network: 5 microseconds

As clearly shown in this example, significantly reducing latency means addressing the time it takes to look up the client's profile. A quick inspection shows the client profile is loaded from a database and cached locally. Further testing shows when the profile is in the local cache (a simple hash table), response time is usually under a microsecond, but when the cache is missed it takes several *hundred* milliseconds to load the profile. The average of 3.2 milliseconds was almost entirely the result of one very slow response (of about 320 milliseconds) caused by a cache miss. Likewise, the client's reported 3.6-millisecond average response time turns out to be a single very slow response (350 milliseconds) and 99 fast responses that took around 100 microseconds each.

Means and outliers. Most systems exhibit some variance in latency from one event to the next. In some cases the variance (and especially the highest-latency outliers) drive the design much more so than the average case. It is important to understand which statistical measure of latency is appropriate to the specific problem. For example, if you are building a trading system that earns small profits when the latency is below some threshold but incurs massive losses when latency exceeds that threshold, then you should be measuring the peak latency (or, alternatively, the percentage of requests that exceed the threshold) rather than the mean. On the other hand, if the value of the system is more or less inversely proportional to the latency, then measuring (and optimizing) the average latency makes more sense even if it means there are some large outliers.

What are you measuring? Astute readers may have noticed the latency measured inside the quote server application does not quite add up to the latency reported by the client application. That is most likely because they are not actually measuring the same thing. Consider the simplified pseudo-code illustrated in the accompanying figure.

Note the elapsed time measured by the client application includes the time to transmit the request over the network, as well as the time for the response to be transmitted back. The quote server, on the other hand, measures the time elapsed only from the arrival of the quote to when it is sent (or more precisely, when the `send` method returns). The 350-microsecond discrepancy between the average response time measured by the client and the network could cause the equivalent measurement by the quote server, but it might also be the result of delays within the client or server. Moreover, depending on the programming language and operating system, checking the system clock and logging the latency statistics may introduce material delays.

This approach is simplistic, but when combined with code-profiling tools to find the most commonly executed code and resource contention, it is usually good enough to identify the first (and often easiest) targets for latency optimization. It is important to keep this limitation in mind, though.

Measuring distributed systems latency via network traffic capture. Distributed systems pose some additional challenges to latency measurement—as well as some opportunities. In cases where the system is distributed across multiple servers it can be difficult to correlate timestamps of related events. The network itself can be a significant contributor to the latency of the system. Messaging middleware and the networking stacks of operating systems can be complex sources of latency.

At the same time, the decomposition of the overall system into separate processes running on independent servers can make it easier to measure certain interactions accurately between components of the system over the network. Many network devices (such as switches and routers) provide mechanisms for making time-stamped copies of the data that traverse the device with minimal impact on the performance of the device. Most operating systems provide similar capabilities in software, albeit with a somewhat higher risk of delaying the actual traffic. Time-stamped network-traffic captures (often called *packet captures*) can be a useful tool to measure more precisely when a message was exchanged between two parts of the system. These measurements can be obtained without modifying the application itself and generally with very little impact on the performance of the system as a whole.[f]

Clock synchronization. One of the challenges of measuring performance at short time scales across distributed systems is clock synchronization. In general, to measure the time elapsed from when an application on server A transmits a message to a second application on server B, it is necessary to check the time on A's clock when the message is sent and on B's clock when the message arrives, and then subtract those two timestamps to determine the latency. If the clocks on A and B are not in sync, then the computed latency will actually be the real latency plus the clock skew between A and B.

When is this a problem in the real world? Real-world drift rates for the quartz oscillators that are used in most commodity server motherboards are on the order of 10^{-5}, which means the oscillator may be expected to drift by 10 microseconds each second. If uncorrected, it may gain or lose as much as a second over the course of a day. For systems operating at time scales of milliseconds or less, clock skew may render the measured latency meaningless. Oscillators with significantly lower drift rates are available, but without some form of synchronization, they will eventually drift apart. Some mechanism is needed to bring each server's local clock into alignment with some common reference time.

Developers of distributed systems should understand Network Time Protocol (NTP) at a minimum and are encouraged to learn about Precision Time Protocol (PTP) and usage of external signals such as GPS to obtain high-accuracy time synchronization in practice. Those who need time accuracy at the sub-microsecond scale will want to become familiar with hardware implementations of PTP (especially at

the network interface) as well as tools for extracting time information from each core's local clock.[g]

Abstraction vs. reality. Modern software engineering is built upon abstractions that allow programmers to manage the complexity of ever-larger systems. Abstractions do this by simplifying or generalizing some aspect of the underlying system. This does not come for free, though—simplification is an inherently lossy process and some of the lost details may be important. Moreover, abstractions are often defined in terms of function rather than performance.

Somewhere deep below an application are electrical currents flowing through semiconductors and pulses of light traveling down fibers. Programmers rarely need to think of their systems in these terms, but if their conceptualized view drifts too far from reality they are likely to experience unpleasant surprises.

Four examples illustrate this point:

▸ TCP provides a useful abstraction over UDP (User Datagram Protocol) in terms of delivery of a sequence of bytes. TCP ensures bytes will be delivered in the order they were sent even if some of the underlying UDP datagrams are lost. The transmission latency of each byte (the time from when it is written to a TCP socket in the sending application until it is read from the corresponding receiving application's socket) is not guaranteed, however. In certain cases (specifically when an intervening datagram is lost) the data contained in a given UDP datagram may be delayed significantly from delivery to the application, while the missed data ahead of it is recovered.

▸ Cloud hosting provides virtual servers that can be created on demand without precise control over the location of the hardware. An application or administrator can create a new virtual server "on the cloud" in less than a minute—an impossible feat when assembling and installing physical hardware in a data center. Unlike the physical server, however, the location of the cloud server or its location in the network topology may not be precisely known. If a distributed application depends on the rapid exchange of messages between servers, then the physical proximity of those servers may have a significant impact on the overall application performance.

The latency requirements of a given application can be addressed in many ways, and each problem requires a different solution.

▸ Threads allow developers to decompose a problem into separate sequences of instructions that can be allowed to run concurrently, subject to certain ordering constraints, and that can operate on shared resources (such as memory). This allows developers to take advantage of multicore processors without needing to deal directly with issues of scheduling and core assignment. In some cases, however, the overhead of context switches and passing data between cores can outweigh the advantages gained by concurrency.

▸ Hierarchical storage and cache-coherency protocols allow programmers to write applications that use large amounts of virtual memory (on the order of terabytes in modern commodity servers), while experiencing latencies measured in nanoseconds when requests can be serviced by the closest caches. The abstraction hides the fact the fastest memory is very limited in capacity (for example, register files on the order of a few kilobytes), while memory that has been swapped out to disk may incur latencies in the tens of milliseconds.

Each of these abstractions is extremely useful but can have unanticipated consequences for low-latency applications. There are some practical steps to take to identify and mitigate latency issues resulting from these abstractions.

Message and network protocols. The near ubiquity of IP-based networks means that regardless of which messaging product is in use, under the covers the data is being transmitted over the network as a series of discrete packets. The performance characteristics of the network and the needs of an application can vary dramatically—so one size almost certainly does not fit all when it comes to messaging middleware for latency-sensitive distributed systems.

There is no substitute for getting under the hood here. For example, if an application runs on a private network (you control the hardware), communications follow a publisher/subscriber model, and the application can tolerate a certain rate of data loss, then raw multicast may offer significant performance gains over any middleware based on TCP. If an application is distributed across very long distances and data order is not important, then a UDP-based protocol may offer advantages in terms of not stalling

to resend a missed packet. If TCP-based messaging is being used, then it is worth keeping in mind that many of its parameters (especially buffer sizes, slow start, and Nagle's algorithm) are configurable and the "out-of-the-box" settings are usually optimized for throughput rather than latency.[h]

Location. The physical constraint that information cannot propagate faster than the speed of light is a very real consideration when dealing with short time scales and/or long distances. The two largest stock exchanges, NASDAQ and NYSE, run their matching engines in data centers in Carteret and Mahwah, New Jersey, respectively. A ray of light takes 185 microseconds to travel the 55.4km distance between these two locations. Light in a glass fiber with a refractive index of 1.6 and following a slightly longer path (roughly 65 km) takes almost 350 microseconds to make the same one-way trip. Given the computations involved in trading decisions can now be made on time scales of 10 microseconds or less, signal propagation latency cannot be ignored.

Threading. Decomposing a problem into a number of threads that can be executed concurrently can greatly increase performance, especially in multicore systems, but in some cases it may actually be slower than a single-threaded solution.

Specifically, multithreaded code incurs overhead in the following three ways:

▸ When multiple threads operate on the same data, controls are required to ensure the data remains consistent. This may include acquisition of locks or implementations of read or write barriers. In multicore systems, these concurrency controls require that thread execution is suspended while messages are passed between cores. If a lock is already held by one thread, then other threads seeking that lock will need to wait until the first one is finished. If several threads are frequently accessing the same data, then there may be significant contention for locks.

▸ Similarly, when multiple threads operate on the same data, the data itself must be passed between cores. If several threads access the same data but each performs only a few computations on it, then the time required to move the data between cores may exceed the time spent operating on it.

▸ Finally, if there are more threads than cores, then the operating system must periodically perform a context switch in which the thread running on a given core is halted, its state is saved, and another thread is allowed to run. The cost of a context switch can be significant. If the number of threads far exceeds the number of cores, then context switching can be a significant source of delay.

Modern software engineering is built upon abstractions that allow programmers to manage the complexity of ever-larger systems. Abstractions do this by simplifying or generalizing some aspect of the underlying system.

In general, application design should use threads in a way that represents the inherent concurrency of the underlying problem. If the problem contains significant computation that can be performed in isolation, then a larger number of threads is called for. On the other hand, if there is a high degree of interdependency between computations or (worst case) if the problem is inherently serial, then a single-threaded solution may make more sense. In both cases, profiling tools should be used to identify excessive lock contention or context switching. Lock-free data structures (now available for several programming languages) are another alternative to consider.[i]

It is also worth noting the physical arrangement of cores, memory, and I/O may not be uniform. For example, on modern Intel microprocessors certain cores can interact with external I/O (for example, network interfaces) with much lower latency than others, and exchanging data between certain cores is faster than others. As a result, it may be advantageous explicitly to pin specific threads to specific cores.[j]

Hierarchical storage and cache misses. All modern computing systems use hierarchical data storage—a small amount of fast memory combined with multiple levels of larger (but slower) memory. Recently accessed data is cached so that subsequent access is faster. Since most applications exhibit a tendency to access the same memory multiple times in a short period, this can greatly increase performance. To obtain maximum benefit, however, the following three factors should be incorporated into application design:

▸ Using less memory overall (or at least in the parts of the application that are latency sensitive) increases the probability that needed data will be available in one of the caches. In particular, for especially latency-sensitive

applications, designing the app so that frequently accessed data fits within the CPU's caches can significantly improve performance. Specifications vary but Intel's Haswell microprocessors, for example, provide 32KB per core for L1 data cache and up to 40MB of shared L3 cache for the entire CPU.

▸ Repeated allocation and release of memory should be avoided if reuse is possible. An object or data structure that is allocated once and reused has a much greater chance of being present in a cache than one that is repeatedly allocated anew. This is especially true when developing in environments where memory is managed automatically as overhead caused by garbage collection of memory that is released can be significant.

▸ The layout of data structures in memory can have a significant impact on performance because of the architecture of caches in modern processors. While the details vary by platform and are outside the scope of this article, it is generally a good idea to prefer arrays as data structures over linked lists and trees and to prefer algorithms that access memory sequentially since these allow the hardware prefetcher (which attempts to load data preemptively from main memory into cache *before* it is requested by the application) to operate most efficiently. Note also that data that will be operated on concurrently by different cores should be structured so it is unlikely to fall in the same cache line (the latest Intel CPUs use 64-byte cache lines) to avoid cache-coherency contention.

A note on premature optimization. The optimizations just presented should be considered part of a broader design process that takes into account other important objectives including functional correctness, and maintainability. Keep in mind Knuth's quote about premature optimization being the root of all evil; even in the most performance-sensitive environments it is rare that a programmer should be concerned with determining the correct number of threads or the optimal data structure until empirical measurements indicate a specific part of the application is a hot spot. Focus instead should be on ensuring performance requirements are understood early in the design process and the system architecture is sufficiently decomposable to allow detailed measurement of latency as and when optimization becomes necessary. Moreover (and as discussed in the next section), the most useful optimizations may not be in the application code at all.

Changes in design. The optimizations presented so far have been limited to improving the performance of a system for a given set of functional requirements. There may also be opportunities to change the broader design of the system or even to change the functional requirements of the system in a way that still meets the overall objectives but significantly improves performance. Latency optimization is no exception. In particular, there are often opportunities to trade reduced efficiency for improved latency.

Three real-world examples of design trade-offs between efficiency and latency are presented here, followed by an example where the requirements themselves present the best opportunity for redesign.

Speculative precomputation. In certain cases trading efficiency for latency may be possible, especially in systems that operate well below their peak capacity. In particular, it may be advantageous to compute possible outputs in advance, especially when the system is idle most of the time but must react quickly when an input arrives.

A real-world example can be found in the systems used by some firms to trade stocks based on news such as earnings announcements. Imagine the market expects Apple to earn between $9.45 and $12.51 per share. The goal of the trading system, upon receiving Apple's actual earnings, would be to sell some number of shares of Apple stock if the earnings were below $9.45, buy some number of shares if the earnings were above $12.51, and do nothing if the earnings fall within the expected range. The act of buying or selling stocks begins with submitting an order to the exchange. The order consists of (among other things) an indicator of whether the client wishes to buy or sell, the identifier of the stock to buy or sell, the number of shares desired, and the price at which the client wishes to buy or sell. Throughout the afternoon leading up to Apple's announcement, the client would receive a steady stream of market-data messages that indicate the current price at which Apple's stock is trading.

A conventional implementation of this trading system would cache the market-price data and, upon receipt of the earnings data, decide whether to buy or sell (or neither), construct an order, and serialize that order to an array of bytes to be placed into the payload of a message and sent to the exchange.

An alternative implementation performs most of the same steps but does so on every market-data update rather than only upon receipt of the earnings data. Specifically, when each market-data update message is received, the application constructs two new orders (one to buy, one to sell) at the current prices and serializes each order into a message. The messages are cached but not sent. When the next market-data update arrives, the old order messages are discarded and new ones are created. When the earnings data arrives, the application simply decides which (if either) of the order messages to send.

The first implementation is clearly more efficient (it has a lot less wasted computation), but at the moment when latency matters most (that is, when the earnings data has been received), the second algorithm is able to send out the appropriate order message sooner. Note that this example presents application-level precomputation; there is an analogous process of branch prediction that takes place in pipelined processors that can also be optimized (via guided profiling) but is outside the scope of this article.

Keeping the system warm. In some low-latency systems long delays may occur between inputs. During these idle periods, the system may grow "cold." Critical instructions and data may be evicted from caches (costing hundreds of nanoseconds to reload), threads that would process the latency-sensitive input are context-switched out (costing tens of microseconds to resume), or CPUs may switch into power-saving states (costing a few milliseconds to exit). Each of these steps makes sense from an efficiency standpoint (why run a CPU at full power when nothing is happening?), but all of them impose latency penalties when the input data arrives.

In cases where the system may go for hours or days between input events there is a potential operational issue as well: configuration or environmental changes may have "broken" the system

[a] http://copter.ardupilot.com/
[b] http://queue.acm.org/detail.cfm?id=1028895/
[c] http://queue.acm.org/detail.cfm?id=971591/
[d] http://acuityads.com/real-time-bidding/
[e] http://home.web.cern.ch/about/accelerators/cern-neutrinos-gran-sasso/
[f] See https://tools.ietf.org/html/rfc1305/, https://tools.ietf.org/html/rfc5905/; http://www.nist.gov/el/isd/ieee/ieee1588.cfm/; and http://queue.acm.org/detail.cfm?id=2354406/
[g] See http://wireshark.org and http://tcpdump.org/
[h] http://queue.acm.org/detail.cfm?id=2539132/
[i] http://queue.acm.org/detail.cfm?id=2492433/
[j] http://queue.acm.org/detail.cfm?id=2513149/

in some important way, but will not be discovered until the event occurs—when it is too late to fix.

A common solution to both problems is to generate a continuous stream of dummy input data to keep the system "warm." The dummy data needs to be as realistic as possible to ensure it keeps the right data in the caches and that breaking changes to the environment are detected. The dummy data needs to be reliably distinguishable from legitimate data, though, to prevent downstream systems or clients from being confused.

Redundant processing. It is common in many systems to process the same data through multiple independent instances of the system in parallel, primarily for the improved resiliency that is conferred. If some component fails, the user will still receive the result needed. Low-latency systems gain the same resiliency benefits of parallel, redundant processing but can also use this approach to reduce certain kinds of variable latency.

All real-world computational processes of nontrivial complexity have some variance in latency even when the input data is the same. These variations can be caused by minute differences in thread scheduling, explicitly randomized behaviors such as Ethernet's exponential back-off algorithm, or other unpredictable factors. Some of these variations can be quite large: page faults, garbage collections, and network congestion can all cause occasional delays that are several orders of magnitude larger than the typical processing latency for the same input.

Running multiple, independent instances of the system, combined with a protocol that allows the end recipient to accept the first result produced and discard subsequent redundant copies, provides both the benefit of less-frequent outages and the avoidance of some of the larger delays.

Streaming processing and short circuits. Consider a news analytics system whose requirements are understood to be "build an application that can extract corporate earnings data from a press release document as quickly as possible." Separately, it was specified the press releases would be pushed to the system via FTP. The system was thus designed as two applications: one that received the document via FTP; and a second that parsed the document and extracted the earnings data. In the first version of this system, an open source FTP server was used as the first application, and the second application (the parser) assumed it would receive a fully formed document as input, so it would not start parsing the document until it had fully arrived.

Measuring the performance of the system showed that while parsing was typically completed in just a few milliseconds, receiving the document via FTP could take tens of milliseconds from the arrival of the first packet to the arrival of the last packet. Moreover, the earnings data was often present in the first paragraph of the document.

In a multistep process it may be possible for subsequent stages to start processing before prior stages have finished, sometimes referred to as *stream-oriented* or *pipelined processing*. This can be especially useful if the output can be computed from a partial input. Taking this into account, the developers reconceived their overall objective as "build a system that can deliver earnings data as quickly as possible to the client." This broader objective, combined with the understanding the press release would arrive via FTP and it was possible to extract the earnings data from the first part of the document (that is, before the rest of the document had arrived), led to a redesign of the system.

The FTP server was rewritten to forward portions of the document to the parser as they arrived rather than wait for the entire document. Likewise, the parser was rewritten to operate on a stream of incoming data rather than on a single document. The result was that in many cases the earnings data could be extracted within just a few milliseconds of the *start* of the arrival of the document. This reduced overall latency (as observed by the client) by several tens of milliseconds without the internal implementation of the parsing algorithm being any faster.

Conclusion

While latency requirements are common to a wide array of software applications, the financial trading industry and the segment of the news media that supplies it with data have an especially competitive ecosystem that produces challenging demands for low-latency distributed systems.

As with most engineering problems, building effective low-latency distributed systems starts with having a clear understanding of the problem. The next step is measuring actual performance and then, where necessary, making improvements. In this domain, improvements often require some combination of digging below the surface of common software abstractions and/or trading some degree of efficiency for improved latency. 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

Principles of Robust Timing over the Internet
Julien Ridoux and Darryl Veitch
http://queue.acm.org/detail.cfm?id=1773943

Online Algorithms in High-Frequency Trading
Jacob Loveless, Sasha Stoikov, and Rolf Waeber
http://queue.acm.org/detail.cfm?id=2534976f

Andrew Brook is the CTO of Selerity, a provider of real-time news, data, and content analytics. Previously he led development of electronic currency-trading systems at two large investment banks and launched a pre-dot-com startup to deliver AI-powered scheduling software to agile manufacturers.

DOI:10.1145/2755503

Article development led by acmqueue
queue.acm.org

An agile process implementation.

BY PHELIM DOWLING AND KEVIN MCGRATH

Using Free and Open Source Tools to Manage Software Quality

THE PRINCIPLES OF agile software development place more emphasis on individuals and interactions than on processes and tools. They steer us away from heavy documentation requirements and guide us along a path of reacting efficiently to change rather than sticking rigidly to a predefined plan. To support this flexible method of operation, it is important to have suitable applications to manage the team's activities. It is also essential to implement effective frameworks to ensure quality is being built into the product early and at all levels. With these concerns in mind and coming from a budget-conscious perspective, this article will explore the free and open source applications and tools used by one organization in its quest to build process and quality around its projects and products.

How a software development team implements process and handles its application life cycle management can have a huge bearing on the success of the product it is developing.[2] A lean and efficient mode of operation is the key, especially for early-stage start-ups. Many commercial-grade products and frameworks are available for integrating all aspects of your development process: requirements management, project planning, defect management, test case creation and execution, automated testing, load testing, and so on. But these tools can be expensive and may be too much of a burden for some organizations' idea of agility.

Other solutions exist that can support your agile philosophy and not make any dent at all in your hard-earned bottom line. An abundance of free and open source tools have been available for quite a while now. Many companies and organizations are employing open source tools to meet their business needs.[1] These applications often have large communities of active and engaged contributors documenting the features of the product and releasing enhancements on a regular basis. We recommend proper due diligence in examining the suitability of the tools for your needs and determining their track records of performance,

maintenance, and growth. If they pass that check, open source tools can more than meet the needs of small project outfits, funded research and development teams, early start-ups, and even larger-scale enterprises.

Project Management and Issue Tracking

The Telecommunications Software & Systems Group (TSSG)[5] previously used tools such as XPlanner for project planning and Bugzilla for defect management. While individually these tools were quite functional, the lack of integration in terms of gathering requirements, planning, logging defects, and so on became an obstacle to our desire for a more streamlined and agile way of working. Expecting developers to look at one system for bugs to be fixed and another to determine what features to work on was inefficient. The Verification and Validation (VnV) team made efforts to link defects with features but this was manual and laborious. Planning of daily and weekly activities was harder because of the standalone nature of the tools.

This difficulty led to the creation of a subteam of tech leads and process champions to review available tools that might be better suited to our needs. Their remit was to find a tool that would allow us to manage our activities across multiple projects efficiently, was adaptable to our fluid agile ethos, and had zero cost.

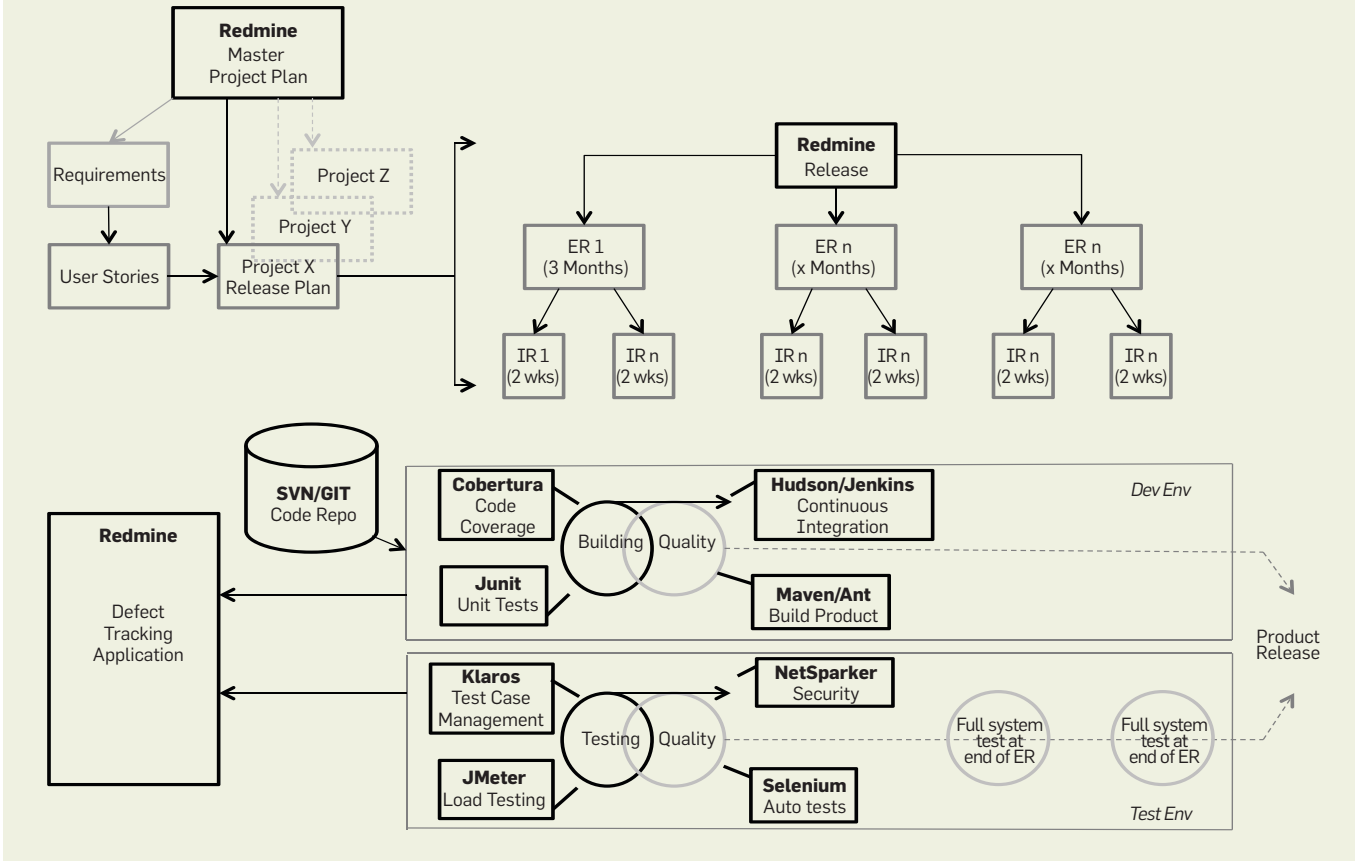
This eventually led to the rollout of Redmine as the organization's primary tool for project management and issue tracking. Redmine is an open source application and is released under the GNU General Public License v2. Written using the Ruby on Rails Web application framework, we have installed our implementation on an Ubuntu VM. Utilizing the open source Apache Web server, TSSG staff access the tool via any common browser. The application's MySQL database is backed up regularly because of the widespread use and critical stature of this tool within the organization. The integral nature of Redmine within the TSSG agile process along with the other tools we use is depicted in the accompanying figure.

At the requirements gathering stage all project requirements are documented in Redmine as user stories. These are initially stored in the project backlog before being allocated to specific engineering releases and biweekly iterations. The user stories are then broken down into subtasks and subfeatures so that distinct and measurable elements of work are planned and executed to achieve the overall goal. These pieces of work are allocated to individuals on the team to be completed within a specific time period.

The Redmine tool facilitates many of the other agile methods used by our engineering teams, including time management, defect tracking, email integration, task boards, sprint cards, and so on.

While there is an active community of developers working on Redmine and creating plugins of value, this is somewhat of a double-edged sword. We have found in the past that upgrading the core product can be problematic due to these plugins. The plugins are not always compatible with newer releases, leading to headaches during the up-

Open source and free tools mapped to the TSSG agile process.



grade and broken functionality for features that teams were previously using.

Code Repository and Continuous Integration

One of the first tasks for a project team is to set up a source code control framework. It is critical to maintain a central area for the team to check in code and have it backed up regularly. Tracking versions of your code will allow you to roll back to a previous revision if required. Spawning branches as the project grows and tagging builds to allow hot fixes are likely requirements. In TSSG, we have used locally hosted versions of both the Apache Subversion system and the Git repository, and both tools have met the needs of our projects.

Another handy integration point we have set up for projects is between Git and Redmine. This allows us to view the Git repository, commits, and stats directly from our project management system.

A concept we are currently exploring is a peer review of code before commit. This would especially apply to junior developers paired with more experienced colleagues. Tools such as Gerrit and Review Board are being examined for suitability in this regard. While it may bring extra overhead before commit we feel it can add extra quality to our products in the long run. Code reviews and peer programming are philosophies that TSSG promotes, but are too often overlooked due to time constraints. If these tools can facilitate code review and peer programming in a relatively seamless manner, we anticipate trialing their use on a subset of projects before pushing them out for general use.

A key element of our agile process is to build our code regularly. Continuous integration will execute all the unit and acceptance tests at build time, and makes regular builds available for verification on the test environment. In the TSSG we have used Hudson and more frequently Jenkins for this. Easily installed and configured, Jenkins builds can be triggered by code check-ins (to the repositories mentioned earlier) or at regular intervals. A user-friendly Web-based interface makes it easy to view a list of builds including the latest. You can view the changes made in any build, and a comprehensive overview of the test results for each build is also presented. Jenkins can also be configured to send email messages to team members on build pass and/or fail. Of course, under-the-hood open source tools such as Ant and Maven are actually generating the builds.

In the TSSG we promote the concept of test-driven development and, while the choice of tool for unit testing will vary depending on the technologies being used, JUnit is the most common. On top of this, we measure the percentage of our code that is covered by tests using tools such as Cobertura. A Cobertura plugin for Jenkins allows the code coverage to be analyzed from the Jenkins build Web page. This element of our process often generates the most heated discussions between the VnV team and the engineering teams. Our goal is a 90% level of code coverage, but there is always debate about the difficulty of achieving this target with certain technologies. Also, the time and resources required to reach the last few are often questioned in terms of return on investment. However, the team mentality promoted in our agile philosophy means that consensus is always reached.

Another aspect of the TSSG's agile process methodology is an independent code review team. This team examines the project's implementation of a build framework using the tools described in this section with a view to building quality into our code from the ground up. They may also advise the developers to implement code analysis using open source tools such as Checkstyle, Gmetrics, or CodeNarc.

Automated Testing

The majority of the projects in the TSSG start as funded research. However, as the project evolves and the solution stabilizes and matures, the product will generally take on a more commercial nature. As this stage approaches we have found it beneficial to implement Selenium automated test suites on the components developed to date.

Our backend implementation is Selenium 2.0 WebDriver running on a Windows server. The Selenium IDE is installed as a Firefox browser add-on for recording the test cases. To allow the results of a test suite to be emailed to the relevant team members we have installed two other pieces of freeware, MDaemon Free mail server and SendEmail, a lightweight command line email client.

The tests are executed in Firefox and recorded via the Selenium IDE. The tests are then saved on the server side. A single file to list all the individual test cases to be executed as part of a test suite is created. A batch file is developed to execute the entire suite and scheduled to run at desired intervals. The batch file calls the Selenium tool and tells it which suite to run (and on which browser) and where to save the results. The final step is to email the results to the distribution list.

Quite often you will find that other users have encountered roadblocks in automating elements of their tests. To overcome this they may have coded a 'user extension' for Selenium and made it openly available. Contained in a flat file configured in your local environment, these coded solutions provided by individual contributors can prove to be very useful. One of the earliest extensions we used was the beatnicClick function. This handy extension acted as a workaround for an issue with timeout failures on certain native click activities, allowing the test to execute the desired functionality seamlessly. Another example of a user extension we have implemented in our framework is for flow control logic. The ability to easily include `goto` statements and `while` loops has helped improve the robustness and coverage provided by our automated test suites.

The Selenium framework implemented in TSSG allows us to execute functions in a browser session, record those steps, and then replay them automatically. The test steps can then be extended to include checks for specific text or elements on the site. You can capture and store variables for reuse within the test suite. Pauses and waits can be added to replicate real user behavior. Screenshots can be captured. Data-driven testing is supported; that is, external files with lists of parameters can be fed into the tests. Other more advanced features include the use of JavaScript snippets, for example to manipulate data or strings.

Although the recording of the tests must be carried out in Firefox, the test execution supported by the Selenium WebDriver allows us to run the tests

against Firefox, IE, Chrome, Safari, and other common browsers.

On certain projects we encountered issues with Selenium not being able to interact with Windows-specific pop-up dialogues. While Selenium provides a feature to handle browser pop-ups (via getWindowHandles and switchTo.window), the framework cannot handle non-browser components. To deal with this we used another open source automation tool, iMacros. Using iMacros to handle the interaction with the external dialogue, the iMacro script was saved as a bookmark in the browser and called from Selenium, allowing seamless automation of the full test.

Selenium works well for our VnV test team, who utilize its record/playback authoring of tests with some extensions as described previously. But it can also be a powerful tool for developers who need to create tests in Java, Ruby, Python, and other languages using the Selenium API.

The most significant implementation of Selenium automated tests by TSSG was on the FeedHenry project. Working within an aggressive biweekly release cycle,[3] the comprehensive automated test suite maintained by the VnV team gave the fledgling company the confidence and space to release features for its enterprise mobile application platform every two weeks. RedHat's €63.5 million acquisition[4] of FeedHenry (www.feedhenry.com), which started as a small research project, is a vindication of both the TSSG's research and innovation model and the successful implementation of our agile process and tools.

A key element of our agile process is to build our code regularly. Continuous integration will execute all the unit and acceptance tests at build time, and makes regular builds available for verification on the test environment.

Load Testing

Performance monitoring is an important part of the software life cycle at TSSG as many of our products will go to live trial and full commercialization. We conduct load testing in two ways. Endurance testing involves evaluating a system's ability to handle a moderate workload for a longer period of time. Volume testing, on the other hand, subjects a system to a heavy load for a limited period of time. Both approaches have identified bottlenecks, bugs, and component limitations in certain projects.

Apache JMeter is an open source load-testing tool designed to load test functional behavior and measure performance. It was originally designed for testing Web applications but has since expanded to other tests. JMeter provides the control needed on the tests and uses multiple threads to emulate multiple users. While JMeter is not a browser, and does not function like one, it can emulate browser behavior in a number of ways—caching, cookie management, and header management.

JMeter includes an HTTP proxy which, when enabled, can be used to record scripts when testing websites. The idea is to point the browser to the JMeter proxy so that JMeter can sniff the requests made through the browser and store them under the specified controller in JMeter. The information derived can be used to create a test plan. JMeter will then play back the HTTP requests to the server, but we can simulate any number of users doing this. We also use Fiddler and Firebug to dig deeper into Web traffic when required.

Unfortunately, JMeter does not execute the JavaScript found in HTML pages. Nor does it render the HTML pages as a browser does (it is possible to view the response as HTML). So, depending on the Web application, it may be necessary to incorporate Selenium tests in a separate test run if you are testing the client-side business logic and use JMeter for the backend components. JMeter also has an extensive set of third-party plugins that extend its functionality with an array of graphs, new load delivery controllers, and other functions that act as add-ons to the original JMeter package.

Recently we have had a requirement to perform load tests across an Openfire IM server instance to test rules from a policy engine we are creating as part of a new project. In this scenario we used Mockaroo to generate a large number of test users with realistic characteristics. We then used Tsung test scripts to fire load at the test environment. Tsung's Erlang foundation means it uses fewer threads than JMeter, allowing larger-scale testing, but it is trumped by JMeter's superior GUI, which allows easy configuration of tests and analysis of results.

Test Case Management

The TSSG has chosen the Klaros test management system for creating and executing manual test cases. A com-

munity edition that is free for unlimited use provides all the functionality required to maintain manual test cases across multiple projects.

Windows and Linux versions are available, and we have installed our instance on a Windows server. The downloadable executable will install an Apache Tomcat application server and a file-based database Apache Derby (although this can be changed to more robust solutions such as MySQL or PostgreSql). Users can interact with the test case system via any common Web browser.

The free edition of Klaros supports multiple projects being maintained simultaneously. Within these projects the test case manager can build up comprehensive test steps, test cases, and test suites. Varying test environments can be maintained for each project, for example, for different OSes or application servers. Also, versioning of the product being tested is supported via the concept of tracking the system under test (SUT).

At execution time, the tester can decide to complete a single test or a full test suite. The test run can be suspended and resumed as needed. The result of each test (pass or fail) and any errors observed are captured at every step. After the test suite has been completed, an easy-to-read report is generated in various formats (pdf, html, csv). This report will display the date, test environment, system under test, a summary table of the test case results, a pie chart of the results, and a more detailed listing of the individual test cases with details such as execution times.

Also available in the community edition is integration with other open source testing tools. Redmine, described earlier in this article, can be configured as an issue management tool so that tickets can be created directly in the chosen bug tracking and project management tool. The continuous integration servers Hudson and Jenkins (described earlier) can also be integrated with Klaros. By installing the Klaros plugin in Hudson or Jenkins, the results of test cases generated at build time can be imported into Klaros via a REST interface. The import is tagged with the appropriate project, test environment, and SUT IDs and the results can then be reported and analyzed as required. Similarly, test results from test automation tools such as Selenium can also be imported and added to the manual test results.

Challenges

Supporting mobile usage of the apps and features produced by our projects is an essential element of our quality offering. The matrix of test scenarios can grow quite large when you want to cover phones and tablets across iOS, Android, and other platforms. Add in the spectrum of manufacturers with their own variations, and comprehensively validating app quality can be a challenge.

On certain projects we have dabbled in the pool of commercial offerings that provide a wide range of devices along with automated test functionality, Device Anywhere and Perfecto Mobile being the most popular. However, long-term use of these products was not financially sustainable on projects with tight budgets. We have made some attempts to automate and script tests against Android devices using ADB and monkeyrunner. MonkeyTalk also looks like a reasonable prospect for both iOS and Android. However, the resources required to build and maintain automated mobile test suites have not always proved justifiable. Mobile projects with agile requirements and aggressive deadlines have not yet proved suitable for large-scale automated testing, while some lack of confidence in emulator testing was also a concern. On the flip side, the advent of wearable technology such as glasses and watches broadens the horizon even further and may warrant a more strategic approach.

Another aspect we have identified for improvement in our process is security testing. Many of our projects are in the areas of e-health, finance, and personal digital data and have obvious requirements in terms of protecting user data. Protecting our own intellectual property and hosted services are also ongoing concerns. We have used Netsparkers community edition along with other free tools on certain projects to try to identify vulnerabilities to cross-site scripting, injection attacks, and so on. The code review team, mentioned previously, examines projects' use of static analysis tools such as Checkstyle and FindBugs to find flaws at that level. However, we feel leveraging existing TSSG expertise in this area can help provide a framework and process for a more bulletproof offering to the overall VnV process.

Conclusion

The TSSG operates without baseline funding and must compete nationally and internationally to finance projects. With a commitment to product excellence epitomized by the VnV teams' role on all projects, the use of free and open source tools to build layers of quality into our applications is a necessity. The many integration points between the tools described are an added benefit in pursuing a zero-cost solution to application life cycle management. The tools described in this article are only some of those used within the TSSG process. Monitoring the evolution of existing and new zero-cost tools is an ongoing task in our freeware and open source strategy. C

Related articles on queue.acm.org

Adopting DevOps Practices in Quality Assurance
James Roche
http://queue.acm.org/detail.cfm?id=2540984

Building Collaboration into IDEs
Li-Te Cheng, et al.
http://queue.acm.org/detail.cfm?id=966803

Breaking The Major Release Habit
Damon Poole
http://queue.acm.org/detail.cfm?id=1165768

References

1. Baldwin H. Reasons companies say yes to open source. (Jan. 6, 2014); http://bit.ly/1uJ7uYK
2. Clarke, P. and O'Connor, R. The influence of SPI on business success in software SMEs: An empirical study. *J. Systems and Software 85*, 10 (2012), 2356–2367.
3. Dowling, P. Successfully transitioning a research project to a commercial spin-out using an Agile software process. In *J. Software: Evolution and Process 26* (May 2014), 468–475; doi: 10.1002/smr.1611
4. Kepes, B. Red Hat acquires FeedHenry to move into the mobile space. *Forbes* (Sept. 18, 2014); http://onforb.es/1CXd4WF
5. TSSG; www.tssg.org/.

Phelim Dowling has 18 years of experience in the IT industry. He joined the TSSG Verification and Validation team in 2006 and has facilitated on projects such as Muzu, FeedHenry, Kodacall, and more recently the EU FP7-funded OpenLab initiative.

Kevin McGrath joined the commercialization team within Waterford Institute of Technology's TSSG in 2006 as a verification and validation engineer. He has worked on FeedHenry SWAGGER, COIN, SSGS, EGP, and Billing4Rent, an EU-funded project.

DOI:10.1145/2699414

Scientific discovery and engineering innovation requires unifying traditionally separated high-performance computing and big data analytics.

BY DANIEL A. REED AND JACK DONGARRA

Exascale Computing and Big Data

NEARLY TWO CENTURIES ago, the English chemist Humphrey Davy wrote "Nothing tends so much to the advancement of knowledge as the application of a new instrument. The native intellectual powers of men in different times are not so much the causes of the different success of their labors, as the peculiar nature of the means and artificial resources in their possession." Davy's observation that advantage accrues to those who have the most powerful scientific tools is no less true today. In 2013, Martin Karplus, Michael Levitt, and Arieh Warshel received the Nobel Prize in chemistry for their work in computational modeling. The Nobel committee said, "Computer models mirroring real life have become crucial for most advances made in chemistry today,"[17] and "Computers unveil chemical processes, such as a catalyst's purification of exhaust fumes or the photosynthesis in green leaves."

Whether describing the advantages of high-energy particle accelerators (such as the Large Hadron Collider and the 2013 discovery of the Higgs boson), powerful astronomy instruments (such as the Hubble Space Telescope, which yielded insights into the universe's expansion and dark energy), or high-throughput DNA sequencers and exploration of metagenomics ecology, ever-more powerful scientific instruments continually advance knowledge. Each such scientific instrument, as well as a host of others, is critically dependent on computing for sensor control, data processing, international collaboration, and access.

However, computing is much more than an augmenter of science. Unlike other tools, which are limited to particular scientific domains, computational modeling and data analytics are applicable to all areas of science and engineering, as they breathe life into the underlying mathematics of scientific models. They enable researchers to understand nuanced predictions, as well as shape experiments more efficiently. They also help capture and analyze the torrent of experimental data being produced by a new generation of scientific instruments made possible by advances in computing and microelectronics.

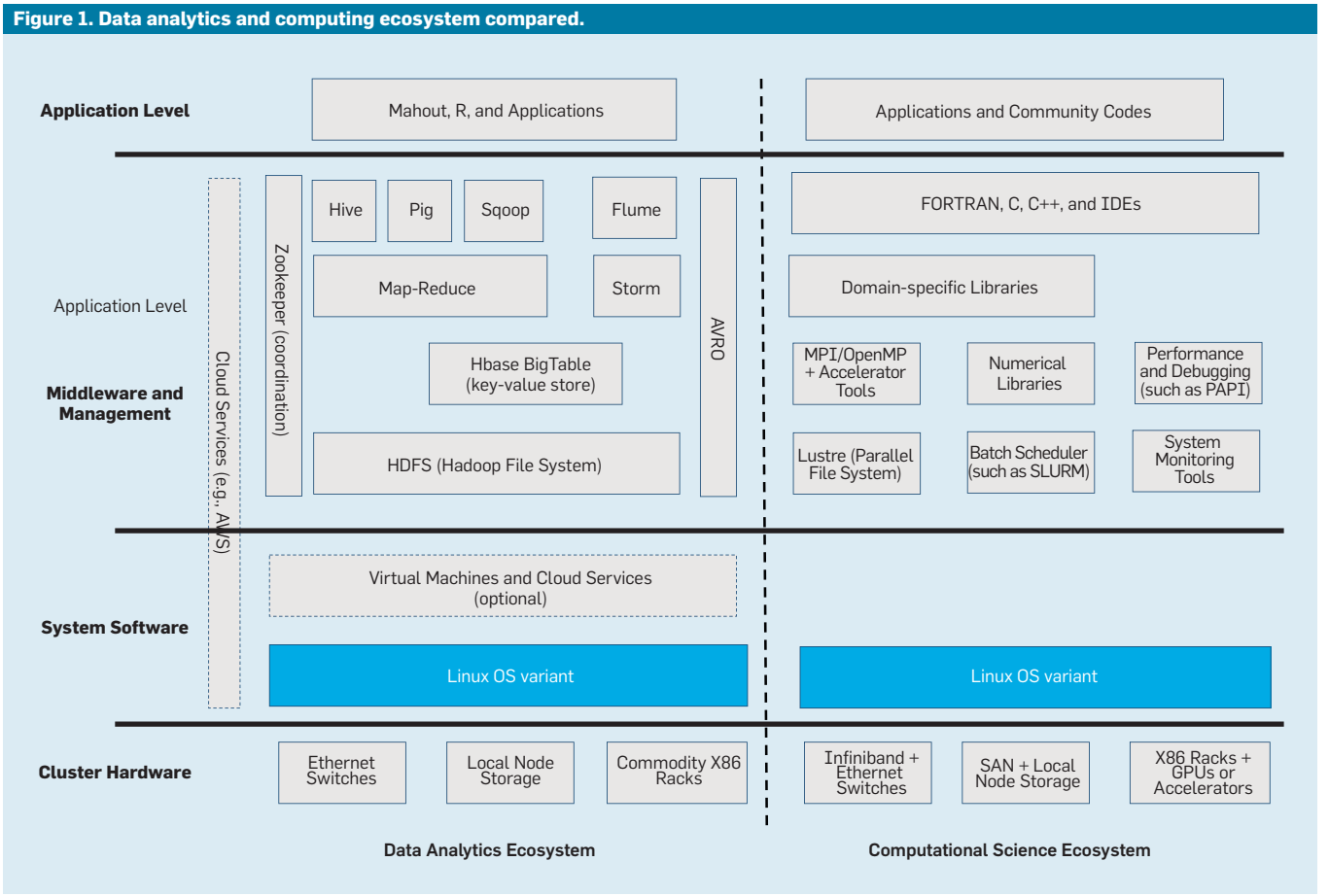
Computational modeling can illuminate the subtleties of complex mathematical models and advance science and engineering where time, cost, or safety precludes experimental assessment alone. Computational models of astrophysical phenomena, on temporal and spatial scales as di-

» key insights

- **The tools and cultures of high-performance computing and big data analytics have diverged, to the detriment of both; unification is essential to address a spectrum of major research domains.**
- **The challenges of scale tax our ability to transmit data, compute complicated functions on that data, or store a substantial part of it; new approaches are required to meet these challenges.**
- **The international nature of science demands further development of advanced computer architectures and global standards for processing data, even as international competition complicates the openness of the scientific process.**

ILLUSTRATION BY PETER BOLLINGER

Figure 1. Data analytics and computing ecosystem compared.



verse as planetary system formation, stellar dynamics, black hole behavior, galactic formation, and the interplay of baryonic and putative dark matter, have provided new insights into theories and complemented experimental data. Sophisticated climate models that capture the effects of greenhouse gases, deforestation, and other planetary changes have been key to understanding the effects of human behavior on the weather and climate change.

Computational science and engineering also enable multidisciplinary design and optimization, reducing prototyping time and costs. Advanced simulation has enabled Cummins to build better diesel engines faster and less expensively, Goodyear to design safer tires much more quickly, Boeing to build more fuel-efficient aircraft, and Procter & Gamble to create better materials for home products.

Similarly, "big data," machine learning, and predictive data analytics have been hailed as the fourth paradigm of science,[12] allowing researchers to extract insights from both scientific instruments and computational simulations. Machine learning has yielded new insights into health risks and the spread of disease via analysis of social networks, Web-search queries, and hospital data. It is also key to event identification and correlation in domains as diverse as high-energy physics and molecular biology.

As with successive generations of other large-scale scientific instruments, each new generation of advanced computing brings new capabilities, along with technical design challenges and economic trade-offs. Broadly speaking, data-generation capabilities in most science domains are growing more rapidly than compute capabilities, causing these domains to become data-intensive.[23] High-performance computers and big-data systems are tied inextricably to the broader computing ecosystem and its designs and markets. They also support national-security needs and economic competitiveness in ways that distinguish them from most other scientific instruments.

This "dual use" model, together with the rising cost of ever-larger computing and data-analysis systems, along with a host of new design challenges at massive scale, are raising new questions about advanced computing research investment priorities, design, and procurement models, as well as global collaboration and competition. This article examines some of these technical challenges, the interdependence of computational modeling and data analytics, and the global ecosystem and competition for leadership in advanced computing. We begin with a primer on the history of advanced computing.

Advanced Computing Ecosystems

By definition, an advanced computing system embodies the hardware, software, and algorithms needed to deliver the very highest capability at any given time. As in Figure 1, the computing and data analytics ecosystems share some attributes, notably reliance on open source software and the x86 hardware ecosystem. However, they differ markedly in their foci and technical approaches. As scientific research increasingly depends on both high-speed

computing and data analytics, the potential interoperability and scaling convergence of these two ecosystems is crucial to the future.

Scientific computing. In the 1980s, vector supercomputing dominated high-performance computing, as embodied in the eponymously named systems designed by the late Seymour Cray. The 1990s saw the rise of massively parallel processing (MPPs) and shared memory multiprocessors (SMPs) built by Thinking Machines, Silicon Graphics, and others. In turn, clusters of commodity (Intel/AMD x86) and purpose-built processors (such as IBM's BlueGene), dominated the previous decade.

Today, these clusters are augmented with computational accelerators in the form of coprocessors from Intel and graphical processing units (GPUs) from Nvidia; they also include high-speed, low-latency interconnects (such as Infiniband). Storage area networks (SANs) are used for persistent data storage, with local disks on each node used only for temporary files. This hardware ecosystem is optimized for performance first, rather than for minimal cost.

Atop the cluster hardware, Linux provides system services, augmented with parallel file systems (such as Lustre) and batch schedulers (such as PBS and SLURM) for parallel job management. MPI and OpenMP are used for internode and intranode parallelism, augmented with libraries and tools (such as CUDA and OpenCL) for coprocessor use. Numerical libraries (such as LAPACK and PETSc) and domain-specific libraries complete the software stack. Applications are typically developed in FORTRAN, C, or C++.

Data analytics. Just a few years ago, the very largest data storage systems contained only a few terabytes of secondary disk storage, backed by automated tape libraries. Today, commercial and research cloud-computing systems each contain many petabytes of secondary storage, and individual research laboratories routinely process terabytes of data produced by their own scientific instruments.

As with high-performance computing, a rich ecosystem of hardware and software has emerged for big-data analytics. Unlike scientific-computing clusters, data-analytics clusters are typically based on commodity Ethernet networks and local storage, with cost and capacity the primary optimization criteria. However, industry is now turning to FPGAs and improved network designs to optimize performance.

Atop this hardware, the Apache Hadoop[25] system implements a MapReduce model for data analytics. Hadoop includes a distributed file system (HDFS) for managing large numbers of large files, distributed (with block replication) across the local storage of the cluster. HDFS and HBase, an open-source implementation of Google's BigTable key-value store,[3] are the big-data analogs of Lustre for computational science, albeit optimized for different hardware and access patterns.

Atop the Hadoop storage system, tools (such as Pig[18]) provide a high-level programming model for the two-phase MapReduce model. Coupled with streaming data (Storm and Flume), graph (Giraph), and relational data (Sqoop) support, the Hadoop ecosystem is designed for data analysis. Moreover, tools (such as Mahout) enable classification, recommendation, and prediction via supervised and unsupervised learning. Unlike scientific computing, application development for data analytics often relies on Java and Web services tools (such as Ruby on Rails).

Scaling Challenges

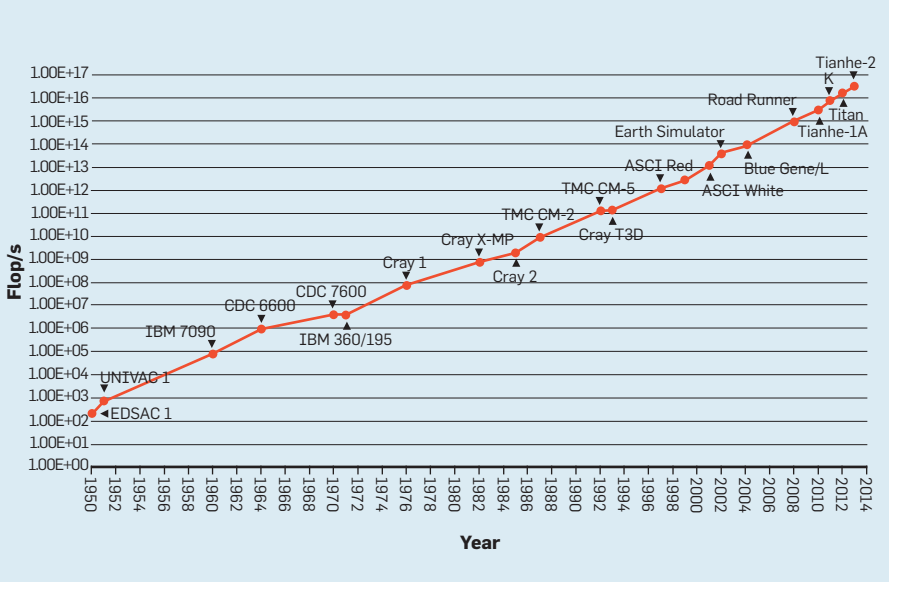
Given the rapid pace of technological change, leading-edge capability is a moving target. Today's smartphone computes as fast as yesterday's supercomputer, and today's personal music collection is as large as yesterday's enterprise-scale storage.

Lest this seem an exaggeration, the measured performance of an Apple iPhone 6 or Samsung Galaxy S5 on standard linear algebra benchmarks now substantially exceeds that of a Cray-1, which is widely viewed as the first successful supercomputer. That same smartphone has storage capacity rivaling the text-based content of a major research library.

Just a few years ago, teraflops (10^{12} floating point operations/second) and terabytes (10^{12} bytes of secondary storage) defined state-of-the-art advanced computing. Today, those same values represent a desk-side PC with Nvidia or Intel Xeon Phi accelerator and local storage. Advanced computing is now defined by multiple petaflops (10^{15} floating operations/second) supercomputing systems and cloud data centers with many petabytes of secondary storage.

Figure 2 outlines this exponential increase in advanced computing capability, based on the widely used High-Performance LINPACK (HPL) benchmark[6] and Top500 list of the world's fastest computers.[16] Although solution of dense linear systems of equations is no longer the best measure of delivered performance on complex scientific and engineering applications, this historical data illustrates how rapidly high-performance computing has evolved. Though high-performance computing has benefited from the same semiconductor advances as commodity com-

Figure 2. Advanced computing performance measured by the HPL benchmark.



puting, sustained system performance has improved even more rapidly due to increasing system size and parallelism.

The growth of personal, business, government, and scientific data has been even more dramatic and well documented. Commercial cloud providers are building worldwide networks of data centers, each costing hundreds of millions of dollars, to support Web search engines, social networks, and cloud services. Concurrently, the volume of scientific data produced annually now challenges the budgets of national research agencies.

As an example, Figure 3 outlines the exponential growth in the number of objects stored in Amazon's Simple Storage Service (S3). Atop such low-level services, companies (such as Netflix) implement advanced recommender systems to suggest movies to subscribers and then stream selections. Scientific researchers also increasingly explore these same cloud services and machine-learning techniques for extracting insight from scientific images, graphs, and text data. There are natural technical and economic synergies among the challenges facing data-intensive science and exascale computing, and advances in both are necessary for future scientific breakthroughs. Data-intensive science relies on the collection, analysis, and management of massive volumes of data, whether obtained from scientific simulations or experimental facilities. In each case, national and international investment in "extreme scale" systems will be necessary to analyze the massive volumes of data that are now commonplace in science and engineering.

Race to the future. For scientific and engineering computing, exascale (10^{18} operations per second) is the next proxy in the long trajectory of exponential performance increases that has continued for more than half a century. Likewise, large-scale data preservation and sustainability within and across disciplines, metadata creation and multidisciplinary fusion, and digital privacy and security define the frontiers of big data. This multifaceted definition of advanced computing encompasses more than simply quantitative measures of sustained arithmetic operation rates or storage capacity and analysis rates; it is also a relative term encompassing qualitative improvements in the usable capabilities of advanced computing systems at all scales. As such, it is intended to suggest a new frontier of practical, delivered capability to scientific and engineering researchers across all disciplines.

However, there are many challenges on the road to ever more advanced computing, including, but not limited to, system power consumption and environmentally friendly cooling, massive parallelism, and component failures, data and transaction consistency, metadata and ontology management, precision and recall at scale, and multidisciplinary data fusion and preservation.

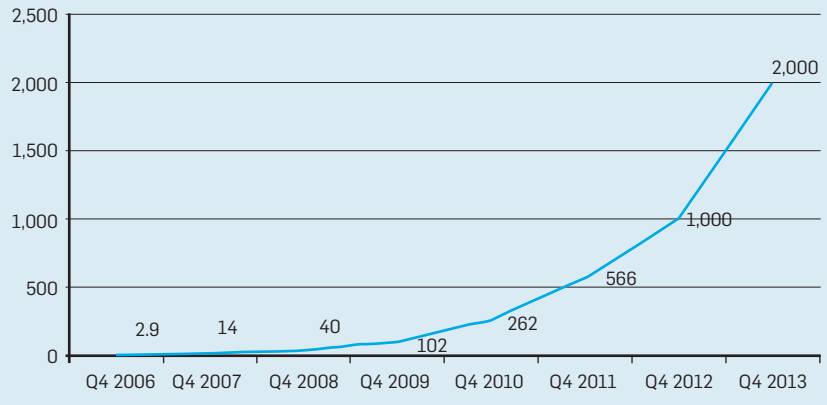
Above all, advanced computing systems must not become so arcane and complex that they and their services are unusable by all but a handful of experts. Open source toolkits (such as Hadoop, Mahout, and Giraph), along with a growing set of domain-specific tools and languages, have allowed many research groups to apply machine learning to large-scale scientific data without deep knowledge of machine-learning algorithms. The same is true of community codes for computational science modeling.

Hardware, software, data, and politics. Historically, high-performance computing advances have been largely dependent on concurrent advances in algorithms, software, architecture, and hardware that enable higher levels of floating-point performance for computational models. Advances today are also shaped by data-analysis pipelines, data architectures, and machine learning tools that manage large volumes of scientific and engineering data.

However, just as changes in scientific instrumentation scale bring new opportunities, they also bring new challenges, some technical but others organizational, cultural, and economic, and they are not self-similar across scales. Today, exascale computing systems cannot be produced in an affordable and reliable way or be subject to realistic engineering constraints on capital and operating costs, usability, and reliability. As the costs of advanced computing and data-analysis systems, whether commercial or scientific, have moved from millions to billions of dollars, design and decision processes have necessarily become more complex and fraught with controversy. This is a familiar lesson to those in high-energy physics and astronomy, where particle accelerators and telescopes have become planetary-scale instrumentation and the province of international consortia and global politics. Advanced computing is no exception.

The research-and-development costs to create an exascale computing system have been estimated by many experts to exceed one billion U.S. dollars, with an annual operating cost of tens of millions of dollars. Concurrently, there is growing recognition that governments and research agencies have substantially underinvested in data retention and management, as evinced by multibillion-dollar private-sector investments in big data and cloud computing. The largest commercial cloud data centers each cost more than $500 million to construct, and Google, Amazon, Microsoft, Facebook, and other companies operate global networks of such centers.

Figure 3. Growth of Amazon S3 objects.



Against this backdrop, U.S. support for basic research is at a decadal low, when adjusted for inflation,[2] and both the U.S. and the European Union continue to experience weak recoveries from the economic downturn of 2008. Further exacerbating the challenges, the global race for advanced computing hegemony is convolved with national-security desires, economic competitiveness, and the future of the mainstream computing ecosystem.

The shift from personal computers to mobile devices has also further raised competition between the U.S.-dominated x86 architectural ecosystem and the globally licensed ARM ecosystem. Concurrently, concerns about national sovereignty, data security, and Internet governance have triggered new competition and political concerns around data services and cloud-computing operations.

Despite these challenges, there is reason for cautious optimism. Every advance in computing technology has driven industry innovation and economic growth, spanning the entire spectrum of computing, from the emerging Internet of Things to ubiquitous mobile devices to the world's most powerful computing systems and largest data archives. These advances have also spurred basic and applied research in every domain of science.

Solving the myriad technical, political, and economic challenges will be neither easy nor even possible by tackling them in isolation. Rather, it will require coordinated planning across government, industry, and academia, commitment to data sharing and sustainability, collaborative research and development, and recognition that both competition and collaboration will be necessary for success. The future of big data and analytics should not be pitted against exascale computing; both are critical to the future of advanced computing and scientific discovery.

Computing technology is poised at important inflection points, at the very largest scale, or leading-edge high-performance computing, and the very smallest scale, or semiconductor processes.

Scientific and Engineering Opportunities

Researchers in the physical sciences and engineering have long been major users of advanced computing and computational models. The more recent adoption by the biological, environmental, and social sciences has been driven in part by the rise of big-data analytics. In addition, advanced computing is now widely used in engineering and advanced manufacturing. From understanding the subtleties of airflow in turbomachinery to chemical molecular dynamics for consumer products to biomass feedstock modeling for fuel cells, advanced computing has become synonymous with multidisciplinary design and optimization and advanced manufacturing.

Looking forward, only a few examples are needed to illustrate the deep and diverse scientific and engineering benefits from advanced computing:

Biology and biomedicine. Biology and biomedicine have been transformed through access to large volumes of genetic data. Inexpensive, high-throughput genetic sequencers have enabled capture of organism DNA sequences and made possible genome-wide association studies for human disease and human microbiome investigations, as well as metagenomics environmental studies. More generally, biological and biomedical challenges span sequence annotation and comparison, protein-structure prediction; molecular simulations and protein machines; metabolic pathways and regulatory networks; whole-cell models and organs; and organisms, environments, and ecologies;

High-energy physics. High-energy physics is both computational- and data-intensive. First-principles computational models of quantum chromodynamics provide numerical estimates and validations of the Standard Model. Similarly, particle detectors require the measurement of probabilities of "interesting" events in large numbers of observations (such as in 10^{16} or more particle collisions observed in a year). The Large Hadron Collider and its experiments necessitated creating a worldwide computing grid for data sharing and reduction, driving deployment of advanced networks and protocols, as well as a hierarchy of data repositories. All were necessary to identify the long-sought Higgs boson;

Climate science. Climate science is also critically dependent on the availability of a reliable infrastructure for managing and accessing large heterogeneous quantities of data on a global scale. It is inherently a collaborative and multidisciplinary effort requiring sophisticated modeling of the physical processes and exchange mechanisms

among multiple Earth realms—atmosphere, land, ocean, and sea ice—and comparing and validating these simulations with observational data from various sources, all collected over long periods. To encourage exploration, NASA has made climate and Earth science satellite data available through Amazon Web Services;

Cosmology and astrophysics. Cosmology and astrophysics are now critically dependent on advanced computational models to understand stellar structure, planetary formation, galactic evolution, and other interactions. These models combine fluid processes, radiation transfer, Newtonian gravity, nuclear physics, and general relativity (among other processes). Underlying them is a rich set of computational techniques based on adaptive mesh refinement and particle-in-cell, multipole algorithms, Monte Carlo methods, and smoothed-particle hydrodynamics;

Astronomy. Complementing computation, whole-sky surveys, and a new generation of automated telescopes are providing new insights. Rather than capture observational data to answer a known question, astronomers now frequently query extant datasets to discover previously unknown patterns and trends. Big-data reduction and unsupervised learning systems are an essential part of this exploratory image analysis;

Cancer treatment. Effective cancer treatment depends on early detection and targeted treatments via surgery, radiation, and chemotherapy. In turn, tumor identification and treatment planning are dependent on image enhancement, feature extraction and classification, segmentation, registration, 3D reconstruction, and quantification. These and other machine-learning techniques provide not only diagnostic validation, they are increasingly used to conduct comparative and longitudinal analysis of treatment regimes;

Experimental and computational materials science. Experimental and computational materials science is key to understanding materials properties and engineering options; for example, neutron scattering allows researchers to understand the structure and properties of materials, macromolecular and biological systems, and the fundamental physics of the neutron by providing data on the internal structure of materials from the atomic scale (atomic positions and excitations) up to the mesoscale (such as the effects of stress);

It is important for all of computer science to design algorithms that communicate as little as possible, ideally attaining lower bounds on the amount of communication required.

Steel production. Steel production via continuous casting accounts for an important fraction of global energy consumption and greenhouse gases production. Even small improvements to this process would have profound societal benefits and save hundreds of millions of dollars. High-performance computers are used to improve understanding of this complex process via comprehensive computational models, as well as to apply those models to find operating conditions to improve the process; and

Text and data mining. The explosive growth of research publications has made finding and tracking relevant research increasingly difficult. Beyond the volume of text, principles have different or similar names across domains. Text classification, semantic graph visualization tools, and recommender systems are increasingly being used to identify relevant topics and suggest relevant papers for study.

There are two common themes across these science and engineering challenges. The first is an extremely wide range of temporal and spatial scales and complex, nonlinear interactions across multiple biological and physical processes. These are the most demanding of computational simulations, requiring collaborative research teams, along with the very largest and most capable computing systems. In each case, the goal is predictive simulation, or gleaning insight that tests theories, identifies subtle interactions, and guides new research.

The second theme is the enormous scale and diversity of scientific data and the unprecedented opportunities for data assimilation, multidisciplinary correlation, and statistical analysis. Whether in biological or physical sciences, engineering or business, big data is creating new research needs and opportunities.

Technical Challenges in Advanced Computing

The scientific and engineering opportunities made possible through advanced computing and data analytics

are deep, but the technical challenges in designing, constructing, and operating advanced computing and data-analysis systems of unprecedented scale are just as daunting. Although cloud-computing centers and exascale computational platforms are seemingly quite different, as discussed earlier, the underlying technical challenges of scale are similar, and many of the same companies and researchers are exploring dual-use technologies applicable to both.

In a series of studies over the past five years, the U.S. Department of Energy identified 10 research challenges[10,15,24] in developing a new generation of advanced computing systems, including the following, augmented with our own comparisons with cloud computing:

Energy-efficient circuit, power, and cooling technologies. With current semiconductor technologies, all proposed exascale designs would consume hundreds of megawatts of power. New designs and technologies are needed to reduce this energy requirement to a more manageable and economically feasible level (such as 20MW–40MW) comparable to that used by commercial cloud data centers;

High-performance interconnect technologies. In the exascale-computing regime, the energy cost to move a datum will exceed the cost of a floating-point operation, necessitating very energy efficient, low-latency, high-bandwidth interconnects for fine-grain data exchanges among hundreds of thousands of processors. Even with such designs, locality-aware algorithms and software will be needed to maximize computation performance and reduce energy needs;

Driven by cost necessity, commercial cloud computing systems have been built with commodity Ethernet interconnects and adopted a bulk synchronous parallel computation model. Although this approach has proven effective, as evidenced by widespread adoption of MapReduce toolkits (such as Hadoop), a lower-cost, convergence interconnect would benefit both computation and data-intensive platforms and open new possibilities for fine-grain data analysis.

Advanced memory technologies to improve capacity. Minimizing data movement and minimizing energy use are also dependent on new memory technologies, including processor-in-memory, stacked memory (Micron's HMC is an early example), and nonvolatile memory approaches. Although the particulars differ for computation and data analysis, algorithmic determinants of memory capacity will be a significant driver of overall system cost, as the memory per core for very large systems will necessarily be smaller than in current designs;

Scalable system software that is power and failure aware. Traditional high-performance computing software has been predicated on the assumption that failures are infrequent; as we approach exascale levels, systemic resilience in the face of regular component failures will be essential. Similarly, dynamic, adaptive energy management must become an integral part of system software, for both economic and technical reasons.

Cloud services for data analytics, given their commercial quality-of-service agreements, embody large numbers of resilience techniques, including geo-distribution, automatic restart and failover, failure injection, and introspective monitoring; the Netflix "Simian Army"[a] is illustrative of these techniques.

Data management software that can handle the volume, velocity, and diversity of data. Whether computationally generated or captured from scientific instruments, efficient in situ data analysis requires restructuring of scientific workflows and applications, building on lessons gleaned from commercial data-analysis pipelines, as well as new techniques for data coordinating, learning, and mining. Without them, I/O bottlenecks will limit system utility and applicability;

Programming models to express massive parallelism, data locality, and resilience. The widely used communicating sequential process model, or MPI programming, places the burden of locality and parallelization on application developers. Exascale computing systems will have billion-way parallelism and frequent faults. Needed are more expressive programming models able to deal with this behavior and simplify the developer's efforts while supporting dynamic, fine-grain parallelism.

Much can be learned from Web and cloud services where abstraction layers and domain-specific toolkits allow developers to deploy custom execution environments (virtual machines) and leverage high-level services for reduction of complex data. The scientific computing challenge is retaining expressivity and productivity while also delivering high performance.

Reformulation of science problems and refactoring solution algorithms. Many thousands of person-years have been invested in current scientific and engineering codes and in data mining and learning software. Adapting scientific codes to billion-way parallelism will require redesigning, or even reinventing, the algorithms and potentially reformulating the science problems. Integrating data-analytics software and tools with computation is equally daunting; programming languages and models differ, as do the communities and cultures. Understanding how to do these things efficiently and effectively will be key to solving mission-critical science problems;

Ensuring correctness in the face of faults, reproducibility, and algorithm verification. With frequent transient and permanent faults, lack of reproducibility in collective communication, and new mathematical algorithms with limited verification, computation validation and correctness assurance will be much more important for the next generation of massively parallel systems, whether optimized for scientific computing, data analysis, or both;

Mathematical optimization and uncertainty quantification for discovery, design, and decision. Large-scale computations are themselves experiments that probe the sample space of numerical models. Understanding the sensitivity of computational predictions to model inputs and assumptions, particularly when they involve complex, multidisciplinary applications requires new tools and techniques for application validation and assessment. The equally important analogs in large-scale data analytics and machine learning are precision (the fraction of retrieved data that is relevant) and recall (the fraction of relevant data retrieved); and

a http://techblog.netflix.com/2011/07/netflix-simian-army.html

Software engineering and supporting structures to enable productivity. Although programming tools, compilers, debuggers, and performance-enhancement tools shape research productivity for all computing systems, at scale, application design and management for reliable, efficient, and correct computation is especially daunting. Unless researcher productivity increases, the time to solution may be dominated by application development, not computation.

Similar hardware and software studies[1,14] chartered by the U.S. Defense Advanced Research Projects Agency identified the following challenges, most similar to those cited by the Department of Energy studies:

Energy-efficient operation. Energy-efficient operation to achieve desired computation rates subject to overall power dissipation;

Memory capacity. Primary and secondary memory capacity and access rates, subject to power constraints;

Concurrency and locality. Concurrency and locality to meet performance targets while allowing some threads to stall during long-latency operations;

Resilience. Resilience, given large component counts, shrinking silicon feature sizes, low-power operation, and transient and permanent component failures;

Application scaling. Application scaling subject to memory capacity and communication latency constraints;

Managing parallelism. Expressing and managing parallelism and locality in system software and portable programming models, including runtime systems, schedulers, and libraries; and

Software tools. Software tools for performance tuning, correctness assessment, and energy management.

Moreover, a 2011 study by the U.S. National Academy of Sciences (NAS)[9] suggested that, barring a breakthrough, the exponential increases in performance derived from shrinking semiconductor feature size and architectural innovation are nearing an end. This study, along with others, suggests computing technology is poised at important inflection points, at the very largest scale, or leading-edge high-performance computing, and the very smallest scale, or semiconductor processes. The computing community remains divided on possible approaches, with strong believers that technical obstacles limiting extension of current approaches will be overcome and others who believe more radical technology and design approaches (such as quantum and superconducting devices) may be required.

Hardware and architecture challenges. Although a complete description of the hardware and software technical challenges just outlined is beyond the scope of this article, review of a selected subset is useful to illuminate the depth and breadth of the problems and their implications for the future of both advanced computing and the broader deployment of next-generation consumer- and business-computing technologies.

Post-Dennard scaling. For decades, Moore's "law" has held true due to the hard work and creativity of a great many people, as well as many billions of dollars of investment in process technology and silicon foundries. It has also rested on the principle of Dennard scaling,[5,13] providing a recipe for shrinking transistors and yielding smaller circuits with the same power density. Decreasing a transistor's linear size by a factor of two thus reduced power by a factor of four, or with both voltage and current halving.

Although transistor size continues to shrink, with 22-nanometer feature size now common, transistor power consumption no longer decreases accordingly. This has led to limits on chip clock rates and power consumption, along with design of multicore chips and the rise of dark silicon—chips with more transistors than can be active simultaneously due to thermal and power constraints.[8]

These semiconductor challenges have stimulated a rethinking of chip design, where the potential performance advantage of architectural tricks—superpipelining, scoreboarding, vectorization, and parallelization—must be balanced against their energy consumption. Simpler designs and function-specific accelerators often yield a better balance of power consumption and performance. This architectural shift will be especially true if the balance of integer, branch, and floating-point operations shifts to support in situ data analysis and computing.

Chip power limits have stimulated great interest in the ARM processor ecosystem. Because ARM designs were optimized for embedded and mobile devices, where limited power consumption has long been a design driver, they have simpler pipelines and instruction decoders than x86 designs.

In this new world, hardware/software co-design becomes de rigeur, with devices and software systems interdependent. The implications are far fewer general-purpose performance increases, more hardware diversity, elevation of multivariate optimization (such as power, performance, and reliability) in programming models, and new system-software-resource-management challenges.

Resilience and energy efficiency at scale. As advanced computing and data analysis systems grow ever larger, the assumption of fully reliable operation becomes much less credible. Although the mean time before failure for individual components continues to increase incrementally, the large overall component count for these systems means the systems themselves will fail more frequently. To date, experience has shown failures can be managed but only with improved techniques for detecting and understanding component failures.

Data from commercial cloud data centers suggests some long-held assumptions about component failures and lifetimes are incorrect.[11,20–22] A 2009 Google study[22] showed DRAM error rates were orders-of-magnitude higher than previously reported, with over 8% of DIMMs affected by errors in a year. Equally surprising, these were hard errors, rather than soft, correctable (via error-correcting code) errors.

In addition to resilience, scale also brings new challenges in energy management and thermal dissipation. Today's advanced computing and data-analysis systems consume megawatts of power, and cooling capability and peak power loads limit where many systems can be placed geographically. As commercial cloud operators have learned, energy infrastructure and power are a substantial fraction of total system cost at scale, necessitating new infrastructure approaches and operating models, including low-power designs,

cooling approaches, energy accountability, and operational efficiencies.

Software and algorithmic challenges. Many of the software and algorithmic challenges for advanced computing and big-data analytics are themselves consequences of extreme system scale. As noted earlier, advanced scientific computing shares many of the scaling problems of Web and cloud services but differs in its price-performance optimization balance, emphasizing high levels of performance, whether for computation or for data analysis. This distinction is central to the design choices and optimization criteria.

Given the scale and expected error rates of exascale computing systems, design and implementation of algorithms must be rethought from first principles, including exploration of global synchronization-free (or at least minimal) algorithms, fault-oblivious and error-tolerant algorithms, architecture-aware algorithms suitable for heterogeneous and hierarchical organized hardware, support for mixed-precision arithmetic, and software for energy-efficient computing.

Locality and scale. As noted earlier, putative designs for extreme-scale computing systems are projected to require billion-way computational concurrency, with aggressive parallelism at all system levels. Maintaining load balance on all levels of a hierarchy of algorithms and platforms will be the key to efficient execution. This will likely require dynamic, adaptive runtime mechanisms[4] and self-aware resource allocation to tolerate not only algorithmic imbalances but also variability in hardware performance and reliability.

In turn, the energy costs and latencies for communication will place an even greater premium on computation locality than today. Inverting long-held models, arithmetic operations will be far less energy intensive and more efficient than communication. Algorithmic complexity is usually expressed in terms of number of operations performed rather than quantity of data movement to memory. This is directly opposed to the expected costs of computation at large scale, where memory movement will be very expensive and operations will be nearly free, an issue of importance to both floating-point-intensive and data-analysis algorithms.

Programming models and tools are perhaps the biggest point of divergence between the scientific-computing and big-data ecosystems.

The temporal cost of data movement will challenge traditional algorithmic design approaches and comparative optimizations, making redundant computation sometimes preferable to data sharing and elevating communication complexity to parity with computation. It is therefore important for all of computer science to design algorithms that communicate as little as possible, ideally attaining lower bounds on the amount of communication required. It will also require models and methods to minimize and tolerate (hide) latency, optimize data motion, and remove global synchronization.

Adaptive system software. Resource management for today's high-performance computing systems remains rooted in a deus ex machina model, with coordinated scheduling and tightly synchronized communication. However, extreme scale, hardware heterogeneity, system power, and heat-dissipation constraints and increased component failure rates influence not only the design and implementation of applications, they also influence the design of system software in areas as diverse as energy management and I/O. Similarly, as the volume of scientific data grows, it is unclear if the traditional file abstractions and parallel file systems used by technical computing will scale to trans-petascale data analysis.

Instead, new system-software and operating-system designs will need to support management of heterogeneous resources and non-cache-coherent memory hierarchies, provide applications and runtime with more control of task scheduling policies, and manage global namespaces. They must also expose mechanisms for finer measurement, prediction, and control of power management, allowing schedulers to map computations to function-specific accelerators and manage thermal envelopes and application energy profiles. Commercial cloud providers have already faced many of these problems, and their experience in large-scale resource management has much to offer the scientific computing ecosystem.

Parallel programming support. As the diversity, complexity, and scale of advanced computing hardware has increased, the complexity and difficulty of

developing applications has increased as well, with many operating functions now subsumed by applications. Application complexity has been further exacerbated by the increasingly multidisciplinary nature of applications that combine algorithms and models spanning a range of spatiotemporal scales and algorithmic approaches.

Consider the typical single-program multiple-data parallel-programming or bulk-synchronous parallel model, where application data is partitioned and distributed across the individual memories or disks of the computation nodes, and the nodes share data via network message passing. In turn, the application code on each node manages the local, multilevel computation hierarchy—typically multiple, multithreaded, possibly heterogeneous cores, and (often) a GPU accelerator—and coordinates I/O, manages application checkpointing, and oversees power budgets and thermal dissipation. This daunting level of complexity and detailed configuration and tuning makes developing robust applications an arcane art accessible to only a dedicated and capable few.

Ideally, future software design, development, and deployment will raise the abstraction level and include performance and correctness in mind at the outset rather than ex situ. Beyond more performance-aware design and development of applications based on integrated performance and correctness models, these tools must be integrated with compilers and runtime systems, provide more support for heterogeneous hardware and mixed programming models, and provide more sophisticated data processing and analysis.

Programming models and tools are perhaps the biggest point of divergence between the scientific-computing and big-data ecosystems. The latter emphasizes simple abstractions (such as key-value stores and MapReduce), along with semantics-rich data formats and high-level specifications. This has allowed many developers to create complex machine-learning applications with little knowledge of the underlying hardware or system software. In contrast, scientific computing has continued to rely largely on traditional languages and libraries.

New programming languages and models, beyond C and FORTRAN, will help. Given the applications software already in place for technical computing, a radical departure is not realistic. Programming features found in new languages (such as Chapel and X10) have already had an indirect effect on existing program models. Existing programming models (such as OpenMP) have already benefited through recent extensions for, say, task parallelism, accelerators, and thread affinity.

Domain-specific languages (DSLs) are languages that specialize to a particular application domain, representing a means of extending the existing base-language by hosting DSL extensions. Embedded DSLs are a pragmatic way to exploit the sophisticated analysis and transformation capabilities of the compilers for standard languages. The developer writes the application using high-level primitives a compiler will transform into efficient low-level code to optimize the performance on the underlying platform.

Algorithmic and mathematics challenges. Exascale computing will put greater demand on algorithms in at least two areas: the need for increasing amounts of data locality to perform computations efficiently and the need to obtain much higher levels of fine-grain parallelism, as high-end systems support increasing numbers of compute threads. As a consequence, parallel algorithms must adapt to this environment, and new algorithms and implementations must be developed to exploit the computational capabilities of the new hardware.

Significant model development, algorithm redesign, and science-application reimplementation, supported by (an) exascale-appropriate programming model(s), will be required to exploit the power of exascale architectures. The transition from current sub-petascale and petascale computing to exascale computing will be at least as disruptive as the transition from vector to parallel computing in the 1990s.

Economic and Political Challenges

The technical challenges of advanced computing and big-data analytics are shaped by other elements of the broader computing landscape. In particular, powerful smartphones and cloud computing services are rapidly displacing the PC and local servers as the computing standard. This shift has also triggered international competition for industrial and business advantage, with countries and regions investing in new technologies and system deployments.

Computing ecosystem shifts. The Internet and Web-services revolution is global, and U.S. influence, though

ILLUSTRATION BY PETER BOLLINGER

substantial, is waning. Notwithstanding Apple's phenomenal success, most smartphones and tablets are now designed, built, and purchased globally, and the annual sales volume of smartphones and tablets exceeds that of PCs and servers.

This ongoing shift in consumer preferences and markets is accompanied by another technology shift. Smartphones and tablets are based on energy-efficient microprocessors—a key component of proposed exascale computing designs—and systems-on-a-chip (SoCs) using the ARM architecture. Unlike Intel and AMD, which design and manufacture the x86 chips found in today's PCs and most leading-edge servers and HPC systems, ARM does not manufacture its own chips. Rather, it licenses its designs to others, who incorporate the ARM architecture into custom SoCs that are manufactured by global semiconductor foundries like Taiwan's TSMC.

International exascale projects. The international competition surrounding advanced computing mixes concern about economic competitiveness, shifting technology ecosystems (such as ARM and x86), business and technical computing (such as cloud computing services and data centers), and scientific and engineering research. The European Union, Japan, China, and U.S. have all launched exascale computing projects, each with differing emphasis on hardware technologies, system software, algorithms, and applications.

European Union. The European Union (EU) announced the start of its exascale research program in October 2011 with €25 million in funding for three complementary research projects in its Framework 7 effort. The Collaborative Research into Exascale Systemware, Tools and Applications (CRESTA), Dynamical Exascale Entry Platform (DEEP), and Mont-Blanc projects will each investigate different exascale challenges using a co-design model spanning hardware, system software, and software applications. This initiative represents Europe's first sustained investment in exascale research.

CRESTA brings together four European high-performance computing centers: Edinburgh Parallel Computing Centre (project lead), the High Performance Computing Center Stuttgart, Finland's IT Center for Science Ltd., and Partner Development Center Sweden, as well as the Dresden University of Technology, which will lend expertise in performance optimization. In addition, the CRESTA team also includes application professionals from European science and industry, as well as HPC vendors, including HPC tool developer Allinea and HPC vendor Cray. CRESTA focuses on the use of applications as co-design drivers for software development environments, algorithms and libraries, user tools, and underpinning and crosscutting technologies.

The Mont-Blanc project, led by the Barcelona Supercomputing Center, brings together European technology providers ARM, Bull, Gnodal, and major supercomputing organizations involved with the Partnership for Advanced Computing in Europe (PRACE) project, including Juelich, Leibniz-Rechenzentrum, or LRZ, GENCI, and CINECA. The project intends to deploy a first-generation HPC system built from energy-efficient embedded technologies and conduct the research necessary to achieve exascale performance with energy-efficient designs.

DEEP, led by Forschungszentrum Juelich, seeks to develop an exascale-enabling platform and optimization of a set of grand-challenge codes. The system is based on a commodity cluster and accelerator design—Cluster Booster Architecture—as a proof-of-concept for a 100 petaflop/s PRACE production system. In addition to the lead partner, Juelich, project partners include Intel, ParTec, LRZ, Universität Heidelberg, German Research School for Simulation Sciences, Eurotech, Barcelona Supercomputing Center, Mellanox, École Polytechnique Fédérale de Lausanne, Katholieke Universiteit Leuven, Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, the Cyprus Institute, Universität Regensburg, CINECA, a consortium of 70 universities in Italy, and Compagnie Générale de Géophysique-Veritas.

Japan. In December 2013, the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) selected RIKEN to develop and deploy an exascale system by 2020. Selection was based on its experience developing and operating the K computer, which, at 10 petaflop/s, was ranked the fastest supercomputer in the world in 2011. Estimated to cost ¥140 billion ($1.38 billion), the exascale system design will be based on a combination of general-purpose processors and accelerators and involve three key Japanese computer vendors—Fujitsu, Hitachi, and NEC—as well as technical support from the University of Tokyo, University of Tsukuba, Tokyo Institute of Technology, Tohoku University, and RIKEN.

China. China's Tianhe-2 system is the world's fastest supercomputer today. It contains 16,000 nodes, each with two Intel Xeon processors and three Intel Xeon Phi coprocessors. It also contains a proprietary high-speed interconnect, called TH Express-2, designed by the National University for Defense Technology (NUDT). NUDT conducts research on processors, compilers, parallel algorithms, and systems. Based on this work, China is expected to produce a 100-petaflop/s systems in 2016 built entirely from Chinese-made chips, specifically the Shen-Wei processor, and interconnects. Tianhe-2 was to be upgraded from a peak of 55 petaflop/s to 100 petaflop/s in 2015, but the U.S. Department of Commerce has restricted exports of Intel processors to NUDT, the National Supercomputing Center in Changsha, National Supercomputing Center in Guangzhou, and the National Supercomputing Center in Tianjin due to national-security concerns.

U.S. Historically, the U.S. Networking and Information Technology Research and Development program has spanned several research missions and agencies, with primary leadership by the Department of Energy (DOE), Department of Defense (DoD), and National Science Foundation (NSF). DOE is today the most active deployer of high-performance computing systems and developer of plans for exascale computing. In contrast, NSF and DoD have focused more on broad cyberinfrastructure and enabling-technologies research, including research cloud services and big-data analytics. Although planning continues, the U.S. has not yet mounted an advanced computing initiative similar to those under way in Europe and Japan.

International collaboration. Although global competition for advanced computing and data-analytics leadership continues, there is active international collaboration. The International Exascale Software Project (IESP) is one such example in advanced computing. With seed funding from governments in Japan, the E.U., and the U.S., as well as supplemental contributions from industry stakeholders, IESP was formed to empower ultra-high-resolution and data-intensive science and engineering research through 2020.

In a series of meetings, the international IESP team developed a plan for a common, high-quality computational environment for petascale/exascale systems. The associated roadmap for software development would take the community from its current position to exascale computing.[7]

Conclusion

Computing is at a profound inflection point, economically and technically. The end of Dennard scaling and its implications for continuing semiconductor-design advances, the shift to mobile and cloud computing, the explosive growth of scientific, business, government, and consumer data and opportunities for data analytics and machine learning, and the continuing need for more-powerful computing systems to advance science and engineering are the context for the debate over the future of exascale computing and big data analysis. However, certain things are clear:

Big data and exascale. High-end data analytics (big data) and high-end computing (exascale) are both essential elements of an integrated computing research-and-development agenda; neither should be sacrificed or minimized to advance the other;

Algorithms, software, applications. Research and development of next-generation algorithms, software, and applications is as crucial as investment in semiconductor devices and hardware; historically the research community has underinvested in these areas;

Information technology ecosystem. The global information technology ecosystem is in flux, with the transition to a new generation of low-power mobile devices, cloud services, and rich data analytics; and

Private and global research. Private-sector competition and global-research collaboration are both necessary to address design, test, and deploy exascale-class computing and data-analysis capabilities.

There are great opportunities and great challenges in advanced computing, in both computation and data analysis. Scientific discovery via computational science and data analytics is truly the "endless frontier" about which Vannevar Bush spoke so eloquently in 1945. The challenges are for all of computer science to sustain the research, development, and deployment of the high-performance computing infrastructure needed to enable those discoveries.

Acknowledgments

We are grateful for insights and perspectives we received from the DARPA and DOE exascale hardware, software, and application study groups. We also acknowledge the insightful comments and suggestions from the reviewers of earlier drafts of this article. Daniel A. Reed acknowledges support from the National Science Foundation under NSF grant ACI-1349521. Jack Dongarra acknowledges support from the National Science Foundation under NSF grant ACI-1339822 and by the Department of Energy under DOE grant DE-FG02-13ER26151.

References

1. Amarasinghe, S. et al. *Exascale Software Study: Software Challenges in Extreme-Scale Systems.* Defense Advanced Research Projects Agency, Arlington, VA, 2009; http://www.cs.rice.edu/~vs3/PDF/Sarkar-ACS-July-2011-v2.pdf
2. American Association for the Advancement of Science. *Guide to R&D Funding - Historical Data.* AAAS, Washington, D.C., 2015; http://www.aaas.org/page/historical-trends-federal-rd
3. Chang, F. et al. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems 26*, 2 (June 2008), 4:1–4:26.
4. Datta, K. et al. Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing* (Austin, TX, Nov. 15–21). IEEE Press, Piscataway, NJ, 2008, 1–12.
5. Dennard, R.H., Gaensslen, F.H., Yu, H.-n., Rideout, V.L., Bassous, E., and LeBlanc, A.R. Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE Journal of Solid State Circuits 9*, 5 (Jan. 1974), 256–268.
6. Dongarra, J.J. The LINPACK benchmark: An explanation. In *Proceedings of the First International Conference on Supercomputing* (Athens, Greece, June 8–12). Springer-Verlag, New York, 1988, 456–474.
7. Dongarra, J.J. et al. The international exascale software project roadmap. *International Journal of High Performance Computing Applications 25*, 1 (Feb. 2011), 3–60.
8. Esmaeilzadeh, H., Blem, E., Amant, R.S., Sankaralingam, K., and Burger, D. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture* (San Jose, CA, June 4–8). ACM, New York, 2011, 365–376.
9. Fuller, S.H. and Millett, L.I. Computing performance: Game over or next level? *Computer 44*, 1 (Jan. 2011), 31–38.
10. Geist, A. and Lucas, R. Major computer science challenges at exascale. *International Journal of High Performance Applications 23*, 4 (Nov. 2009), 427–436.
11. Gill, P., Jain, N., and Nagappan, N. Understanding network failures in data centers: Measurement, analysis, and implications. *Proceedings of ACM SIGCOMM 41*, 4 (Aug, 2011), 350–361.
12. Hey, T., Tansley, S., and Tolle, K. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, Redmond, WA, 2009; http://research.microsoft.com/en-us/UM/redmond/about/collaboration/fourthparadigm/4th_PARADIGM_BOOK_complete_HR.pdf
13. Kamil, S., Shalf, J., and Strohmaier, E. Power efficiency in high-performance computing. In *Proceedings of the Fourth Workshop on High-Performance, Power-Aware Computing* (Miami, FL, Apr.). IEEE Press, 2008.
14. Kogge, P., Bergman, K., Borkar, S. et al. *Exascale Computing Study: Technology Challenges in Achieving Exascale Systems.* U.S. Defense Advanced Research Projects Agency, Arlington, VA, 2008; http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf
15. Lucas, R., Ang, J., Bergman, K., Borkar, S. et al. *Top Ten Exascale Research Challenges.* Office of Science, U.S. Department of Energy, Washington, D.C., Feb. 2014; http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf
16. Meuer, H., Strohmaier, E., Dongarra, J. and Simon, H. *Top 500 Supercomputer Sites, 2015*; http://www.top500.org
17. Nobelprize.org. Nobel Prize in Chemistry 2013; http://www.nobelprize.org/nobel_prizes/chemistry/laureates/2013/press.html
18. Olston, C., Reed, B., Strivastava, U., Kumar, R., and Tomkins, A. Pig Latin: A not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (Vancouver, BC, Canada, June 9–12). ACM Press, New York, 2008, 1099–1110.
19. Partnership for Advanced Computing in Europe (PRACE). 2014; http://www.prace-ri.eu/
20. Pinheiro, E., Weber, W.-D., and Barroso, L.A. Failure trends in a large disk drive population. In *Proceedings of the Fifth USENIX Conference on File and Storage Technologies* (San Jose, CA, Feb. 13–16). USENIX Association, Berkeley, CA, 2007.
21. Schroeder, B. and Gibson, G.A. Understanding disk failure rates: What does an MTTF of 1,000,000 hours mean to you? *ACM Transactions on Storage 3*, 3 (Oct. 2007), 8.
22. Schroeder, B., Pinheiro, E., and Weber, W.-D. DRAM errors in the wild: A large-scale field study. In *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems* (Seattle, WA, June). ACM Press, New York, 2009, 193–204.
23. U.S. Department of Energy. *Synergistic Challenges in Data-Intensive Science and Exascale Computing.* Report of the Advanced Scientific Computing Advisory Committee Subcommittee, Mar. 30, 2013; http://science.energy.gov/~/media/ascr/ascac/pdf/reports/2013/ASCAC_Data_Intensive_Computing_report_final.pdf
24. U.S. Department of Energy. *The Opportunities and Challenges of Exascale Computing.* Office of Science, Washington, D.C., 2010; http://science.energy.gov/~/media/ascr/ascac/pdf/reports/Exascale_subcommittee_report.pdf
25. White, T. *Hadoop: The Definitive Guide.* O'Reilly Media, May 2012.

Daniel A. Reed (dan-reed@uiowa.edu) is Vice President for Research and Economic Development, and Professor of Computer Science, Electrical, and Computer Engineering and Medicine, at the University of Iowa.

Jack Dongarra (dongarra@icl.utk.edu) holds an appointment at the University of Tennessee, Oak Ridge National Laboratory, and the University of Manchester.

Watch the authors discuss their work in this exclusive *Communications* video. http://cacm.acm.org/videos/exascale-computing-and-big-data

DOI:10.1145/2699418

A deep, fine-grain analysis of rhetorical structure highlights crucial sentiment-carrying text segments.

BY ALEXANDER HOGENBOOM, FLAVIUS FRASINCAR, FRANCISKA DE JONG, AND UZAY KAYMAK

Using Rhetorical Structure in Sentiment Analysis

POPULAR WEB SITES like Twitter, Blogger, and Epinions let users vent their opinions on just about anything through an ever-increasing amount of short messages, blog posts, and reviews. Automated sentiment-analysis techniques can extract traces of people's sentiment, or attitude toward certain topics, from such texts.[6]

This analysis can yield a competitive advantage for businesses,[2] as one-fifth of all tweets[10] and one-third of all blog posts[16] discuss products or brands. Identifying people's opinions on these topics,[11] and identifying the pros and cons of specific products,[12] are therefore promising applications for sentiment-analysis tools.

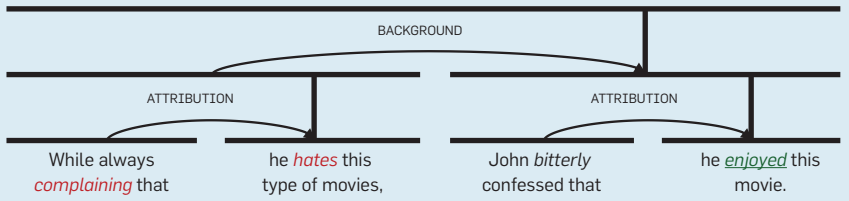
» key insights

- **Traditional automated sentiment analysis techniques rely mostly on word frequencies and thus fail to capture crucial nuances in text.**
- **A better understanding of a text's sentiment can be obtained by guiding the analysis by the text's rhetorical structure.**
- **A deep, fine-grain analysis of the rhetorical structure of smaller units of text can highlight crucial sentiment-carrying text segments and help correctly interpret these segments.**

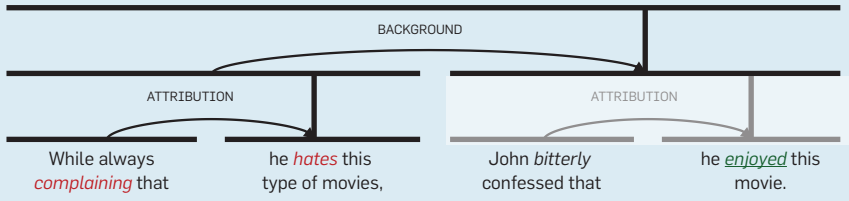
Many commercial sentiment-analysis systems rely mostly on the occurrence of sentiment-carrying words, although more sophistication can be introduced in several ways. Among open research issues identified by Feldman[6] is the role of textual structure. Structural aspects may contain valuable information;[19,24] sentiment-carrying words in a conclusion may

Figure 1. Interpretations of a positive RST-structured sentence consisting of nuclei (marked with vertical lines) and satellites.

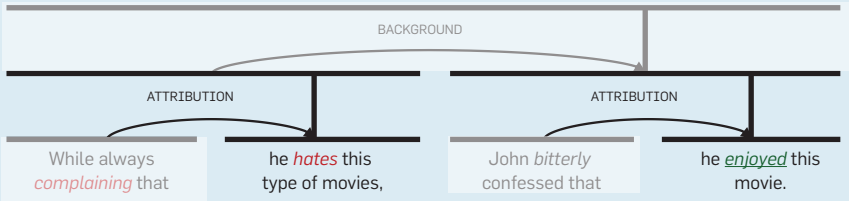
Negative words are in red in italics, and positive words are in green and underlined in italics. Sentiment-carrying words with relatively high intensity are brighter. Horizontal lines indicate the spans of the RST elements at each level of the hierarchical rhetorical structure. Arrows and their (capitalized) captions represent the relations of satellite elements to nucleus elements. Text segments and RST elements assigned a relatively low weight in the analysis of the conveyed sentiment are more transparent than those with higher weights.



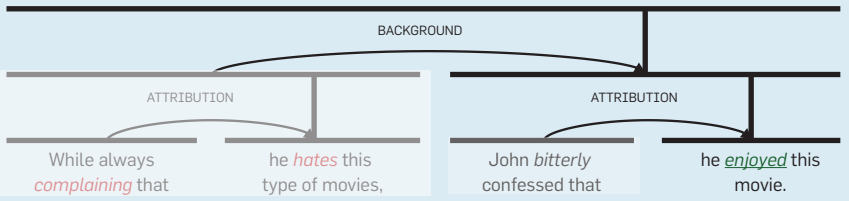
(a) Accounting for sentiment-carrying words only.



(b) Accounting for rhetorical roles as identified by the RST tree's top-level split.



(c) Accounting for the rhetorical roles of the leaf nodes of the RST tree.



(d) Accounting for the full, hierarchical rhetorical structure.

contribute more to a text's overall sentiment than sentiment-carrying words in, say, background information.

Existing work typically uses structural aspects of text to distinguish important text segments from less-important segments in terms of their contribution to a text's overall sentiment, subsequently weighting a segment's conveyed sentiment in accordance with its importance. A segment's importance is often related to its POSITION in a text,[19] yet recently developed methods make coarse-grain distinctions between segments based on their rhetorical roles[5,8,24] by applying Rhetorical Structure Theory (RST).[13]

While the benefit of exploiting isolated rhetorical roles for sentiment analysis has been demonstrated in existing work, the full rhetorical structure in which these roles are defined has thus far been ignored. However, a text segment with a particular rhetorical role can consist of smaller subordinate segments that are rhetorically related to one another, thus forming a hierarchical rhetorical tree structure. Important segments may contain less-important parts. Since accounting for such structural aspects of a text enables better understanding of the text,[15] we hypothesize that guiding sentiment analysis through a deep analysis of a text's rhetorical structure can yield better understanding of conveyed sentiment with respect to an entity of interest.

Our contribution is threefold. First, as an alternative to existing shallow RST-guided sentiment analysis approaches that typically focus on rhetorical relations in top-level splits of RST trees, we propose to focus on the leaf nodes of RST trees or, alternatively, to account for the full RST trees. Second, we propose a novel RST-based weighting scheme that is more refined than existing weighting schemes.[8,24] Third, rhetorical relations across sentences and paragraphs can provide useful context for the rhetorical structure of their subordinate text segments. Whereas existing work mostly guides sentiment analysis through sentence-level analyses of rhetorical structure (if at all), we therefore additionally incorporate paragraph-level and document-level analyses of rhetorical structure into the process. We thus account for rhetorical relations across sentences and paragraphs.

Sentiment Analysis and Rhetorical Relations

Automated sentiment analysis is related to natural language processing, computational linguistics, and text mining. Typical tasks include distinguishing subjective text segments from objective ones and determining the polarity of words, sentences, text segments, and documents.[18] We address the binary document-level polarity classification task, dealing with classifying documents as positive or negative.

The state of the art in sentiment analysis has been reviewed extensively.[6,18] Existing methods range from machine learning methods that exploit patterns in vector representations of text to lexicon-based methods that account for semantic orientation in individual words by matching the words with a sentiment lexicon, listing words and their associated sentiment. Lexicon-based methods are typically more robust across domains and texts,[24] allowing for intuitive incorporation of deep linguistic analyses.

Deep linguistic analysis is a key success factor for sentiment-analysis systems, as it helps deal with "compositionality,"[21] or the way the semantic

orientation of text is determined by the combined semantic orientations of its constituent phrases. This compositionality can be captured by accounting for the grammatical[20] or the discursive[4,5,8,22,24] structure of text.

RST[13] is a popular discourse-analysis framework, splitting text into rhetorically related segments that may in turn be split, thus yielding a hierarchical rhetorical structure. Each segment is either a nucleus or a satellite. Nuclei constitute a text's core and are supported by satellites that are considered less important. In total, 23 types of relations between RST elements have been proposed;[13] for example, a satellite may form an elaboration on or a contrast with a nucleus.

Consider the positive sentence "While always complaining that he hates this type of movies, John bitterly confessed that he enjoyed this movie," which contains mostly negative words. RST can split the sentence into a hierarchical rhetorical structure of text segments (see Figure 1). The top-level nucleus contains the core message "John bitterly confessed he enjoyed this movie," with a satellite providing background information. This background satellite consists of a nucleus ("he hates this type of movies") and an attributing satellite ("While always complaining that"). Similarly, the core consists of a nucleus ("he enjoyed this movie") and an attributing satellite ("John bitterly confessed that"). The sentence conveys a positive overall sentiment toward the movie, due to the way the sentiment-carrying words are used in the sentence; the actual sentiment is conveyed by the nucleus "he enjoyed this movie."

Several methods of using rhetorical relations for sentiment analysis are available, with some, including our own methods, relying more strongly on RST5,[8,24] than others.[4,22] In order to identify rhetorical relations in text, the most successful methods use the Sentence-level PArsing of DiscoursE, or SPADE, tool,[23] which creates an RST tree for each sentence. Another parser is the HIgh-Level Discourse Analyzer, or HILDA,[9] which parses discourse structure at the document level by means of a greedy bottom-up tree-building method that uses machine-learning classifiers to iteratively assign RST relation labels to those (compound) segments of a document most likely to be rhetorically related. SPADE and HILDA take as input free text (a priori divided into paragraphs and sentences) and produce LISP-like representations of the text and its rhetorical structure.

This analysis can yield a competitive advantage for businesses, as one-fifth of all tweets and one-third of all blog posts discuss products or brands.

Document-level polarity classification has been guided successfully by analyses of the most relevant text segments, as identified by differentiating between top-level nuclei and satellites in sentence-level RST trees.[24] Another, more elaborate, method of utilizing RST in sentiment analysis accounts for the different types of relations between nuclei and satellites,[8] yielding improvements over methods that only distinguish nuclei from satellites.[5,8]

Polarity Classification Guided by Rhetorical Structure

Existing methods do not use RST to its full extent yet typically focus on top-level splits of sentence-level RST trees and thus employ a rather shallow, coarse-grain analysis. However, in RST, rhetorical relations are defined within a hierarchical structure that could be accounted for. Important nuclei may contain less-important satellites that should be treated accordingly. We therefore propose to guide polarity classification through a deep analysis of a text's hierarchical rhetorical structure rather than its isolated rhetorical relations. We account for rhetorical relations within and across sentences by allowing for not only sentence-level but also paragraph-level and document-level analyses of RST trees.

Fine-grain analysis. Figure 1 outlines the potential of RST-guided polarity classification. Based on the sentiment-carrying words alone, our example sentence is best classified as negative, as in Figure 1a. Accounting for the rhetorical roles of text segments as identified by the RST tree's top-level split enables a more elaborate but still coarse-grain analysis of the overall sentiment, as in Figure 1b. The top-level nucleus contains as many positive as negative words and may thus be classified as either positive or negative. The negative words in the top-level satellite trigger a negative classification of this segment, which is a potentially irrelevant seg-

ment that should be assigned a lower weight in the analysis.

However, such a coarse-grain analysis does not capture the nuances of lower-level splits of an RST tree; for instance, the top-level nucleus in our example consists of two segments, one of which is the sentence's actual core ("he enjoyed this movie"), whereas the other is less relevant and should therefore be assigned a lower weight in the analysis. Accounting for the rhetorical roles of the leaf nodes of an RST tree rather than the top-level splits can thus enable a more accurate sentiment analysis, as in Figure 1c.

However, an exclusive focus on leaf nodes of RST trees does not account for the text segments' rhetorical roles being defined within the context of the rhetorical roles of the segments that embed them; for instance, the second leaf node in our example RST tree is a nucleus of a possibly irrelevant background satellite, whereas the fourth leaf node is the nucleus of the nucleus and constitutes the actual core. The full rhetorical structure must be considered in the analysis in order to account for this circumstance, as in Figure 1d.

We propose a lexicon-based sentiment-analysis framework that can perform an analysis of the rhetorical structure of a piece of text at various levels of granularity and that can subsequently use this information for classifying the text's overall polarity. Our framework (see Figure 2) takes several steps in order to classify the polarity of a document.

Word-level sentiment scoring. Our method first splits a document into paragraphs, sentences, and words. Then, for each sentence, it automatically determines the part-of-speech (POS) and lemma of each word. It then disambiguates the word sense of each word using an unsupervised algorithm that iteratively selects the word sense with the highest semantic similarity to the word's context.[8] It then retrieves the sentiment of each word, associated with its particular combination of POS, lemma, and word sense from a sentiment lexicon like SentiWordNet.[1]

Rhetorical structure processing. In order to guide the polarity classification of a document d by its rhetorical structure, sentiment scores are computed for each segment s_i. Our framework supports several methods of computing such scores: a baseline method plus several RST-based methods that can be applied to sentence-level, paragraph-level, and document-level RST trees.

Baseline. As a baseline, we consider text segments to be the sentences s_d of document d, with their associated baseline sentiment score $\zeta^B_{s_i}$ being the weighted sum of the score ζ_{t_j} of each word t_j and its weight w_{t_j}, or

$$\zeta^B_{s_i} = \sum_{t_j \in s_i} (\zeta_{t_j} \cdot w_{t_j}), \forall s_i \in S_d. \quad (1)$$

Top-level rhetorical structure. Our framework additionally supports the top-level RST-based method applied in existing work, an approach we refer to as T. The sentiment score $\zeta^B_{s_i}$ of a top-level RST segment s_i is defined as the sum of the sentiment ζ_{t_j} associated with each word t_j in segment s_i, weighted with a weight w_{s_i} associated with the segment's rhetorical role, or

$$\zeta^T_{s_i} = \sum_{t_j \in s_i} (\zeta_{t_j} \cdot w_{s_i}), \forall s_i \in T_d, \quad (2)$$

with T_d representing all top-level RST nodes in the RST trees for document d.

Leaf-level rhetorical structure. Another method is our leaf-level RST-based analysis L. The sentiment score $\zeta^L_{s_i}$ of an RST segment s_i from the leaf nodes L_d of an RST tree for document d is computed as the summed sentiment score of its words, weighted for the segment's rhetorical role, or

$$\zeta^L_{s_i} = \sum_{t_j \in s_i} (\zeta_{t_j} \cdot w_{s_i}), \forall s_i \in L_d. \quad (3)$$

Hierarchical rhetorical structure. The last supported approach is our method of accounting for the full path from an RST tree root to a leaf node, such that the sentiment conveyed by the latter can be weighted while accounting for its rhetorical context. In our hierarchy-based sentiment-scoring method H, we model the sentiment score $\zeta^H_{s_i}$ of a leaf-level RST segment s_i as a function of the sentiment scores of its words and the weights w_{r_n} associated with the rhetorical role of each node r_n from the nodes P_{s_i} on the path from the root to the leaf, or

$$\zeta^H_{s_i} = \sum_{t_j \in s_i} \zeta_{t_j} \cdot \left(\frac{\sum_{r_n \in P_{s_i}} \left(|w_{r_n}| \cdot \delta^{-(\lambda_{r_n}-1)} \right)}{\sum_{r_n \in P_{s_i}} \delta^{-(\lambda_{r_n}-1)}} \right) \cdot \prod_{r_n \in P_{s_i}} \mathrm{sgn}(w_{r_n}), \forall s_i \in L_d, \delta > 1, \quad (4)$$

where δ represents a diminishing factor and λ_{r_n} signals the level of node r_n in the RST tree, with the level of the root node being 1. For $\delta > 1$, each subsequent level contributes less than its parent to the segment's RST-based weight, thus preserving the hierarchy of the relations in the path.

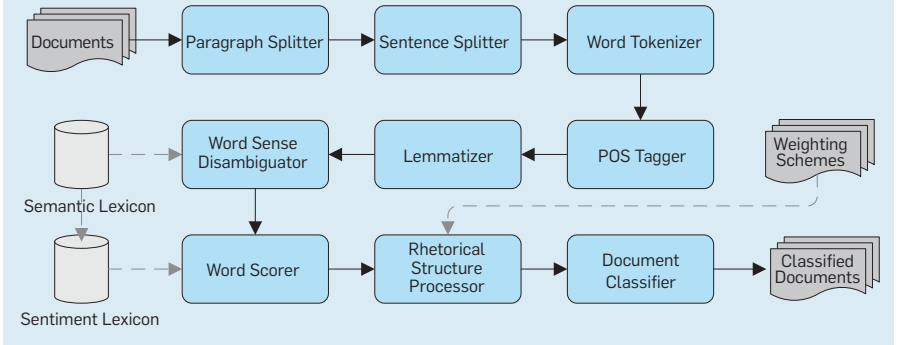
Classifying document polarity. The segment-level sentiment scores can be aggregated in order to determine the overall polarity of a document. The baseline, top-level, leaf-level, and hierarchy-based sentiment scores ζ^B_d, ζ^T_d, ζ^L_d, and ζ^H_d for document d are defined as

$$\zeta^B_d = \sum_{s_i \in S_d} \zeta^B_{s_i}, \quad (5)$$

$$\zeta^T_d = \sum_{s_i \in T_d} \zeta^T_{s_i}, \quad (6)$$

$$\zeta^L_d = \sum_{s_i \in L_d} \zeta^L_{s_i}, \quad (7)$$

Figure 2. A schematic overview of our sentiment-analysis framework; solid arrows indicate information flow, and dashed arrows indicate a used-by relationship.



$$\zeta_d^H = \sum_{s_i \in L_d} \zeta_{s_i}^H. \quad (8)$$

The resulting document-level sentiment scores can be used to classify document d's polarity c_d as negative (−1) or positive (1), following

$$c_d = \begin{cases} -1 \text{ if } (\zeta_d - \epsilon) < 0, \\ 1 \quad \text{else.} \end{cases} \quad (9)$$

Here, ϵ represents an offset that corrects a possible bias in the sentiment scores caused by people's tendency to write negative reviews with rather positive words,[24] or

$$\epsilon = 0.5\left(\frac{\sum_{d\in\Phi}\zeta_d}{|\Phi|} + \frac{\sum_{d\in N}\zeta_d}{|N|}\right), \quad (10)$$

with Φ and N denoting the respective subsets of positive and negative documents in a training set.

Weighting schemes. We consider six different weighting schemes. Two serve as baselines and are applicable to the baseline sentiment scoring approach as defined in Equations (1) and (5). The others apply to our three RST-based approaches as defined in (2) and (6), in (3) and (7), and in (4) and (8), respectively.

Our BASELINE scheme serves as an absolute baseline and assigns each word a weight of 1; structural aspects are not accounted for. A second baseline is a position-based scheme. In this POSITION scheme, word weights are uniformly distributed, ranging from 0 for the first word to 1 for the last word of a text, as an author's views are more likely to be summarized near the end of a text.[19]

The first RST-based scheme (I) assigns a weight of 1 to nuclei and a weight of 0 to satellites;[24] the second RST-based scheme (II) matches the second set of weights for nuclei and satellites used by Taboada et al.,[24] or 1.5 and 0.5, respectively. In both schemes I and II, we set the diminishing factor for the H method to 2, such that each level in a tree is at least as important as all its subsequent levels combined, thus enforcing a strict hierarchy.

Another RST-specific scheme is the extended scheme X in which we differentiate satellite weights by their RST relation type.[8] Additionally, we propose a novel extension of X: the full weighting scheme F, in which we not only differentiate satellite weights but also nucleus weights by their RST relation type.

Table 1. Characteristics of our considered polarity-classification approaches, or our baselines, our SPADE-based sentence-level RST-guided methods, and our HILDA-based sentence-level, paragraph-level, and document-level RST-guided methods.

Method	Unit	RST	Weights
BASELINE	DOCUMENT	NONE	BASELINE
POSITION	DOCUMENT	NONE	POSITION
SPADE.S.T.I	Sentence	Top-level	I
SPADE.S.T.II	Sentence	Top-level	II
SPADE.S.T.X	Sentence	Top-level	X
SPADE.S.T.F	Sentence	Top-level	F
SPADE.S.L.I	Sentence	Leaf-level	I
SPADE.S.L.II	Sentence	Leaf-level	II
SPADE.S.L.X	Sentence	Leaf-level	X
SPADE.S.L.F	Sentence	Leaf-level	F
SPADE.S.H.I	Sentence	Hierarchical	I
SPADE.S.H.II	Sentence	Hierarchical	II
SPADE.S.H.X	Sentence	Hierarchical	X
SPADE.S.H.F	Sentence	Hierarchical	F
HILDA.S.T.I	Sentence	Top-level	I
HILDA.S.T.II	Sentence	Top-level	II
HILDA.S.T.X	Sentence	Top-level	X
HILDA.S.T.F	Sentence	Top-level	F
HILDA.S.L.I	Sentence	Leaf-level	I
HILDA.S.L.II	Sentence	Leaf-level	II
HILDA.S.L.X	Sentence	Leaf-level	X
HILDA.S.L.F	Sentence	Leaf-level	F
HILDA.S.H.I	Sentence	Hierarchical	I
HILDA.S.H.II	Sentence	Hierarchical	II
HILDA.S.H.X	Sentence	Hierarchical	X
HILDA.S.H.F	Sentence	Hierarchical	F
HILDA.P.T.I	Paragraph	Top-level	I
HILDA.P.T.II	Paragraph	Top-level	II
HILDA.P.T.X	Paragraph	Top-level	X
HILDA.P.T.F	Paragraph	Top-level	F
HILDA.P.L.I	Paragraph	Leaf-level	I
HILDA.P.L.II	Paragraph	Leaf-level	II
HILDA.P.L.X	Paragraph	Leaf-level	X
HILDA.P.L.F	Paragraph	Leaf-level	F
HILDA.P.H.I	Paragraph	Hierarchical	I
HILDA.P.H.II	Paragraph	Hierarchical	II
HILDA.P.H.X	Paragraph	Hierarchical	X
HILDA.P.H.F	Paragraph	Hierarchical	F
HILDA.D.T.I	Document	Top-level	I
HILDA.D.T.II	Document	Top-level	II
HILDA.D.T.X	Document	Top-level	X
HILDA.D.T.F	Document	Top-level	F
HILDA.D.L.I	Document	Leaf-level	I
HILDA.D.L.II	Document	Leaf-level	II
HILDA.D.L.X	Document	Leaf-level	X
HILDA.D.L.F	Document	Leaf-level	F
HILDA.D.H.I	Document	Hierarchical	I
HILDA.D.H.II	Document	Hierarchical	II
HILDA.D.H.X	Document	Hierarchical	X
HILDA.D.H.F	Document	Hierarchical	F

Table 2. Performance measures of our considered baselines, our SPADE-based sentence-level RST-guided methods, and our HILDA-based sentence-level, paragraph-level, and document-level RST guided methods, based on tenfold cross-validation on the movie-review dataset. The best performance in each group of methods is in bold for each performance measure.

	Positive			Negative			Overall	
Method	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**	**Accuracy**	**F_1**
Baseline	0.632	0.689	0.659	0.658	0.599	0.627	0.644	0.643
Position	0.637	0.713	0.673	0.674	0.593	0.631	0.653	0.652
SPADE.S.T.I	0.638	0.675	0.656	0.655	0.617	0.635	0.646	0.646
SPADE.S.T.II	0.640	0.688	0.663	0.663	0.613	0.637	0.651	0.650
SPADE.S.T.X	**0.693**	**0.725**	**0.709**	**0.712**	**0.679**	**0.695**	**0.702**	**0.702**
SPADE.S.T.F	0.703	0.726	0.715	0.717	0.694	0.705	0.710	0.710
SPADE.S.L.I	0.636	0.702	0.667	0.667	0.598	0.631	0.650	0.649
SPADE.S.L.II	0.640	0.700	0.669	0.669	0.607	0.637	0.654	0.653
SPADE.S.L.X	0.699	0.715	0.707	0.708	0.692	0.700	0.704	0.703
SPADE.S.L.F	0.705	0.731	0.718	0.721	0.694	0.707	0.713	0.712
SPADE.S.H.I	0.647	0.678	0.662	0.662	0.630	0.645	0.654	0.654
SPADE.S.H.II	0.642	0.696	0.668	0.668	0.612	0.639	0.654	0.653
SPADE.S.H.X	0.707	0.723	0.715	0.716	**0.700**	0.708	0.712	0.711
SPADE.S.H.F	**0.710**	**0.738**	**0.724**	**0.727**	0.699	**0.713**	**0.719**	**0.718**
HILDA.S.T.I	0.633	0.676	0.654	0.652	0.608	0.629	0.642	0.642
HILDA.S.T.II	0.636	0.686	0.660	0.659	0.607	0.632	0.647	0.646
HILDA.S.T.X	0.692	0.709	0.701	0.702	0.685	0.693	0.697	0.697
HILDA.S.T.F	0.697	**0.745**	**0.720**	**0.726**	0.676	0.700	0.711	0.710
HILDA.S.L.I	0.629	0.685	0.656	0.654	0.596	0.624	0.641	0.640
HILDA.S.L.II	0.636	0.685	0.660	0.659	0.608	0.632	0.647	0.646
HILDA.S.L.X	0.698	0.711	0.705	0.706	0.693	0.699	0.702	0.702
HILDA.S.L.F	**0.705**	0.732	0.718	0.721	0.693	**0.707**	**0.713**	**0.712**
HILDA.S.H.I	0.634	0.675	0.654	0.653	0.611	0.631	0.643	0.643
HILDA.S.H.II	0.638	0.688	0.662	0.661	0.609	0.634	0.649	0.648
HILDA.S.H.X	0.699	0.693	0.696	0.695	**0.701**	0.698	0.697	0.697
HILDA.S.H.F	0.699	0.740	0.719	0.724	0.682	0.702	0.711	0.711
HILDA.P.T.I	0.618	0.638	0.628	0.626	0.605	0.615	0.622	0.621
HILDA.P.T.II	0.628	0.674	0.650	0.648	0.600	0.623	0.637	0.637
HILDA.P.T.X	0.681	0.697	0.689	0.690	0.674	0.682	0.686	0.685
HILDA.P.T.F	0.703	0.702	0.702	0.702	0.703	0.703	0.703	0.702
HILDA.P.L.I	0.632	0.684	0.657	0.656	0.602	0.628	0.643	0.642
HILDA.P.L.II	0.633	0.685	0.658	0.657	0.603	0.629	0.644	0.643
HILDA.P.L.X	0.690	0.705	0.697	0.698	0.683	0.691	0.694	0.694
HILDA.P.L.F	0.701	**0.720**	**0.710**	**0.712**	0.693	0.702	0.707	0.706
HILDA.P.H.I	0.583	0.609	0.596	0.591	0.565	0.578	0.587	0.587
HILDA.P.H.II	0.629	0.682	0.655	0.653	0.598	0.624	0.640	0.639
HILDA.P.H.X	0.706	0.683	0.694	0.693	0.716	0.704	0.700	0.699
HILDA.P.H.F	**0.713**	0.692	0.703	0.701	**0.722**	**0.711**	**0.707**	**0.707**
HILDA.D.T.I	0.627	0.616	0.621	0.622	0.633	0.628	0.625	0.624
HILDA.D.T.II	0.627	0.650	0.639	0.637	0.614	0.625	0.632	0.632
HILDA.D.T.X	0.682	0.689	0.685	0.686	0.678	0.682	0.684	0.683
HILDA.D.T.F	0.684	0.696	0.690	0.691	0.679	0.685	0.688	0.687
HILDA.D.L.I	0.627	0.679	0.652	0.650	0.596	0.622	0.638	0.637
HILDA.D.L.II	0.631	0.687	0.658	0.656	0.598	0.626	0.643	0.642
HILDA.D.L.X	0.689	0.719	0.704	0.706	0.675	0.690	0.697	0.697
HILDA.D.L.F	0.701	**0.727**	**0.714**	**0.717**	0.690	0.703	**0.709**	**0.708**
HILDA.D.H.I	0.580	0.516	0.546	0.564	0.627	0.594	0.572	0.570
HILDA.D.H.II	0.630	0.663	0.646	0.645	0.611	0.627	0.637	0.637
HILDA.D.H.X	0.706	0.696	0.701	0.700	**0.710**	0.705	0.703	0.703
HILDA.D.H.F	**0.707**	0.708	0.707	0.707	0.706	**0.707**	0.707	0.707

For both X and F, the weights and the diminishing factor δ can be optimized.

Evaluation

By means of a set of experiments, we evaluate the variants of our polarity-classification approach guided by structural aspects of text. We focus on a widely used collection of 1,000 positive and 1,000 negative English movie reviews.[17]

Experimental setup. In the Java implementation of our framework, we detect paragraphs using the <P> and </P> tags in the HTML files. For sentence splitting, we rely on the preprocessed reviews.[17] The framework uses the Stanford Tokenizer[14] for word segmentation. For POS tagging and lemmatization, our framework uses the OpenNLP[3] POS tagger and the JWNL API,[25] respectively. We link word senses to WordNet,[7] thus enabling retrieval of their sentiment scores from SentiWordNet[1] by subtracting the associated negativity scores from the positivity scores. This retrieval method yields real numbers ranging from −1 (negative) to 1 (positive). In the final aggregation of word scores, our framework assigns a weight to each word by means of one of our methods (see Table 1), most of which are RST-based.

We evaluate our methods on the accuracy, precision, recall, and F_1-score on positive and negative documents, as well as on the overall accuracy and macro-level F_1-score. We assess the statistical significance of performance differences through paired, one-sided t-tests, comparing methods against one another in terms of their mean performance measures over all 10 folds, under the null hypothesis that the mean performance of a method is less than or equal to the mean performance of another method.

In order to evaluate our methods, we apply tenfold cross-validation. For each fold, we optimize the offsets, weights, and diminishing factor δ for weighting schemes X and F using particle-swarm optimization,[5] with particles searching a solution space, where their coordinates correspond with the weights and offsets (between −2 and 2) and δ (between 1 and 2). The fitness of a particle is its macro-level F_1-score on a training set.

Experimental results. Our experimental analysis consists of four steps. First, we compare the performance of our considered methods. We then analyze the optimized weights and diminishing factors. Subsequently, we demonstrate how documents are typically perceived by distinct methods. Last, we discuss several caveats for our findings.

Performance. Table 2 presents our methods' performance, whereas Figure 3 shows the *p*-values for the macro-level F_1-scores for all combinations of methods, ordered from top to bottom and from left to right in accordance with an initial ranking from worst- to best-performing methods. We sort the methods based on their weighting scheme (BASELINE, POSITION, I, II, X, and F), analysis level (HILDA.D, HILDA.P, HILDA.S, and SPADE.S), and RST analysis method (T, L, and H), respectively. Darker colors indicate lower *p*-values, with darker rows and columns signaling weak and competitive approaches, respectively. Given a correct ordering, darker colors should be right from the diagonal, toward the upper-right corner of Figure 3.

Three trends can be observed in Table 2 and Figure 3. First, weighting schemes X and F significantly outperform the I, II, BASELINE, and POSITION schemes, with F significantly outperforming X in most cases. Conversely, schemes I, II, BASELINE, and POSITION do not exhibit clearly significant performance differences with respect to one another.

A second trend is that methods guided by document-level RST trees are typically outperformed by comparable methods utilizing paragraph-level RST trees. Sentence-level RST trees yield the best performance. An explanation lies in misclassifications of rhetorical relations being potentially more harmful in larger RST trees; a misclassified relation in one of the top levels of a document-level RST tree can cause a misinterpretation of a large part of the document.

The third trend is that methods applying the hierarchy-based RST analysis method H typically slightly outperform comparable approaches that use the leaf-based analysis L instead, which in turn significantly outperform comparable approaches using the top-level RST analysis method T. The deeper analyses L and H clearly yield a significant advantage over the rather shallow analysis method T.

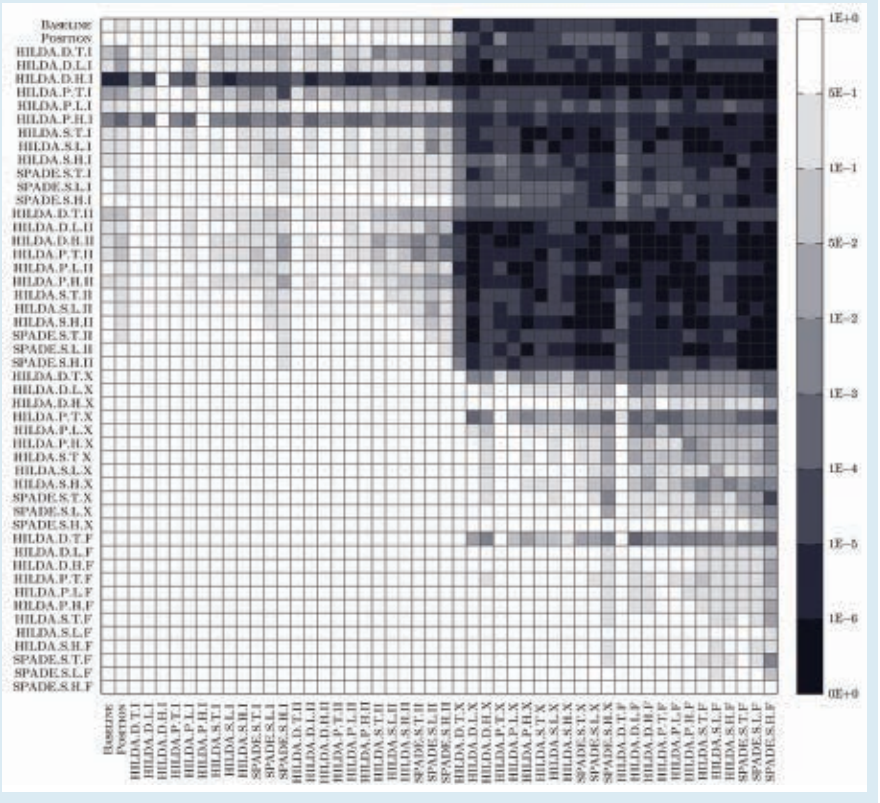
Some methods stand out in particular. First, HILDA.D.T and HILDA.P.T are relatively weak, especially when using weighting schemes I and II. The top-level RST analysis method T is typically too coarse-grain for larger RST trees, as, for instance, documents are segmented in only two parts when using document-level RST trees. Other weak methods are HILDA.D.H.I and HILDA.P.H.I. The combination of naïve weighting scheme I with deep, hierarchy-based analysis of the rhetorical structure of a document or its paragraphs results in a narrow focus on very specific segments.

Approaches that stand out positively are those applying hierarchy-based RST analysis H to sentence-level RST trees, with weighting schemes X and F, or HILDA.S.H.X, HILDA.S.H.F, SPADE.S.H.X, and SPADE.S.H.F. These approaches perform comparably well because they involve detailed analysis of rhetorical structure; the analysis is performed on the smallest considered units (sentences), the hierarchy of RST trees is accounted for, and the weights are differentiated per type of rhetorical relation. These results confirm our hypothesis that the sentiment conveyed by a text can be captured more adequately by incorporating a deep analysis of the text's rhetorical structure.

Optimized weights. The optimized weights for distinct types of nuclei and satellites, as defined in RST,[13] exhibit several patterns. First, nucleus elements are generally rather important in weighting scheme X, with most weights ranging between 0.5 and 1. The sentiment expressed in elaborating satellites is typically assigned a similar importance in weighting scheme X. Furthermore, contrasting satellites mostly receive weights around or below 0. Background satellites are typically assigned relatively low weights as well.

In weighting scheme F, nuclei and satellites in an attributing relation are typically both assigned weights around 0.5. Conversely, for background and contrasting relations, satellites are

Figure 3. The *p*-values for the paired, one-sided *t*-test assessing the null hypothesis of the mean macro-level F_1-scores of the methods in the columns being lower than or equal to the mean macro-level F_1-scores of the methods in the rows.



more clearly distinct from nuclei. Background satellites are typically assigned less importance than their associated nuclei, with respective weights of 0 and 1. For contrasting relations, nuclei are mostly assigned weights between 0.5 and 1, whereas optimized satellite weights are typically negative.

The optimized values for the diminishing factor δ are typically around 1.5 and 1.25 for weighting schemes X and F, respectively. These values result in the first 15 (for X) or 30 (for F) levels of RST trees being accounted for. Interestingly, with some (document-level) RST trees being more than 100 levels deep in our corpus, the optimized diminishing factors effectively disregard the lower, most-fine-grain parts of RST trees, thus realizing a balance between the level of detail and the potential noise in the analysis.

Processing a document. The observed differences in performance of our polarity-classification methods originate in how these methods perceive a document in the sentiment-analysis process. Figure 4 visualizes how the interpretations of a movie review differ across various methods.

Our software assigns many words in our example positive sentiment, whereas fewer words are identified as carrying negative sentiment. When assigning each part of the review an equal weight (see Figure 4a), this relative abundance of positive words suggests the review is rather positive, whereas it is in fact negative.

Our best-performing RST-based baseline, SPADE.S.T.X, considers only the top-level splits of sentence-level RST trees and yields a rather coarse-grain segmentation of the sentences, leaving little room for more subtle nuance (see Figure 4b). Nevertheless, SPADE.S.T.X successfully highlights the conclusion and the arguments supporting this conclusion. The polarity of some positive words is even inverted, as the reviewer uses them in a rather negative way.

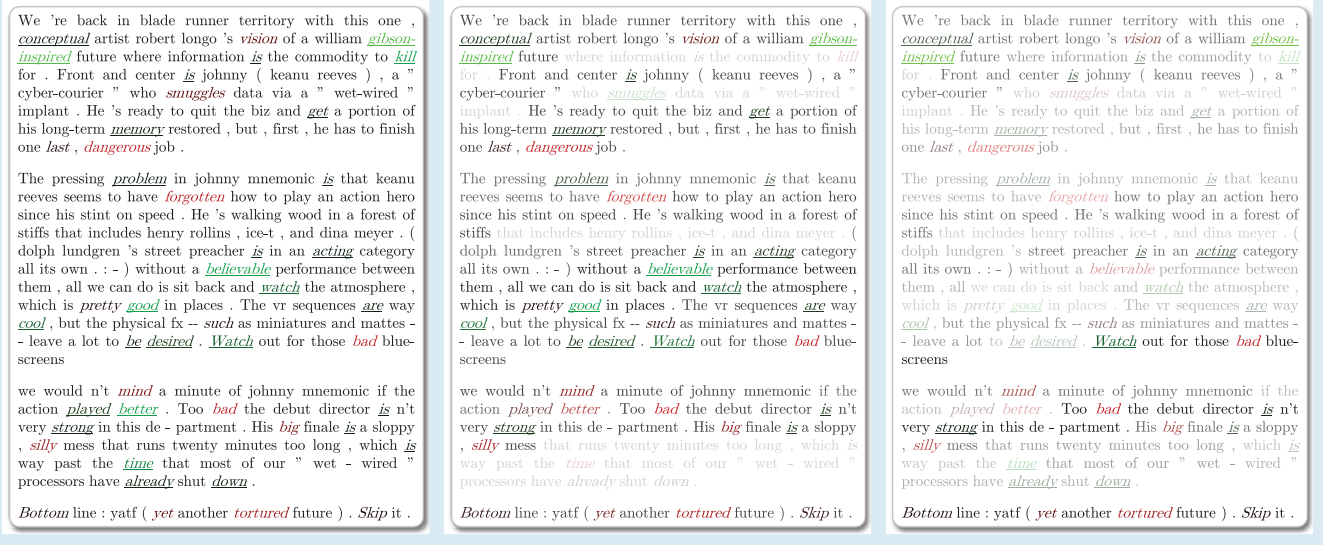
SPADE.S.H.F, our best-performing approach, yields a very detailed analysis in which subtle distinctions are made between small text segments (see Figure 4c). Such nuance helps bring out the parts of the review that are most relevant with respect to its overall sentiment. SPADE.S.H.F ignores most of the irrelevant background information in the first paragraph and highlights the reviewer's main concerns in the second and third paragraphs. Moreover, sentiment related to the movie's good aspects is often inverted and mostly ignored by SPADE.S.H.F. The overall recommendation is emphasized in the last paragraph. All in all, it is the incorporation of the detailed analysis of the review's rhetorical structure into the sentiment-analysis process that makes it possible for our polarity classifier to better understand the review.

Caveats. Our failure analysis has revealed that some misclassifications of authors' sentiment are due to misinterpreted sarcasm and an occasional misinterpretation of proverbs. Additionally, not all sentiment-carrying words are identified as such by SentiWordNet, as their word sense cannot be successfully disambiguated or the word cannot be found in our sentiment lexicon. Other challenges are related to the information content of documents; for instance, in our corpus, some reviewers tend to mix their opinionated statements with plot details containing irrelevant sentiment-carrying words. Additionally, some reviewers evaluate a movie by comparing it with other movies. Such statements require a distinction between entities and their associated sentiment, as well as real-world knowledge to be incorporated into the analysis.

Even though our RST-based polarity-classification methods cannot cope particularly well with the specific phe-

Figure 4. Movie review cv817_3675, as processed by various sentiment-analysis methods.

Negative words are in red in italics, and positive words are in green and underlined in italics. Sentiment-carrying words with high intensity are brighter, and words carrying less sentiment are darker. Text segments assigned a relatively low weight in the analysis of the conveyed sentiment are more transparent than text segments with higher weights.



nomena mentioned earlier, they significantly outperform our non-RST baselines. However, these improvements come at a cost of processing times increasing by almost a factor of 10. The bottleneck here is formed by the RST parsers, rather than by the application of our weighting schemes.

Conclusion

We have demonstrated that sentiment analysis can benefit from deep analysis of a text's rhetorical structure, enabling the distinction be made between important text segments and less-important ones in terms of their contribution to a text's overall sentiment. This is a significant step forward with respect to existing work, which is limited to guiding sentiment analysis by shallow analyses of rhetorical relations in (mostly sentence-level) rhetorical structure trees.

Our contribution is threefold. First, our novel polarity-classification methods guided by deep leaf-level or hierarchy-based RST analyses significantly outperform existing approaches, which are guided by shallow RST analyses or by no RST-based analyses at all. Second, the novel RST-based weighting scheme in which we differentiate the weights of nuclei and satellites by their RST relation type significantly outperforms existing schemes. And third, we have compared the performance of polarity-classification approaches guided by sentence-level, paragraph-level, and document-level RST trees, thus demonstrating that RST-based polarity classification works best when focusing on RST trees of smaller units of a text (such as sentences).

In future work, we aim to investigate the applicability of other, possibly more scalable methods for exploiting (discourse) structure of text in sentiment analysis. We also plan to validate our findings on other corpora covering other domains and other types of text.

Acknowledgments

Special thanks go to Bas Heerschop and Frank Goossen for their contributions in the early stages of this work. We are partially supported by the Dutch national research program COMMIT (http://www.commit-nl.nl/about-commit).

References

1. Baccianella, S., Esuli, A., and Sebastiani, F. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation* (Valletta, Malta, May 19–21). European Language Resources Association, Paris, France, 2010, 2200–2204.
2. Bal, D., Bal, M., van Bunningen, A., Hogenboom, A., Hogenboom, F., and Frasincar, F. Sentiment analysis with a multilingual pipeline. In *Proceedings of the 12th International Conference on Web Information System Engineering* (Sydney, Australia, Oct. 12–14) Volume 6997 of Lecture Notes in Computer Science. Springer, Berlin, Germany, 2011, 129–142.
3. Baldridge, J. and Morton, T. *OpenNLP*, 2004; http://opennlp.sourceforge.net/
4. Chardon, B., Benamara, F., Mathieu, Y., Popescu, V., and Asher, N. Measuring the effect of discourse structure on sentiment analysis. In *Proceedings of the 14th International Conference on Intelligent Text Processing and Computational Linguistics* (University of the Aegean, Samos, Greece, Mar. 24–30) Volume 7817 of Lecture Notes in Computer Science. Springer, Berlin, Germany, 2013, 25–37.
5. Chenlo, J., Hogenboom, A., and Losada, D. Rhetorical structure theory for polarity estimation: An experimental study. *Data and Knowledge Engineering 94*, B (Nov. 2014), 135–147.
6. Feldman, R. Techniques and applications for sentiment analysis. *Commun. ACM 56*, 4 (Apr. 2013), 82–89.
7. Fellbaum, C. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.
8. Heerschop, B., Goossen, F., Hogenboom, A., Frasincar, F., Kaymak, U., and de Jong, F. Polarity analysis of texts using discourse structure. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management* (Glasgow, U.K., Oct. 24–28). ACM Press, New York, 2011, 1061–1070.
9. Hernault, H., Prendinger, H., duVerle, D., and Ishizuka, M. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse 1*, 3 (Dec. 2010), 1–33.
10. Jansen, B., Zhang, M., Sobel, K., and Chowdury, A. Twitter power: Tweets as electronic word of mouth. *Journal of the American Society for Information Science and Technology 60*, 11 (Nov. 2009), 2169–2188.
11. Kim, S. and Hovy, E. Determining the sentiment of opinions. In *Proceedings of the 20th International Conference on Computational Linguistics* (University of Geneva, Switzerland, Aug. 23–27). Association for Computational Linguistics, Stroudsburg, PA, 2004, 1367–1373.
12. Kim, S. and Hovy, E. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics* (Sydney, Australia, July 17–21). Association for Computational Linguistics, Stroudsburg, PA, 2006, 483–490.
13. Mann, W. and Thompson, S. Rhetorical structure theory: Toward a functional theory of text organization. *Text 8*, 3 (Jan. 1988), 243–281.
14. Manning, C., Grow, T., Grenager, T., Finkel, J., and Bauer, J. *Stanford Tokenizer*, 2010; http://nlp.stanford.edu/software/tokenizer.shtml
15. Marcu, D. The rhetorical parsing of unrestricted texts: A surface-based approach. *Computational Linguistics 26*, 3 (Sept. 2000), 395–448.
16. Melville, P., Sindhwani, V., and Lawrence, R. Social media analytics: Channeling the power of the blogosphere for marketing insight. In *Proceedings of the First Workshop on Information in Networks* (New York, Sept. 25–26, 2009); http://www.prem-melville.com/publications/
17. Pang, B. and Lee, L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics* (Barcelona, Spain, July 21–26). Association for Computational Linguistics, Stroudsburg, PA, 2004, 271–280.
18. Pang, B. and Lee, L. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval 2*, 1 (Jan. 2008), 1–135.
19. Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing* (Honolulu, HI, Oct. 25–27). Association for Computational Linguistics, Stroudsburg, PA, 2002, 79–86.
20. Sayeed, A., Boyd-Graber, J., Rusk, B., and Weinberg, A. Grammatical structures for word-level sentiment detection. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Montreal, Canada, July 3–8). Association for Computational Linguistics, Stroudsburg, PA, 2012, 667–676.
21. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (Seattle, WA, Oct. 18–21). Association for Computational Linguistics, Stroudsburg, PA, 2013, 1631–1642.
22. Somasundaran, S., Namata, G., Wiebe, J., and Getoor, L. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing* (Singapore, Aug. 6–9). Association for Computational Linguistics, Stroudsburg, PA, 2009, 170–179.
23. Soricut, R. and Marcu, D. Sentence-level discourse parsing using syntactic and lexical information. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference* (Edmonton, Canada, May 27–June 1). Association for Computational Linguistics, Stroudsburg, PA, 2003, 149–156.
24. Taboada, M., Voll, K., and Brooke, J. *Extracting Sentiment as a Function of Discourse Structure and Topicality Technical Report 20*. Simon Fraser University, Burnaby, B.C. Canada, 2008; http://www.cs.sfu.ca/research/publications/techreports/#2008
25. Walenz, B. and Didion, J. *OpenNLP*, 2008; http://jwordnet.sourceforge.net/

Alexander Hogenboom (hogenboom@ese.eur.nl) is a Ph.D. student at Erasmus University Rotterdam, the Netherlands, where he is affiliated with the Econometric Institute, the Erasmus Centre for Business Intelligence at the Erasmus Research Institute of Management, and the Erasmus Studio.

Flavius Frasincar (frasincar@ese.eur.nl) is an assistant professor of information systems at the Econometric Institute and founding member of the Erasmus Centre for Business Intelligence at the Erasmus Research Institute of Management of Erasmus University Rotterdam, the Netherlands.

Franciska de Jong (f.m.g.dejong@utwente.nl) is a full professor of language technology at the University of Twente, the Netherlands, and a professor of e-research for the social sciences and the humanities at Erasmus University Rotterdam, the Netherlands, where she is also director of the Erasmus Studio.

Uzay Kaymak (u.kaymak@ieee.org) is a full professor of information systems in health care in the Department of Industrial Engineering & Innovation Sciences at the Eindhoven University of Technology, the Netherlands.

DOI:10.1145/2699390

Theory on passwords has lagged practice, where large providers use back-end smarts to survive with imperfect technology.

BY JOSEPH BONNEAU, CORMAC HERLEY, PAUL C. VAN OORSCHOT, AND FRANK STAJANO

Passwords and the Evolution of Imperfect Authentication

PASSWORDS HAVE DOMINATED human-computer authentication for 50 years despite consensus among researchers that we need something more secure and deserve something more user friendly. Much published research has focused on specific aspects of the problem that can be easily formalized but do not actually havea major influence on real-world design goals, which are never authentication per se, but rather protection of user accounts and sensitive data. As an example of this disconnect, academic research often recommends strict password-composition policies (such as length requirements and mandating digits and nonalphabetic characters) despite the lack of evidence they actually reduce harm.

We argue that critically revisiting authentication as a whole and passwords' role therein is required to understand today's situation and provide a meaningful look ahead. Passwords were originally deployed in the 1960s for access to time-shared mainframe computers, an environment unrecognizable by today's Web users. Many practices have survived with few changes even if no longer appropriate.[9,19] While partly attributable to inertia, this also represents a failure of the academic literature to provide approaches that are convincingly better than current practices.

We identify as outdated two models that still underlie much of the current password literature. First is the model of a random user who draws passwords uniformly and independently from some set of possible passwords. It has resulted in overestimates of security against guessing and encouraged ineffectual policies aimed at strengthening users' password choices. The second is that of an offline attack against the password file. This model has inflated the importance of unthrottled password guessing relative to other threats (such as client malware, phishing, channel eavesdropping, and plaintext leaks from the back-end) that are more difficult to analyze but significantly more important in practice. Together, these models have inspired an awkward jumble of contradictory advice that is impossible for humans to follow.[1,19,30,36]

The focus of published research on clean, well-defined problems has caused the neglect of the messy complications of real-world Web authentication. This misplaced focus continues to hinder the applicability of password research to practice. Failure to recognize

» key insights

- **Simplistic models of user and attacker behaviors have led the research community to emphasize the wrong threats.**
- **Authentication is a classification problem amenable to machine learning, with many signals in addition to the password available to large Web services.**
- **Passwords will continue as a useful signal for the foreseeable future, where the goal is not impregnable security but reducing harm at acceptable cost.**

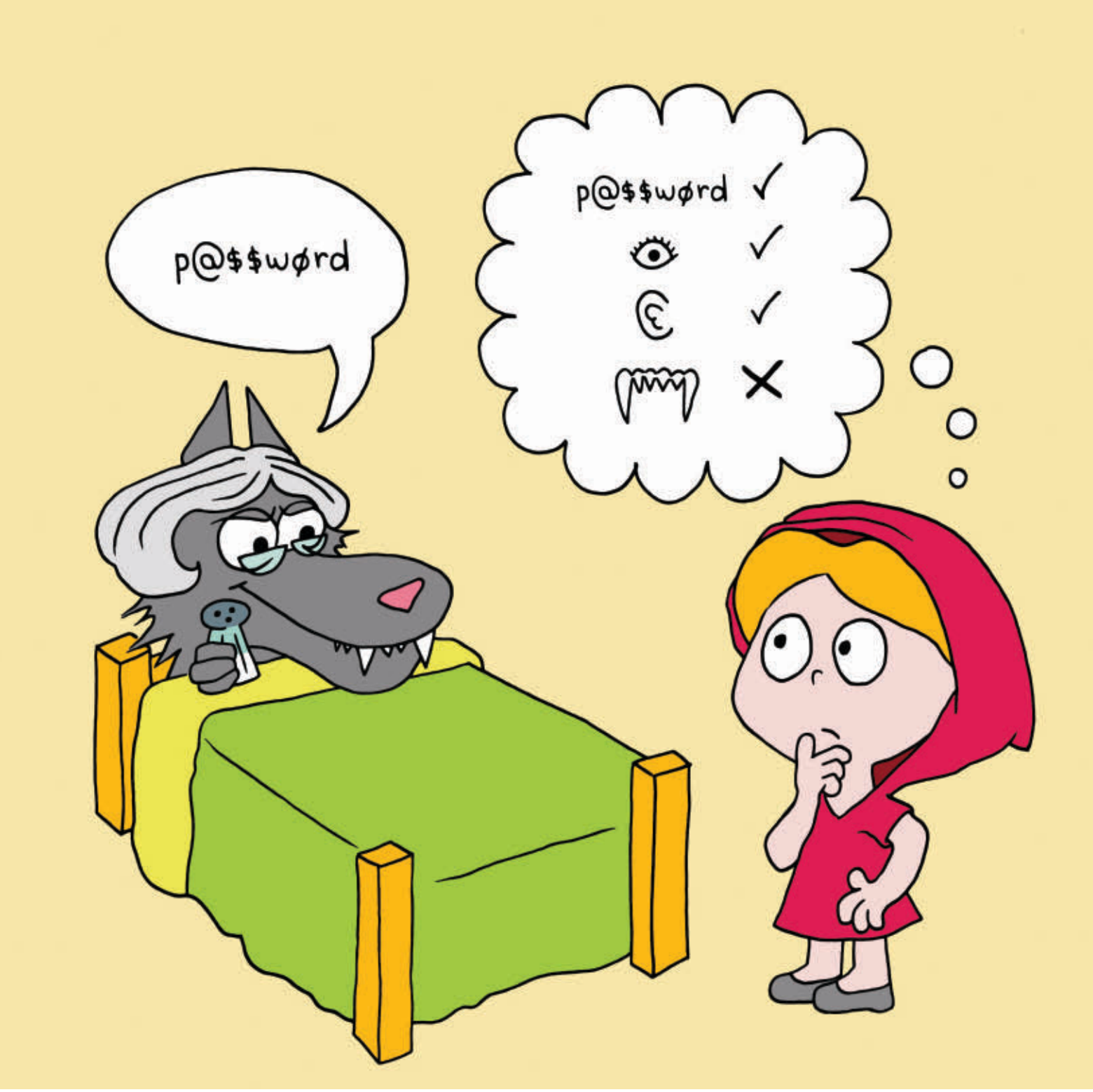


IMAGE BY IWONA USAKIEWICZ/ANDRIJ BORYS ASSOCIATES.
BASED ON IDEA BY JOSEPH BONNEAU AND FRANK STAJANO

the broad range of usability, deployability, and security challenges in Web authentication has produced a long list of mutually incompatible password requirements for users and countless attempts by researchers to find a magic-bullet solution despite drastically different requirements in different applications. No single technology is likely to "solve" authentication perfectly for all cases; a synergistic combination is required. Industry has already moved in this direction. Many leading providers have bolstered, not replaced, passwords with multiple parallel complementary authentication mechanisms. These are combined, often using machine learning, so as to minimize cost and annoyance while providing enough security for e-commerce and online social interaction to prosper. We expect authentication to gradually become more secure and less obtrusive, even if perhaps technically inelegant under the hood.

This trend is not without downsides. It strongly favors the largest providers with extensive knowledge of their users' habits. It makes authentication more privacy invasive and increasingly difficult to comprehend for users and researchers alike. We encourage researchers to acknowledge this trend and focus on addressing related security, privacy, and usability challenges.

Lessons from the Past

From the beginning, passwords have been a security band-aid. During de-

velopment of the first time-sharing operating systems in the 1960s, passwords were added to protect against practical jokes and researchers using more resources than authorized. The 1961 Compatible Time-Sharing System at MIT was likely the first to deploy passwords in this manner. Security issues arose immediately. Multiple cases were reported of users guessing one another's passwords and also at least one leak of the master password file that was stored in unencrypted form. Yet these issues were easily addressed administratively, as all users were part of the same academic organization.

With development of access control in MULTICS and Unix in the 1970s, passwords were adapted to protect sensitive data and computational resources. MULTICS protected passwords by storing them in hashed form, a practice invented by Roger Needham and Mike Guy at the University of Cambridge in the 1960s. Robert Morris's and Ken Thompson's seminal 1979 treatment of password security[25] described the evolution toward dedicated password hashing and salting via the `crypt()` function, along with the first analysis of dictionary attacks and brute-force guessing.

A decade later, the 1988 Morris Internet worm demonstrated the vulnerability of many systems to password guessing. Administrators adapted by storing password hashes in more heavily protected shadow password files[24] and, sometimes, proactively checking user passwords for guessability.[35]

With the mid-1990s advent of the World Wide Web and e-commerce, early attempts were made to replace passwords with public-key cryptography via secure sockets layer (SSL) client certificates or the competing secure electronic transaction (SET) protocol. Ultimately, managing certificates and private keys client-side proved too burdensome and a market never developed. Instead, secure connections on the Web almost universally rely on one-way authenticated SSL. Servers are authenticated by a certificate at the SSL layer, while users are left to prove their identity later with no explicit protocol support. Text-based passwords entered in HTML forms in exchange for HTTP cookies have become the dominant, albeit never formally specified, protocol for user authentication.

As Web-based services proliferated, usability problems arose that had not existed for system passwords. Resetting forgotten passwords, previously a manual task for IT support staff, was automated through email, creating a common central-point-of-failure for most users. The increased number of accounts held by individual users scuttled the assumption of dedicated passwords per account, and password reuse became commonplace. Phishing grew into a major concern, but antiphishing proposals requiring protocol or user-interface changes failed to gain adoption. Instead, the primary countermeasure involves blacklists of known phishing sites and machine-learning classifiers to recognize new phishing sites as they arise.[16]

Attempts to make a dedicated business out of authentication in the consumer space have failed consistently. While there has long been interest in deploying hardware tokens as a second factor, standalone tokens (such as RSA SecurID) have seen limited deployment outside enterprise environments, likely due to the cost of tokens relative to the value of free online accounts. Microsoft Passport and OpenID, among many attempts to offer single sign-on for the Web, have failed to gain mass adoption.

The widespread availability of smartphones may be changing the equation, however, as in the early 2010s a number of online services, including Facebook, Google, and Twitter deployed free smartphone applications to act as a second factor based on the emerging time-based one-time password (TOTP) standard.[27] Other services send codes via short message service (SMS) as a backup authentication mechanism. A few services have offered dedicated tokens as a second factor, typically in environments at greater risk of fraud (such as eBay and *World of Warcraft*).

Random models for user behavior. In addition to being regarded as the weak link in password systems, users are also typically the most difficult component to model. Ideally, they would choose passwords composed of random characters. But even researchers who acknowledge this is an idealized model have usually underestimated the gap between modeled behavior and reality. Security policies and research models have been slow to adjust to the magnitude of the inaccuracies revealed by new data sources (such as leaked password datasets and large-scale measurement studies).[5]

One of the best-known early sources on password policies is the U.S. Defense Department's circa 1985 *Green Book*,[38] which specified detailed policies for mitigating the risk of password guessing, including rate limiting, hashing passwords at rest, and limiting the lifetime of passwords. The *Green Book* avoided the complexity of user behavior altogether, putting forth as one of three main principles: "Since many user-created passwords are particularly easy to guess, all passwords should be machine-generated."

The same year, NIST published its Password Usage guidelines in the Federal Information Processing Standards (FIPS) series[39] that were heavily derived from the *Green Book*. In addition to the recommended machine-chosen passwords, the FIPS guidelines allowed user-chosen passwords with the caveat that users "shall be instructed to use a password selected from all acceptable passwords at random, if possible, or to select one that is not related to their personal identity, history, or environment." Today, nearly all nonmilitary applications allow user-chosen passwords due to the perceived difficulty of remembering machine-chosen passwords. Yet the FIPS guidelines retained most other recommendations from the Green Book unchanged, including calculations of password security based on allowed characters and length requirements, limits on password lifetime, and forced updates. This encouraged the unrealistically optimistic assumption that users choose passwords similarly to random password generators that has persisted to this day.

Estimating password strength via "entropy." The guessing resistance of user-chosen passwords is often estimated by modeling passwords as random choices from a uniform distribution. This enables straightforward calculations of expected guessing times in the tradition of the 1985 *Green Book*. To attempt to capture the fact many users choose from a relatively small number

of common passwords (despite the very large theoretical space from which to choose text passwords), researchers often choose a relatively small uniform distribution. The logarithm of the size of this uniform distribution in this model is often called "entropy," in reference to Claude Shannon's famous measure H_1 of the minimum average number of bits needed to encode symbols drawn from a distribution. Unfortunately, Shannon entropy models the number of guesses needed by an attacker who can check in constant time if an unknown password is a member of an arbitrarily sized set of possibilities. This does not correspond to any real guessing attack.

A more direct metric is "guesswork," G,[25] the expected number of queries by an adversary guessing individual passwords sequentially until all are found. It can be shown that H_1 provides a lower bound on G.[25] However, G is also problematic, as it can be highly skewed by rare but difficult-to-guess passwords.[5] To address this bias, "partial" (or "marginal") guessing metrics have been developed.[5,32] One formulation is partial guesswork (G_α), which models an attacker making only enough guesses to have a probability α of succeeding.[5] This encapsulates the traditional G when $\alpha = 1$, with lower values of α modeling more realistic attackers. Such metrics have been proven not to be lower-bounded in general by Shannon entropy.[5,32] While partial-guessing metrics provide an appropriate mathematical model of password-guessing difficulty, they require a very large sample to be estimated accurately, typically millions of passwords,[5] and have thus found limited practical use.

Heuristic measures of password strength are often needed for smaller datasets. NIST's *Electronic Authentication Guidelines*[8] (in many ways an update to the *Green Book*) acknowledged the mathematical unsoundness of Shannon entropy for guessing but still introduced a heuristic method for estimating "entropy" (NIST entropy) of password distributions under various composition policies. This model has since been used in many academic studies, though it has been found to produce relatively inaccurate estimates in practice.[23,40] The preferred empirical approach, albeit dependent on the configuration of a particular tool, is to simply run a popular open-source password-cracking library against a set of passwords and evaluate the average number of guesses needed to find a given proportion of them.[23,42] This approach can be applied to even a single password to evaluate its relative strength, though this clearly overestimates security relative to any real adversaries who use a more favorable cracking library.

Text-based passwords entered in HTML forms in exchange for HTTP cookies have become the dominant, albeit never formally specified, protocol for user authentication on the Web.

Improving password strength. Simple measures like Shannon and NIST entropy make increases in password strength seem tantalizingly close. Composition policies that increase the minimum length or expand the classes of characters a password seem to cause reliable increases in these measures if passwords are random; for example, the NIST guidelines suggest requiring at least one uppercase and non-alphabetic character. While acknowledging users may insert them in predictable places, the guidelines still estimate an increase in guessing difficulty of a password by six bits (or a factor of 64) compared to a policy of allowing any password. However, experiments have shown this is likely an overestimate by an order of magnitude.[40]

Such password policies persist despite imposing a high usability cost, discussed later, though tellingly, their use is far less common at sites facing greater competition (such as webmail providers[15]) than at sites with little competition (such as universities and government services). Instead, research suggests the most effective policy is simply using a large blacklist[34] to limit the frequency of the most common passwords, bounding online guessing attacks to a predictable level, and conceding that many users will choose passwords vulnerable to offline guessing.

A related goal has been to nudge users toward better passwords through feedback (such as graphical meters that indicate estimated strength of their password, as they choose it). In an experimental setting, very aggressive strength meters can make guessing user-chosen passwords dramatically more difficult.[37] However, in studies using meters typical of those found in practice, with users who were not prompted to consider their password, the impact of meters was negligible;

many users failed to notice them at all.[12] An empirical data point comes from Yahoo!, where adding a password-strength meter did improve password security but only marginally.[5]

Independence when choosing multiple passwords. A random user model often further assumes every password will be independently chosen. In practice, this is rarely true on the Web, as users cope with the large number of accounts by password reuse, sometimes with slight modification; for example, a 2007 telemetry study estimated the median user has 25 password-protected accounts but only six unique passwords.[14] This has direct security implications, as leaks at one website can compromise security at another. Even if a user has not exactly reused the password, attackers can guess small variations that may double their chances of success in an online-guessing scenario.[10] Related password choices similarly undermine the security goals of forced password updates, as an attacker with knowledge of a user's previous sequence of passwords can often easily guess the next password.[43]

Offline vs. online threats. The security literature distinguishes between online attackers who must interact with a legitimate party to authenticate and offline attackers who are limited only in terms of their computational resources.

Superficially, offline attackers are far more powerful, as they typically can make an unbounded number of guesses and compare them against a known hash of the password. Yet many additional avenues of attack are available to the online attacker: stealing the password using client-side malware, phishing the password using a spoofed site, eavesdropping the password as it is transmitted, stealing the password from the authentication server, stealing the password from a second authentication server where the user has reused it, and subverting the automated password reset process.

A critical observation is that strong passwords do not help against any of these other attacks. Even the strongest passwords are still static secrets that can be replayed and are equally vulnerable to phishing, theft, and eavesdropping. Mandating stronger passwords does nothing to increase security against such attacks.

In addition to being regarded as the weak link in password systems, users are also typically the most difficult component to model.

Offline guessing (cracking). Much attention has been devoted to devising strategies for picking passwords complex enough to resist offline cracking. Yet this countermeasure may stop real-world damage in at most a narrow set of circumstances.[13] For an attacker without access to the password file, any guessing must be done online, which can be rate-limited. If passwords in a leaked file are unhashed, they are exposed in plaintext regardless of complexity; if hashed but unsalted, then large "rainbow tables"[31] allow brute-force look-up up to some length.[a] Only if the attacker has obtained a password file that had been properly hashed and salted do password-cracking efficiency and password strength make a real difference. And yet, while hashing and salting have long been considered best practice by security professionals, they are far from universal. Empirical estimates suggest over 40% of sites store passwords unhashed;[7] recent large-scale password file leaks revealed many were plaintext (such as RockYou and Tianya), hashed but unsalted (such as LinkedIn), improperly hashed (such as Gawker), or reversibly encrypted (such as Adobe).

Finally, offline attackers may be interrupted if their breach is detected and administrators can force affected users to reset their passwords. Password resets are often not instituted at breached websites due to fear of losing users; they are even less commonly mandated for users whose password may have been leaked from a compromised third-party website where it may have been reused.

Online guessing. Online attackers can verify whether any given password guess is correct only by submitting it to the authentication server. The number of guesses that can be sent is limited. A crude "three strikes" model is an obvious way of throttling attacks, but relatively few sites implement such a deterministic policy,[7] probably to avoid denial of service.

Nonetheless, online guessing attacks are in some ways much more

a Freely available tables quickly reverse the MD5 hash of any alphanumeric password of up to 10 characters. Using the passwords from RockYou (a site that had a major password leak in 2009), we can estimate such a table would cover 99.7% of users for low-value online accounts.

costly to mount than offline attacks, on a per-guess basis. Whereas offline, an attacker might check a billion guesses on a single host, online an attacker might need thousands of hosts. First, if we assume IP addresses that send millions of failed attempts will be blocked, the load must be distributed. Also, the load could exceed legitimate traffic; in a service with one million users where the average user logs in once per day, a total of one billion guesses (one thousand guesses per account) is as many login requests as the legitimate population generates in three years. If legitimate users fail 5% or so of the time (due to, say, typos or forgetting) the attacker will generate as many fail events as the legitimate population generates in 60 years.

Choosing a password to withstand an offline attack is thus much more difficult than choosing one to withstand an online attack. Yet the additional effort pays off only in the very restricted circumstances in which offline attacks can occur.[13] It makes little sense to focus on this risk when offline attacks are dwarfed by other vectors (such as malware).

Today's "Overconstrained" World

Passwords offer compelling economic advantages over the alternatives, with lowest start-up and incremental costs per user. Due largely to their status as the incumbent solution, they also have clear "deployability" advantages (such as backward compatibility, interoperability, and no migration costs). But it is not these factors alone that are responsible for their longevity; the "password replacement problem" is both underspecified and overconstrained.[6,20]

It is underspecified in that there is no universally agreed set of concrete requirements covering diverse environments, technology platforms, cultures, and applications; for example, many authentication proposals become utterly unworkable on mobile devices with touchscreens, many Asian languages are now typed with constant graphical feedback that must be disabled during password entry, and many large websites must support both low-value forum accounts and important e-commerce or webmail accounts through a single system. It is simultaneously overconstrained in that no single solution can be expected to address all requirements, ranging from financial to privacy protection. The list of usability, deployability, and security requirements is simply too long (and rarely documented explicitly).

An in-depth review of 35 proposed password alternatives using a framework of 25 comparison criteria found no proposal beats passwords on all fronts.[6] Passwords appear to be a Pareto equilibrium, requiring some desirable property *X* be given up to gain any new benefit *Y*, making passwords very difficult to replace.

Reviewing how categories of these password alternatives compare to regular passwords yields insight. Password managers—software that can remember and automatically type passwords for users—may improve security and usability in the common case but are challenging for users to configure across all user agents. This problem also affects some graphical password schemes,[4] while others offer insufficient security gains to overcome change-resisting inertia. Biometric schemes, besides their significant deployment hurdles, appear poorly suited to the unsupervised environment of Web authentication; fraudsters can just replay digital representations of fingerprints or iris patterns. Schemes using hardware tokens or mobile phones to generate one-time access codes may be promising, with significant security advantages, but ubiquitous adoption remains elusive due to a combination of usability issues and cost.

Federated authentication, or "single sign-on" protocols, in which users are authenticated by delegation to central identity providers, could significantly reduce several problems with passwords without completely eliminating them. Yet, besides introducing serious privacy issues, they have been unable to offer a business model sufficiently appealing to relying sites. The most successful deployment to date, Facebook Connect (a version of OAuth), incentivizes relying parties with user data, mandating a central role for Facebook as sole identity provider, which does little for privacy.

With no clear winner satisfying all criteria, inertia becomes a substantial hurdle and the deck is stacked against technologies hoping to replace passwords entirely. A better choice is to prioritize competing requirements depending on organizational priorities and usage scenarios and aim for gradual adoption. Given their universal support as a base user-authentication mechanism, passwords are sensibly implemented first, offering the cheapest way to get things up and running when an idea is not yet proven and security is not yet critical, with no learning curve or interoperability hurdles. Low adoption costs also apply to users of new sites, who need low barriers when exploring new sites they are not sure they will return to. Financial websites are the rare exception, with offline capital and users whose accounts are all clearly valuable.

The list of challenges to would-be alternatives goes on. Improving security despite any decline in usability may mean losing potential new users (and sometimes existing users) to competitors. Some alternatives require server or client software modifications by various independent parties and are often a showstopper; some others expect large numbers of users to change their existing habits or be trained to use new mechanisms. However, some are only partial solutions or address only a subset of security threats; some are even less user friendly, though in new and different ways. Moreover, as mentioned earlier, some are more costly and bring other deployment challenges (such as interoperability, compatibility with existing infrastructure, and painful migration).

Advice to users. Users face a plethora of advice on passwords: use a different one for each account; change them often; use a mix of letters, punctuation, symbols, and digits; make them at least eight characters long; avoid personal information (such as names and birthdays); and do not write them down. These suggestions collectively pose an unrealistic burden and are sometimes mutually incompatible; a person cannot be expected to memorize a different complex password for each of, say, 50 accounts, let alone change all of them on a rolling basis. Popular wisdom has summarized the password advice of the security experts as "Pick something you cannot remember, and do not write it down."

Each bit of advice may be useful

against a specific threat, motivating security professionals to offer them in an attempt to cover their bases and avoid blame for any potential security breaches regardless of the inconvenience imposed on users. This approach is predicted to lead to failure by the "compliance budget" model[3] in which the willingness of each user to comply with annoying security requirements is a finite, exhaustible resource that should be managed as carefully as any other budget. Indeed, websites (such as online stores), whose users are free to vote with their wallets, are much more careful about not exceeding their customers' compliance budget than sites (such as universities) whose users are "captive."[15]

Useful security advice requires a mature risk-management perspective and rough quantification of the risks and costs associated with each countermeasure. It also requires acknowledging that, with passwords as deployed today, users have little control over the most important countermeasures. In particular, running a personal computer free of malware may be the most important step, though it is challenging and often ignored in favor of password advice, which is simpler to conceptualize but far less important. Likewise, good rate limiting and compromise detection at the server-side are critical, as discussed earlier, but users have no agency other than to patronize better-implemented sites.

Choosing extremely strong passwords, as is often advised, is of far more limited benefit; evidence it reduces harm in practice is elusive. As noted earlier, password cracking is rarely a critical step in attacks. Hence making passwords strong enough to resist dedicated cracking attacks seems an effort poorly rewarded for all but the most critical Web accounts. For important accounts, password-selection advice should encourage passwords not easily guessed by acquaintances and sufficient for withstanding a reasonable amount of online guessing, perhaps one million guesses. About half of users are already above this bar,[5] but discouraging simple dictionary passwords via advice, strength meters, and blacklists remains advisable to help out the others.

Advice to avoid reusing passwords is also common. While it is a good defense against cross-site password compromise, it is, for most users, incompatible with remembering passwords. Better advice is probably to avoid reusing passwords for important accounts and not to worry about the large number of accounts of little value to an attacker (or their owner).

Moreover, we consider the advice against writing passwords to be outmoded for the Web. Stories involving "Post-it notes on the monitor" usually refer to corporate passwords where users feel no personal stake in their security. Most users understand written-down passwords should be kept in a wallet or other safe location generally not available to others, even acquaintances. With this caveat, written passwords are a worthwhile trade-off if they encourage users to avoid very weak passwords. Password managers can be an even better trade-off, improving usability (no remembering, no typing) and allowing a different strong password for each account. However, they introduce a single point of failure, albeit perhaps no more vulnerable than Web accounts already due to the prevalence of email-based password reset.

Multidimensional Future

We appear stuck between the intractable difficulty of replacing passwords and their ever-increasing insecurity and burden on users. Many researchers have predicted the dam will burst soon and the industry will simply have to pay the necessary costs to replace passwords. However, these predictions have been made for over a decade.

The key to understanding how large service providers manage, using what appears to be a "broken" technology, is that websites do not need perfection. The problem of compromised accounts is just one of many forms of abuse, along with spam, phishing, click fraud, bots, fake accounts, scams, and payment fraud. None of them has been completely defeated technologically, but all are managed effectively enough to keep things running.

In nearly every case, techniques that "solve" the problem technically have lost out to ones that manage them statistically; for example, despite many proposals to end spam, including cryptographic protocols to prevent domain spoofing and microcharges for each email message sent, most email providers have settled for approaches that classify mail based on known patterns of attacker behavior. These defenses are not free or easy to implement, with large Web operators often devoting significant resources toward keeping pace with abuse as it evolves. Yet, ultimately, this cost is typically far less than any approach requiring users to change behavior.

In the case of authentication, banks provide a ready example of living with imperfect technology. Even though credit-card numbers are essentially static secrets, which users make no attempt to conceal from merchants, fraud is kept to acceptable levels by back-end classifiers. Technologies like "chip and PIN" have not been a magic bullet where deployed.[2] Cards are still stolen, PINs can be guessed or observed, signature transactions still exist as a fallback, and online payments without a PIN, or "card not present" transactions, are still widespread.

Yet banks survive with a non-binary authentication model where all available information is considered for each transaction on a best-effort basis. Web authentication is converging on a similar model, with passwords persisting as an imperfect signal supplemented by many others.

Web authentication as classification. Behind the scenes, many large websites have already transitioned to a risk-based model for user authentication. This approach emerged by the early 2000s at online financial sites.[41] While an incorrect password means access should be denied, a correct password is just one signal or feature that can be used by a classifier to determine whether or not the authentication attempt involves the genuine account owner.

The classifier can take advantage of many signals besides the password, including the user's IP address; geolocation; browser information, including cookies; the time of login; how the password is typed; and what resources are being requested. Unlike passwords, these implicit signals are available with no extra effort by the user.[21] Mobile devices introduce many new

signals from sensors that measure user interaction.[11] While none of these signals is unforgeable, each is useful; for example, geolocation can be faked by a determined adversary,[28] and browser fingerprinting techniques appear to be an endless arms race.[29] Nonetheless, both may be valuable in multidimensional solutions, as the difficulty of forging all signals can be significant in practice; for example, by combining 120 such signals, Google reported a 99.7% reduction in 2013 in the rate of accounts compromised by spammers.[18]

Unlike traditional password authentication, the outcome is not binary but a real-valued estimate of the likelihood the attempt is genuine. Generally these results will be discretized, as users must be given access to some resource or not, and any classifier will inevitably make false accept and false reject errors. Sites will continue to develop their machine-learning techniques to reduce these errors and may deploy new technology (such as two-factor authentication and origin-bound certificates[17]) to increase the number (and quality) of signals available.

Web authentication is by no means an easy domain to address for machine learning. The trade-off between false accepts and false rejects is difficult to get right. For financial sites, false accepts translate to fraud but can usually be recovered from by reversing any fraudulent payments. However, for sites where false accepts result in disclosure of sensitive user data, the confidentiality violations can never be undone, making them potentially very costly. Meanwhile, false rejects annoy customers, who may switch to competitors.

Obtaining a large sample of ground truth to train the classifier is another challenge, as it is difficult to find examples of attacks administrators do not yet know about. Financially motivated attackers are again likely the easiest to deal with, as their attacks typically must be scalable, leading to a large volume of attacks and hence training data. Nonfinancially motivated attackers (such as ex-partners) may be more difficult to detect algorithmically, but users are far better positioned to deal with them in real life. Targeted attacks, including "advanced persistent threats," which are technically sophisticated and aimed at a single user account, are the most difficult challenge, as attackers can tailor techniques to victims and leave relatively little signal available for classifiers.

Only if the attacker has obtained a password file that had been properly hashed and salted do password-cracking efficiency and password strength make a real difference.

New modes of operation. Authentication by classification enables fundamentally new methods of operation. Authentication can be a more flexible process, with additional information demanded as needed if the classifier's confidence is low or the user attempts a particularly sensitive operation, a process called "progressive authentication";[33] for example, a site may ask users to confirm their identity by SMS or phone call if it notices suspicious concurrent activity on their account from geographically distant locations. "Multi-level authentication" becomes possible, with users given limited access when the classifier's confidence is relatively low. In the U.K., some banks offer users a read-only view of their account with just a password but require a security token to actually transfer money out.

Sites may also ask for less information, including not requiring a password to be entered, when they have reasonable confidence, from secondary signals, the correct user is present. A form of this is already in place—where persistent session cookies once allowed password-less login for a predetermined duration, the decision of when to recheck the password is now made dynamically by a risk classifier instead. A stronger version is opportunistic two-factor authentication, ensuring correct authentication when the second factor is present but enabling fallback security if the password is still correct and enough additional signals are presented.[17]

The limit of this evolution is "continual authentication." Instead of simply checking passwords at the entrance gate, the classifier can monitor the actions of users after letting them in and refining its decision based on these additional signals. Ultimately, continual authentication may mean the authentication process grows directly intertwined with other abuse-detection systems.

Changes to the user experience. As sites aim to make correct authentication decisions "magically" on the back-

end through machine learning, most changes to the user experience should be positive. Common cases will be increasingly streamlined; users will be asked to type passwords (or take other explicit action) as rarely as possible. However, users also face potential downsides, as systems grow increasingly opaque and difficult to understand.

First, users may see more requests for second factors (such as a one-time code over SMS) when the classifier's confidence is low. Users may also face more cases (such as when traveling or switching to a new machine) where they are unable to access their own account despite correctly remembering their password, akin to unexpected credit-card rejections while abroad. Increased rejections may increase the burden on "fallback" authentication, to which we still lack a satisfactory solution.

As authentication systems grow in complexity, their automated decisions may cause users increased confusion and distress. Users are less likely to buy into any system that presents them with inconveniences they do not understand. Training users to respond with their credentials to asynchronous security challenges on alternative channels may also pave the way for novel phishing attacks. Even with careful user interface design, users may end up confused as to what the genuine authentication ceremony[22] should be.

Another challenge is that better classifiers may break some access-control practices on top of passwords users have grown accustomed to; for example, users who share passwords with their spouses or their assistants may face difficulty if classifiers are able to (correctly) determine another human is using their password, even though this is what the user intended.

Finally, typing passwords less often could in fact decrease their usability, as users are more likely to forget them if they go long periods between needing to type them.

Advantages of scale. Authentication may represent a classic example of the winner-take-all characteristics that appear elsewhere on the Web, since it offers benefits to scale in two different ways: First, large services are more likely to be accepted by relying parties as an identification service. Being accepted by more relying parties in turn encourages users to register accounts, further enhancing the attractiveness of these identity providers to relying parties. The second, "two-sided market," or positive feedback loop, is for user data. Large services with more user data can provide more accurate authentication. This attracts users to interact with the services more frequently, providing even more data for authentication. Developing and maintaining a complex machine-learning-based approach to authentication requires a relatively large fixed technical cost and low marginal costs per user, further advantaging the largest identity providers.

Targeted attacks, including "advanced persistent threats," technically sophisticated and aimed at a single user account, are the most difficult challenge, as they can tailor techniques to victims and leave relatively little signal available for classifiers.

One consequence of this consolidation is that, lacking access to the volumes of real-world data collected by larger service providers, independent researchers may be limited in their ability to contribute to several important research topics for which the limits of artificial datasets and mental extrapolation make empirically grounded research essential. Other areas of Web research (such as networking, which requires massive packet capture or search-engine research requiring huge numbers of user queries) have likewise become difficult for researchers with access to only public data sources.

There are also troubling privacy implications if relying parties require users to sign up with a large service that, in turn, requires a significant amount of personal information to perform authentication well. This information may be inherently sensitive (such as time and location of login activity) or difficult to change if leaked (such as behavioral biometrics like typing patterns). Many users already trust highly sensitive information to large online services, but authentication may be a motivating factor to collect more data, store it longer, and share it with more parties.

Conclusion

Passwords offer plenty of examples of divergence between theory and practice; estimates of strength, models of user behavior, and password-composition policies that work well in theory generally remain unsupported by

evidence of reduced harm in practice and have in some cases been directly contradicted by empirical observation. Yet large Web services appear to cope with insecure passwords largely because shortcomings can be covered up with technological smarts in the back-end. This is a crucial, if unheralded, evolution, driven largely by industry, which is well experienced in data-driven engineering. Researchers who adapt their models and assumptions to reflect this trend will be in a stronger position to deliver relevant results. This evolution is still in its early stages, and there are many important and interesting questions about the long-term results that have received little or no study to this point. There is also scope for even more radical rethinking of user authentication on the Web; clean-slate approaches may seem too risky for large companies but can be explored by more agile academic researchers. Tackling these novel challenges is important for ensuring published research is ahead of industry practice, rather than the other way around.

Acknowledgments

Joseph Bonneau is funded by a Secure Usability Fellowship from Simply Secure and the Open Technology Fund. Paul van Oorschot is funded by a Natural Sciences and Engineering Research Council of Canada Chair in Authentication and Computer Security and a Discovery Grant. Frank Stajano is partly supported by European Research Council grant 307224.

References

1. Adams, A. and Sasse, M. Users are not the enemy. *Commun. ACM 42*, 12 (Dec. 1999), 41–46.
2. Anderson, R., Bond, M., and Murdoch, S.J. Chip and spin. *Computer Security Journal 22*, 2 (2006), 1–6.
3. Beautement, A., Sasse, M.A., and Wonham, M. The compliance budget: Managing security behaviour in organisations. In *Proceedings of the New Security Paradigms Workshop* (Lake Tahoe, CA, 2008).
4. Biddle, R., Chiasson, S., and van Oorschot, P.C. Graphical passwords: Learning from the first 12 years. *ACM Computing Surveys 44*, 4 (Aug. 2012), article 19:1–41.
5. Bonneau, J. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *Proceedings of the IEEE Symposium on Security and Privacy* (San Francisco, CA, May 20–23). IEEE Press, 2012.
6. Bonneau, J., Herley, C., van Oorschot, P.C., and Stajano, F. The quest to replace passwords: A framework for comparative evaluation of Web authentication schemes. In *Proceedings of the IEEE Symposium on Security and Privacy* (San Francisco, CA, May 20–23). IEEE Press, 2012.
7. Bonneau, J. and Preibusch, S. The password thicket: Technical and market failures in human authentication on the Web. In *Proceedings of the Workshop on the Economics of Information Security* (Arlington, VA, June 14–15, 2010).
8. Burr, W.E., Dodson, D.F., Newton, E.M., Perlner, R.A., Polk, W.T., Gupta, S., and Nabbus, E.A. *Electronic Authentication Guideline*. National Institute of Standards and Technology S.P. 800-63-2, Gaithersburg, MD, 2013.
9. Cheswick, W.R. Rethinking passwords. *Commun. ACM 56*, 2 (Feb. 2013), 40–44.
10. Das, A., Bonneau, J., Caesar, M., Borisov, N., and Wang, X. The tangled web of password reuse. In *Proceedings of the Network and Distributed System Security Symposium* (San Diego, CA, Feb. 23–26). Internet Society, Reston, VA, 2014.
11. De Luca, A., Hang, A., Brudy, F., Lindner, C., and Hussmann, H. Touch me once and I know it's you!: Implicit authentication based on touchscreen patterns. In *Proceedings of the ACM CHI Conference* (Austin, TX, May 5–10). ACM Press, New York, 2012.
12. Egelman, A., Sotirakopoulos, A., Muslukhov, I., Beznosov, K., and Herley, C. Does my password go up to eleven?: The impact of password meters on password selection. In *Proceedings of the ACM CHI Conference* (Paris, France, Apr. 27–May 2). ACM Press, New York, 2013.
13. Florêncio, D., Herley, C., and van Oorschot, P.C. An administrator's guide to Internet password research. In *Proceedings of the 28th Large Installation System Administration Conference* (Seattle, WA, Nov. 9–14). USENIX Association, Berkeley, CA, 2014.
14. Florêncio, D. and Herley, C. A large-scale study of Web password habits. In *Proceedings of the 16th International Conference on the World Wide Web* (Banff, Alberta, Canada, May 8–12, 2007).
15. Florêncio, D. and Herley, C. Where do security policies come from? In *Proceedings of the ACM Symposium On Usable Privacy and Security* (Redmond, WA, July 14–16). ACM Press, New York, 2010.
16. Garera, S., Provos, N., Chew, M., and Rubin, A.D. A framework for detection and measurement of phishing attacks. In *Proceedings of the Fifth ACM Workshop on Recurring Malcode* (Alexandria, VA). ACM Press, New York, 2007.
17. Grosse, E. and Upadhyay, M. Authentication at scale. *IEEE Security & Privacy Magazine 11* (2013), 15–22.
18. Hearn, M. An update on our war against account hijackers. Google Security Team blog, Feb. 2013.
19. Herley, C. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *Proceedings of the ACM New Security Paradigms Workshop* (Oxford, U.K., Sept. 8–11). ACM Press, New York, 2009.
20. Herley, C. and van Oorschot, P.C. A research agenda acknowledging the persistence of passwords. *IEEE Security & Privacy 10*, 1 (2012), 28–36.
21. Jakobsson, M., Shi, E., Golle, P., and Chow, R. Implicit authentication for mobile devices. In *Proceedings of the Fourth USENIX Workshop on Hot Topics in Security* (Montreal, Canada, Aug. 11). USENIX Association, Berkeley, CA, 2009.
22. Karlof, C., Tygar, J.D., and Wagner, D. Conditioned-safe ceremonies and a user study of an application to Web authentication. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium* (San Diego, CA, Feb. 8–11). Internet Society, Reston, VA, 2009.
23. Kelley, P.G., Komanduri, S., Mazurek, M.L., Shay, R., Vidas, T., Bauer, L., Christin, N., Cranor, L.F., and Lopez, J. Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms. In *Proceedings of the IEEE Symposium on Security and Privacy* (San Francisco, CA, May 20–23). IEEE Press, 2012.
24. Klein, D. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the Second USENIX Security Workshop*. USENIX Association, Berkeley, CA, 1990.
25. Massey, J.L. Guessing and entropy. In *Proceedings of the 1994 IEEE International Symposium on Information Theory* (June 27–July 1). IEEE Press, 1994, 204.
26. Morris, R. and Thompson, K. Password security: A case history. *Commun. ACM 22*, 11 (1979), 594–597.
27. M'Raihi, D., Machani, S., Pei, M., and Rydell, J. *TOTP: Time-Based One-Time Password Algorithm, RFC 6238*. The Internet Engineering Task Force, Fremont, CA, May 2011.
28. Muir, J.A. and van Oorschot, P.C. Internet geolocation: Evasion and counterevasion. *ACM Computer Surveys 42*, 1 (Dec. 2009), 4:1–13.
29. Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., and Vigna, G. Cookieless monster: Exploring the ecosystem of Web-based device fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy* (Berkeley, CA, May 19–22). IEEE Press, 2013.
30. Norman, D.A. The way I see it: When security gets in the way. *Interactions 16*, 6 (Nov. 2009), 60–63.
31. Oechslin, P. Making a faster cryptanalytic time-memory trade-off. In *Proceedings of the 23rd Annual International Advances in Cryptology* (Santa Barbara, CA, Aug. 17–21), Springer, 2003.
32. Pliam, J.O. On the incomparability of entropy and marginal guesswork in brute-force attacks. In *Proceedings of INDOCRYPT: The First International Conference on Cryptology in India* (Calcutta, India, Dec. 10–13). Springer, 2000.
33. Riva, O., Qin, C., Strauss, K., and Lymberopoulos, D. Progressive authentication: Deciding when to authenticate on mobile phones. In *Proceedings of the 21st USENIX Security Symposium* (Bellevue, WA, Aug. 8–10). USENIX Society, Berkeley, CA, 2012.
34. Schechter, S., Herley, C., and Mitzenmacher, M. Popularity is everything: A new approach to protecting passwords from statistical-guessing attacks. In *Proceedings of the Fifth USENIX Workshop on Hot Topics in Security* (Washington, D.C., Aug. 10). USENIX Society, Berkeley, CA, 2010.
35. Spafford, E. Observations on reusable password choices. In *Proceedings of the USENIX Security Workshop*. USENIX Society, Berkeley, CA, 1992.
36. Stajano, F. Pico: No more passwords! In *Proceedings of the 19th International Security Protocols Workshop Lecture Notes in Computer Science 7114* (Cambridge, U.K., Mar. 28–30). Springer, 2011.
37. Ur, B., Kelley, P.G., Komanduri, S., Lee, J., Maass, M., Mazurek, M.L., Passaro, T., Shay, R., Vidas, T., Bauer, L., Christin, N., and Cranor, L.F. How does your password measure up? The effect of strength meters on password creation. In *Proceedings of the USENIX Security Symposium* (Bellevue, WA, Aug. 8–10). USENIX Society, Berkeley, CA, 2012.
38. U.S. Department of Defense. *Password Management Guideline. Technical Report CSC-STD-002-85*. Washington, D.C., 1985.
39. U.S. National Institute of Standards and Technology. *Password Usage. Federal Information Processing Standards Publication 112*. Gaithersburg, MD, May 1985.
40. Weir, M., Aggarwal, S., Collins, M., and Stern, H. Testing metrics for password creation policies by attacking large sets of revealed passwords. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (Chicago, IL, Oct. 4–8). ACM Press, New York, 2010.
41. Williamson, G.D. Enhanced authentication in online banking. *Journal of Economic Crime Management 4*, 2 (2006).
42. Yan, J., Blackwell, A., Anderson, R., and Grant, A. Password memorability and security: Empirical results. *IEEE Security & Privacy 2*, 5 (2004), 25–31.
43. Zhang, Y., Monrose, F., and Reiter, M. The security of modern password expiration: An algorithmic framework and empirical analysis. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (Chicago, IL, Oct. 4–8). ACM Press, New York, 2010.

Joseph Bonneau (jbonneau@cs.stanford.edu) is a postdoctoral research fellow at Stanford University, Stanford, CA, and a technology fellow at the Electronic Frontier Foundation, San Francisco, CA.

Cormac Herley (cormac@microsoft.com) is a principal researcher at Microsoft Research, Redmond, WA.

Paul C. van Oorschot (paulv@scs.carleton.ca) is a professor of computer science and Canada Research Chair in Authentication and Computer Security in the School of Computer Science at Carleton University, Ottawa, Ontario, Canada.

Frank Stajano (frank.stajano@cl.cam.ac.uk) is a Reader in Security and Privacy at the University of Cambridge, Cambridge, U.K., where he is also the head of the Academic Centre of Excellence in Cyber Security Research, and a Fellow of Trinity College.

DOI:10.1145/2699411

Open-universe probability models show merit in unifying efforts.

BY STUART RUSSELL

Unifying Logic and Probability

PERHAPS THE MOST enduring idea from the early days of AI is that of a *declarative* system reasoning over explicitly represented knowledge with a general inference engine. Such systems require a formal language to describe the real world; and *the real world has things in it*. For this reason, classical AI adopted first-order logic—the mathematics of objects and relations—as its foundation.

The key benefit of first-order logic is its expressive power, which leads to concise—and hence learnable—models. For example, the rules of chess occupy 10^0 pages in first-order logic, 10^5 pages in propositional logic, and 10^{38} pages in the language of finite automata. The power comes from separating predicates from their arguments and quantifying over those arguments: so one can write rules about $On(p, c, x, y, t)$ (piece p of color c is on square x, y at move t) without filling in each specific value for c, p, x, y, and t.

Modern AI research has addressed another important property of the real world—*pervasive uncertainty* about both its state and its dynamics—using probability theory. A key step was Pearl's development of *Bayesian networks*, which provided the beginnings of a formal language for probability models and enabled rapid progress in reasoning, learning, vision, and language understanding. The expressive power of Bayes nets is, however, limited. They assume a fixed set of *variables*, each taking a value from a fixed *range*; thus, they are a *propositional* formalism, like Boolean circuits. The rules of chess and of many other domains are beyond them.

What happened next, of course, is that classical AI researchers noticed the pervasive uncertainty, while modern AI researchers noticed, or remembered, that the world has things in it. Both traditions arrived at the same place: *the world is uncertain* **and** *it has things in it*. To deal with this, we have to *unify logic and probability*.

But how? Even the meaning of such a goal is unclear. Early attempts by Leibniz, Bernoulli, De Morgan, Boole, Peirce, Keynes, and Carnap (surveyed by Hailperin[12] and Howson[14]) involved attaching probabilities to logical sentences. This line of work influenced AI

» key insights

- **First-order logic and probability theory have addressed complementary aspects of knowledge representation and reasoning: the ability to describe complex domains concisely in terms of objects and relations and the ability to handle uncertain information. Their unification holds enormous promise for AI.**
- **New languages for defining open-universe probability models appear to provide the desired unification in a natural way. As a bonus, they support probabilistic reasoning about the existence and identity of objects, which is important for any system trying to understand the world through perceptual or textual inputs.**

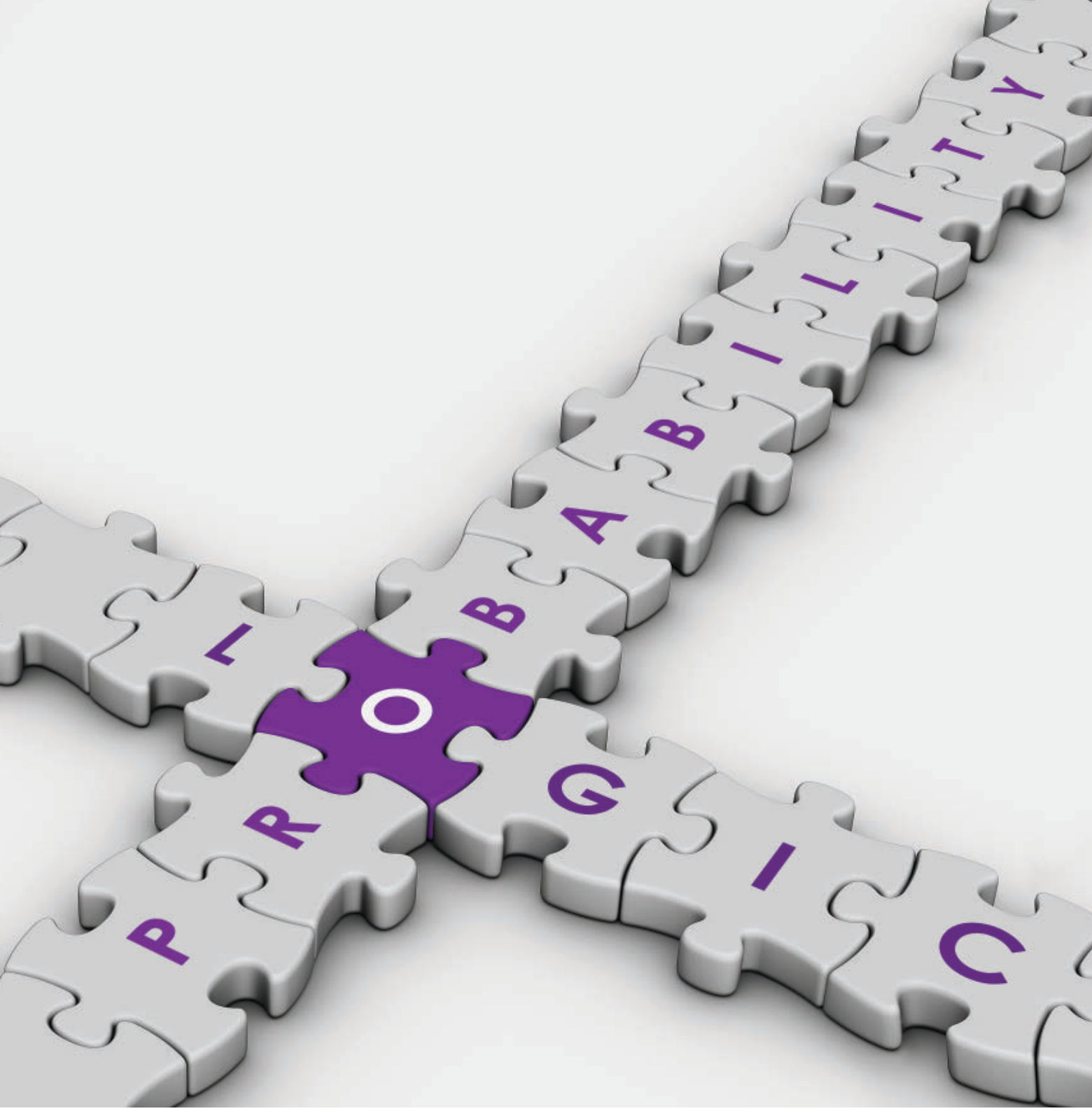


IMAGE BY ALMAGAMI

research but has serious shortcomings as a vehicle for representing knowledge.

An alternative approach, arising from both branches of AI and from statistics, combines the syntactic and semantic devices of logic (composable function symbols, logical variables, quantifiers) with the compositional semantics of Bayes nets. The resulting languages enable the construction of very large probability models and have changed the way in which real-world data are analyzed.

Despite their successes, these approaches miss an important consequence of uncertainty in a world of things: *uncertainty about what things are in the world*. Real objects seldom wear unique identifiers or preannounce their existence like the cast of a play. In areas such as vision, language understanding, Web mining, and computer security, the existence of objects must be *inferred* from raw data (pixels, strings, and so on) that contain no explicit object references.

The difference between knowing all the objects in advance and inferring their existence from observation corresponds to the distinction between *closed-universe* languages such as SQL and logic programs and *open-universe* languages such as full first-order logic. This article focuses in particular on open-universe probability models. The section on Open-Universe Models describes a formal language, Bayesian logic or BLOG, for writing such models.[21]

It gives several examples, including (in simplified form) a global seismic monitoring system for the Comprehensive Nuclear Test-Ban Treaty.

Logic and Probability

This section explains the core concepts of logic and probability, beginning with *possible worlds*.[a] A possible world is a formal object (think "data structure") with respect to which the truth of any assertion can be evaluated.

For the language of propositional logic, in which sentences are composed from proposition symbols $X_1, \ldots, X_n$ joined by logical connectives ($\wedge$, $\vee$, $\neg$, $\Rightarrow$, $\Leftrightarrow$), the possible worlds $\omega \in \Omega$ are all possible assignments of *true* and *false* to the symbols. First-order logic adds the notion of *terms*, that is, expressions referring to objects; a term is a constant symbol, a logical variable, or a k-ary function applied to k terms as arguments. Proposition symbols are replaced by atomic sentences, consisting of either predicate symbols applied to terms or equality between terms. Thus, *Parent*(*Bill*, *William*) and *Father*(*William*) = *Bill* are atomic sentences. The quantifiers $\forall$ and $\exists$ make assertions across all objects, for example,

$$\forall p, c \;\; (Parent(p, c) \wedge Male(p)) \Leftrightarrow Father(c) = p.$$

For first-order logic, a possible world specifies (1) a set of domain elements (or objects) $o_1, o_2, \ldots$ and (2) mappings from the constant symbols to the domain elements and from the function and predicate symbols to functions and relations on the domain elements. Figure 1a shows a simple example with two constants and one binary predicate. Notice that first-order logic is an open-universe language: even though there are two constant symbols, the possible worlds allow for 1, 2, or indeed arbitrarily many objects. A closed-universe language enforces additional assumptions:

- The *unique names* assumption requires that distinct terms must refer to distinct objects.
- The *domain closure* assumption requires that there are no objects other than those named by terms.

These two assumptions force every world to contain the same objects, which are in one-to-one correspondence with the ground terms of the language (see Figure 1b).[b] Obviously, the set of worlds under open-universe semantics is larger and more heterogeneous, which makes the task of defining open-universe probability models more challenging.

The formal semantics of a logical language define the truth value of a sentence in a possible world. For example, the first-order sentence $A = B$ is true in world ω iff A and B refer to the same object in ω; thus, it is true in the first three worlds of Figure 1a and false in the fourth. (It is *always* false under closed-universe semantics.) Let $T(\alpha)$ be the set of worlds where the sentence α is true; then one sentence α *entails* another sentence β, written $\alpha \models \beta$, if $T(\alpha) \subseteq T(\beta)$. Logical inference algorithms generally determine whether a *query* sentence is entailed by the known sentences.

In probability theory, a *probability model* P for a countable space Ω of possible worlds assigns a probability $P(\omega)$ to each world, such that $0 \leq P(\omega) \leq 1$ and $\Sigma_{\omega \in \Omega} P(\omega) = 1$. Given a probability model, the probability of a logical sentence α is the total probability of all worlds in which α is true:

$$P(\alpha) = \Sigma_{\omega \in T(\alpha)} P(\omega). \qquad (1)$$

The *conditional* probability of one sentence given another is $P(\alpha | \beta) = P(\alpha \wedge \beta) / P(\beta)$, provided $P(\beta) > 0$. A *random variable* is a function from possible worlds to a fixed range of values; for example, one might define the Boolean random variable $V_{A=B}$ to have the value *true* in the first three worlds of Figure 1a and *false* in the fourth. The *distribution* of a random variable is the set of probabilities associated with each of its possible values. For example, suppose the variable *Coin* has values 0 (heads) and 1 (tails). Then the assertion *Coin* $\sim$ *Bernoulli*(0.6) says that *Coin* has a Bernoulli distribution with parameter 0.6, that is, a probability 0.6 for the value 1 and 0.4 for the value 0.

a In logic, a possible world may be called a *model* or *structure*; in probability theory, a *sample point*. To avoid confusion, this paper uses "model" to refer only to probability models.

b The open/closed distinction can also be illustrated with a common-sense example. Suppose a system knows that *Father*(*William*) = *Bill* and *Father*(*Junior*) = *Bill*. How many children does Bill have? Under closed-universe semantics—for example, in a database—he has exactly two; under open-universe semantics, between 1 and ∞.

Figure 1. (a) Some of the infinitely many possible worlds for a first-order, open-universe language with two constant symbols, *A* and *B*, and one binary predicate *R*(*x*, *y*). Gray arrows indicate the interpretations of *A* and *B* and black arrows connect pairs of objects satisfying *R*. (b) The analogous figure under closed-universe semantics; here, there are exactly four possible *x*, *y*-pairs and hence $2^4 = 16$ worlds.

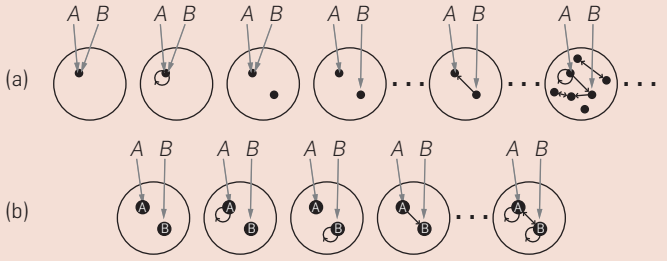
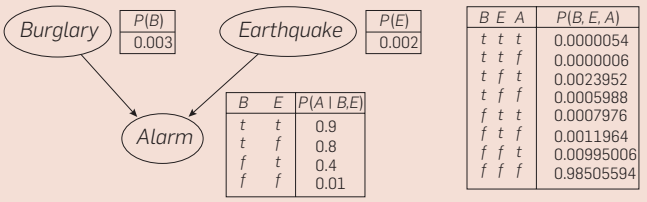


Figure 2. Left: a Bayes net with three Boolean variables, showing the conditional probability of *true* for each variable given its parents. Right: the joint distribution defined by Equation (2).



Unlike logic, probability theory lacks broad agreement on syntactic forms for expressing nontrivial assertions. For communication among statisticians, a combination of English and LaTeX usually suffices; but precise language definitions are needed as the "input format" for general probabilistic reasoning systems and as the "output format" for general learning systems. As noted, *Bayesian networks*[27] provided a (partial) syntax and semantics for the propositional case. The syntax of a Bayes net for random variables $X_1, \ldots, X_n$ consists of a directed, acyclic graph whose nodes correspond to the random variables, together with associated local conditional distributions.[c] The semantics specifies a joint distribution over the variables as follows:

$$P(\omega) = P(x_1, \ldots, x_n) = \prod_i P(x_i | \text{parents}(X_i)). \quad (2)$$

These definitions have the desirable property that *every well-formed Bayes net corresponds to a proper probability model on the associated Cartesian product space*. Moreover, a sparse graph—reflecting a sparse causal structure in the underlying domain—leads to a representation that is exponentially smaller than the corresponding complete enumeration.

The Bayes net example in Figure 2 (due to Pearl) shows two independent causes, *Earthquake* and *Burglary*, that influence whether or not an *Alarm* sounds in Professor Pearl's house. According to Equation (2), the joint probability $P(Burglary, Earthquake, Alarm)$ is given by

$$P(Burglary)\, P(Earthquake)\, P(Alarm \mid Burglary, Earthquake).$$

The results of this calculation are shown in the figure. Notice the eight possible worlds are the same ones that would exist for a propositional logic theory with the same symbols.

A Bayes net is more than just a specification for a distribution over worlds; it is also a stochastic "machine" for *generating* worlds. By sampling the variables in topological order (that is, parents before children), one generates a world exactly according to the distribution defined in Equation (2). This generative view will be helpful in extending Bayes nets to the first-order case.

Adding Probabilities to Logic

Early attempts to unify logic and probability attached probabilities directly to logical sentences. The first rigorous treatment, Gaifman's propositional *probability logic*,[9] was augmented with algorithmic analysis by Hailperin[12] and Nilsson.[23] In such a logic, one can assert, for example,

$$P(Burglary \Rightarrow Earthquake) = 0.997006, \quad (3)$$

a claim that is implicit in the model of Figure 2. The sentence $Burglary \Rightarrow Earthquake$ is true in six of the eight possible worlds; so, by Equation (1), assertion (3) is equivalent to

$$P(ttt) + P(ttf) + P(ftt) + P(ftf) + P(fft) + P(fff) = 0.997006.$$

Because any particular probability model μ assigns a probability to every possible world, such a constraint will be either true or false in μ; thus, μ, a distribution over possible propositional worlds, acts as a single possible world, with respect to which the truth of any probability assertion can be evaluated. Entailment between probability assertions is then defined in exactly the same way as in ordinary logic; thus, assertion (3) entails the assertion

$$P(Burglary \wedge Earthquake) \leq 0.997006,$$

because the latter is true in every probability model in which assertion (3) holds. Satisfiability of sets of such assertions can be determined by linear programming.[12] Hence, we have a "probability logic" in the same sense as "temporal logic"—that is, a deductive logical system specialized for reasoning with probabilistic assertions.

To apply probability logic to tasks such as proving the theorems of probability theory, a more expressive language was needed. Gaifman[8] proposed a *first-order* probability logic, with possible worlds being first-order model structures and with probabilities attached to sentences of (function-free) first-order logic.

Within AI, the most direct descendant of these ideas appears in Lukasiewicz's *probabilistic logic programs*, in which a probability range is attached to each first-order Horn clause and inference is performed by solving linear programs, as suggested by Hailperin. Within the subfield of *probabilistic databases* one also finds logical sentences labeled with probabilities[6]—but in this case probabilities are attached directly to the tuples of the database. (In AI and statistics, probability is attached to general relationships whereas observations are viewed as incontrovertible evidence.) Although probabilistic databases can model complex dependencies, in practice one often finds such systems using global independence assumptions across tuples.

Halpern[13] and Bacchus[3] adopted and extended Gaifman's technical approach, adding *probability expressions* to the logic. Thus, one can write

$$\forall h_1, h_2\ Burglary(h_1) \wedge \neg Burglary(h_2) \Rightarrow P(Alarm(h_1)) > P(Alarm(h_2)),$$

where now *Burglary* and *Alarm* are predicates applying to individual houses. The new language is more expressive but does not resolve the difficulty that Gaifman faced—how to define complete and consistent probability models. Each inequality *constrains* the underlying probability model to lie in a half-space in the high-dimensional space of probability models. Conjoining assertions corresponds to intersecting the constraints. Ensuring that the intersection yields a single point is not easy. In fact, Gaifman's principal result[8] is a single probability model requiring (1) a probability for every possible ground sentence and (2) probability constraints for infinitely many existentially quantified sentences.

Researchers have explored two solutions to this problem. The first involves writing a partial theory and then "completing" it by picking out one canonical model in the allowed set. Nilsson[23] proposed choosing the *maximum entropy* model consistent with the specified constraints. Paskin[24] developed a "maximum-entropy probabilistic logic" with constraints expressed as weights (relative probabilities) attached to first-order clauses. Such models are often called *Markov logic networks* or MLNs[30] and have become a popular technique for applications involving relational data. There is, however, a semantic difficulty with such models: weights trained in one scenario do not generalize to scenarios with different numbers of objects.

c Texts on Bayes nets typically do not define a syntax for local conditional distributions other than tables, although Bayes net software packages do.

Moreover, by adding irrelevant objects to a scenario, one can change the model's predictions for a given query to an arbitrary extent.[16,20]

A second approach, which happens to avoid the problems just noted,[d] builds on the fact that every well-formed Bayes net necessarily defines a unique probability distribution—a *complete theory* in the terminology of probability logics—over the variables it contains. The next section describes how this property can be combined with the expressive power of first-order logical notation.

Bayes Nets with Quantifiers

Soon after their introduction, researchers developing Bayes nets for applications came up against the limitations of a propositional language. For example, suppose that in Figure 2 there are many houses in the same general area as Professor Pearl's: each one needs an *Alarm* variable and a *Burglary* variable with the same CPTs and connected to the *Earthquake* variable in the same way. In a propositional language, this *repeated structure* has to be built manually, one variable at a time. The same problem arises with models for sequential data such as text and time series, which contain sequences of identical submodels, and in models for Bayesian parameter learning, where every instance variable is influenced by the parameter variables in the same way.

At first, researchers simply wrote *programs* to build the networks, using ordinary loops to handle repeated structure. Figure 3 shows pseudocode to build an alarm network for R geological fault regions, each with $H(r)$ houses.

The pictorial notation of *plates* was developed to denote repeated structure and software tools such as BUGS[10] and Microsoft's infer.net have facilitated a rapid expansion in applications of probabilistic methods. In all these tools, the model structure is built by a fixed program, so every possible world has the same random variables connected in the same way. Moreover, the code for constructing the models is not viewed as something that could be the output of a learning algorithm.

d Briefly, the problems are avoided by separating the generative models for object existence and object properties and relations and by allowing for unobserved objects.

An open-universe probability model (OUPM) defines a probability distribution over possible worlds that vary in the objects they contain and in the mapping from symbols to objects. Thus, OUPMs can handle data from sources that violate the closed-universe assumption.

Breese[4] proposed a more declarative approach reminiscent of Horn clauses. Other declarative languages included Poole's Independent Choice Logic, Sato's PRISM, Koller and Pfeffer's probabilistic relational models, and de Raedt's Bayesian Logic Programs. In all these cases, the head of each clause or *dependency statement* corresponds to a parameterized set of child random variables, with the parent variables being the corresponding ground instances of the literals in the body of the clause. For example, Equation (4) shows the dependency statements equivalent to the code fragment in Figure 3:

$$\begin{aligned} &Burglary(h) \sim Bernoulli(0.003) \\ &Earthquake(r) \sim Bernoulli(0.002) \\ &Alarm(h) \sim \\ &\quad CPT[\ldots](Earthquake \\ &\quad (Faultregion(\mathrm{h})), Burglary(\mathrm{h})) \end{aligned} \tag{4}$$

where *CPT* denotes a suitable conditional probability table indexed by the corresponding arguments. Here, h and r are *logical* variables ranging over houses and regions; they are implicitly universally quantified. *FaultRegion* is a function symbol connecting a house to its geological region. Together with a *relational skeleton* that enumerates the objects of each type and specifies the values of each function and relation, a set of dependency statements such as Equation (4) corresponds to an ordinary, albeit potentially very large, Bayes net. For example, if there are two houses in region A and three in region B, the corresponding Bayes net is the one in Figure 4.

Open-Universe Models

As noted earlier, closed-universe languages disallow uncertainty about what things are in the world; the existence and identity of all objects must be known in advance.

In contrast, an open-universe probability model (OUPM) defines a probability distribution over possible worlds that vary in the objects they contain and in the mapping from symbols to objects. Thus, OUPMs can handle data from sources (text, video, radar, intelligence reports, among others) that violate the closed-universe assumption. Given evidence, OUPMs *learn* about the objects the world contains.

Looking at Figure 1a, the first problem is how to ensure that the model specifies a

proper distribution over a heterogeneous, unbounded set of possible worlds. The key is to extend the generative view of Bayes nets from the propositional to the first-order, open-universe case:

- Bayes nets generate propositional worlds one event at a time; each event fixes the value of a variable.
- First-order, closed-universe models such as Equation (4) define generation steps for entire classes of events.
- First-order, open-universe models include generative steps that *add objects to the world* rather than just fixing their properties and relations.

Consider, for example, an open-universe version of the alarm model in Equation (4). If one suspects the existence of up to three geological fault regions, with equal probability, this can be expressed as a *number statement*:

$$\#\,Region \sim UniformInt\,(1, 3)\,. \quad (5)$$

For the sake of illustration, let us assume that the number of houses in a region r is drawn uniformly between 0 and 4:

$$\#\,House(FaultRegion = r) \sim UniformInt(0, 4)\,. \quad (6)$$

Here, *FaultRegion* is called an *origin function*, since it connects a house to the region from which it originates.

Together, the dependency statements (4) and the two number statements (5 and 6), along with the necessary type signatures, specify a complete distribution over all the possible worlds definable with this vocabulary. There are infinitely many such worlds, but, because the number statements are bounded, only finitely many—317,680,374 to be precise—have non-zero probability. Figure 5 shows an example of a particular world constructed from this model.

The BLOG language[21] provides a precise syntax, semantics, and inference capability for open-universe probability models composed of dependency and number statements. BLOG models can be arbitrarily complex, but they inherit the key declarative property of Bayes nets: *every well-formed BLOG model specifies a well-defined probability distribution over possible worlds.*

To make such a claim precise, one must define exactly what these worlds are and how the model assigns a probability to each. The definitions (given in full in Brian Milch's PhD thesis[20]) begin with the objects each world contains. In the standard semantics of typed first-order logic, objects are just numbered tokens with types. In BLOG, each object also has an *origin*, indicating how it was generated. (The reason for this slightly baroque construction will become clear shortly.) For number statements with no origin functions—for example, Equation (5)—the objects have an empty origin; for example, $\langle Region, , 2\rangle$ refers to the second region generated from that statement. For number statements with origin functions—for example, Equation (6)—each object records its origin; for example, $\langle House, \langle FaultRegion, \langle Region, , 2\rangle\rangle, 3\rangle$ is the third house in the second region.

The *number variables* of a BLOG model specify how many objects there are of each type with each possible origin; thus $\#House_{\langle FaultRegion,\langle Region, ,2\rangle\rangle}(\omega) = 4$ means that in world ω there are 4 houses in region 2. The *basic variables* determine the values of predicates and functions for all tuples of objects; thus, $Earthquake_{\langle Region, ,2\rangle}(\omega) = true$ means that in world ω there is an earthquake in region 2. A possible world is defined by the values of all the number variables and basic variables. A world may be generated from the model by sampling in topological order, for example, see Figure 5.

Figure 3. Illustrative pseudocode for building a version of the Bayes net in Figure 2 that handles *R* different regions with *H*(*r*) houses in each.

```
loop for r from 1 to R do
    add node Earthquake_r with no parents, prior 0.002
    loop for h from 1 to H(r) do
        add node Burglary_r,h with no parents, prior 0.003
        add node Alarm_r,h with parents Earthquake_r, Burglary_r,h
```

Figure 4. The Bayes net corresponding to Equation (4), given two houses in region *A* and three in *B*.

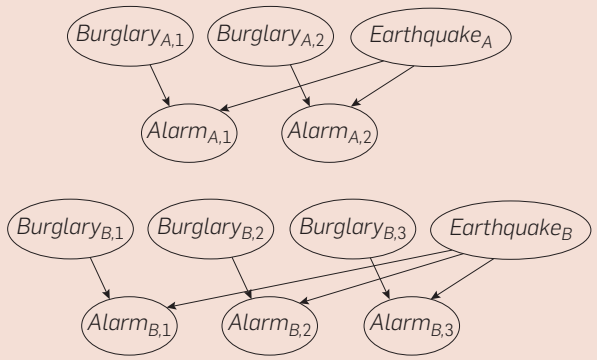


Figure 5. Construction of one possible world by sampling from the burglary/earthquake model. Each row shows the variable being sampled, the value it receives, and the probability of that value conditioned on the preceding assignments.

Variable	Value	Prob.
$\#Region$	2	0.3333
$Earthquake_{\langle Region, ,1\rangle}$	*false*	0.998
$Earthquake_{\langle Region, ,2\rangle}$	*false*	0.998
$\#House_{\langle FaultRegion,\langle Region, ,1\rangle\rangle}$	1	0.2
$\#House_{\langle FaultRegion,\langle Region, ,2\rangle\rangle}$	1	0.2
$Burglary_{\langle House,\langle FaultRegion,\langle Region, ,1\rangle\rangle,1\rangle}$	*false*	0.997
$Burglary_{\langle House,\langle FaultRegion,\langle Region, ,2\rangle\rangle,1\rangle}$	*true*	0.003

The probability of a world so constructed is the product of the probabilities for all the sampled values; in this case, 0.00003972063952. Now it becomes clear why each object contains its origin: this property ensures that every world can be constructed by exactly one sampling sequence. If this were not the case, the probability of a world would be the sum over all possible sampling sequences that create it.

Open-universe models may have infinitely many random variables, so the full theory involves nontrivial measure-theoretic considerations. For example, the number statement $\#Region \sim Poisson(\mu)$ assigns probability $e^{-\mu}\,\mu^k/k!$ to each nonnegative integer k. Moreover, the language allows recursion and infinite types (integers, strings, and so on). Finally, well-formedness disallows cyclic dependencies and infinitely receding ancestor chains; these conditions are undecidable in general, but certain syntactic sufficient conditions can be checked easily.

Examples

The standard "use case" for BLOG has three elements: the *model*, the *evidence* (the known facts in a given scenario), and the *query*, which may be any expression, possibly with free logical variables. The answer is a posterior joint probability for each possible set of substitutions for the free variables, given the evidence, according to the model.[e] Every model includes type declarations, type signatures for the predicates and functions, one or more number statements for each type, and one dependency statement for each predicate and function. (In the examples here, declarations and signatures are omitted where the meaning is clear.) Dependency statements use an if-then-else syntax to handle so-called *context-specific* dependencies, whereby one variable may or may not be a parent of another, depending on the value of a third variable.

e As with Prolog, there may be infinitely many sets of substitutions of unbounded size; designing exploratory interfaces for such answers is an interesting HCI challenge.

Figure 6. BLOG model for citation information extraction. For simplicity the model assumes one author per paper and omits details of the grammar and error models. OM(a, b) is a discrete log-normal, base 10, that is, the order of magnitude is $10^{\alpha\pm b}$.

```
type Researcher, Paper, Citation;
random String Name(Researcher);
random String Title(Paper);
random Paper PubCited(Citation);
random String Text(Citation);
random Boolean Prof(Researcher);
origin Researcher Author(Paper);
#Researcher ~ OM(3,1);
Name(r) ~ CensusDB_NamePrior();
Prof(r) ~ Boolean(0.2);
#Paper(Author=r)
   if Prof(r) then ~ OM(1.5,0.5) else ~ OM(1,0.5);
Title(p) ~ CSPaperDB_TitlePrior();
CitedPaper(c) ~ UniformChoice(Paper p);
Text(c) ~ HMMGrammar(Name(Author(CitedPaper(c))),
                     Title(CitedPaper(c)));
```

Figure 7. BLOG model for radar tracking of multiple targets. $X(a, t)$ is the state of aircraft a at time t, while $Z(b)$ is the observed position of blip b.

```
#Aircraft(EntryTime = t) ~ Poisson(λa);
Exits(a,t) if InFlight(a,t) then ~ Boolean(αe);
InFlight(a,t) =
   (t == EntryTime(a)) | (InFlight(a,t-1) & !Exits(a,t-1));
X(a,t) if t = EntryTime(a) then ~ InitState()
   else if InFlight(a,t) then ~ Normal( F*X(a,t-1),Σx);
#Blip(Source=a, Time=t)
  if InFlight(a,t) then ~ Bernoulli(DetectionProb(X(a,t)));
#Blip(Time=t) ~ Poisson(λf);
Z(b) if Source(b)=null then ~ UniformInRegion(R);
   else ~ Normal(H*X(Source(b),Time(b)),Σb);
```

Citation matching. Systems such as CiteSeer and Google Scholar extract a database-like representation, relating papers and researchers by authorship and citation links, from raw ASCII citation strings. These strings contain no object identifiers and include errors of syntax, spelling, punctuation, and content, which lead in turn to errors in the extracted databases. For example, in 2002, CiteSeer reported over 120 distinct books written by Russell and Norvig.

A generative model for this domain (Figure 6) connects an underlying, unobserved world to the observed strings: there are researchers, who have names; researchers write papers, which have titles; people cite the papers, combining the authors' names and the paper's title (with errors) into the text of the citation according to some grammar. Given citation strings as evidence, a multi-author version of this model, trained in an unsupervised fashion, had an error rate two to three times lower than CiteSeer's on four standard test sets.[25] The inference process in such a vertically integrated model also exhibits a form of collective, knowledge-driven disambiguation: the more citations there are for a given paper, the more accurately each of them is parsed, because the parses have to agree on the facts about the paper.

Multitarget tracking. Given a set of unlabeled data points generated by some unknown, time-varying set of objects, the goal is to detect and track the underlying objects. In radar systems, for example, each rotation of the radar dish produces a set of blips. New objects may appear, existing objects may disappear, and false alarms and detection failures are possible. The standard model (Figure 7) assumes independent, linear–Gaussian dynamics and measurements. Exact inference is provably intractable, but MCMC typically works well in practice. Perhaps more importantly, elaborations of the scenario (formation flying, objects heading for unknown destinations, objects taking off or landing) can be handled by small changes to the model without resorting to new mathematical derivations and complex programming.

Nuclear treaty monitoring. Verifying the Comprehensive Nuclear-Test-Ban Treaty requires finding all seismic events on Earth above a minimum magnitude. The UN CTBTO maintains a network of sensors, the International Monitoring System (IMS); its automated processing software, based on 100 years of seismology research, has a detection failure rate of about 30%. The NET-VISA system,[2] based on an OUPM, significantly reduces detection failures.

The NET-VISA model (Figure 8) expresses the relevant geophysics directly. It describes distributions over the number of events in a given time interval (most of which are naturally occurring) as well as over their time, magnitude, depth, and location. The locations of natural events are distributed according to a spatial prior trained (like other parts of the model) from historical data; man-made events are, by the treaty rules, assumed to occur uniformly. At every station s, each phase (seismic wave type) p from an event e produces either 0 or 1 detections (above-threshold signals); the detection probability depends on the event magnitude and depth and its distance from the station. ("False alarm" detections also occur according to a station-specific rate parameter.) The measured arrival time, amplitude, and other properties of a detection d depend on the properties of the originating event and its distance from the station.

Once trained, the model runs continuously. The evidence consists of detections (90% of which are false alarms) extracted from raw IMS waveform data and the query typically asks for the most likely event history, or *bulletin*, given the data. For reasons previously explained, NET-VISA uses a special-purpose inference algorithm. Results so far are encouraging: for example, in 2009 the UN's SEL3 automated bulletin missed 27.4% of the 27,294 events in the magnitude range 3–4 while NET-VISA missed 11.1%. Moreover, comparisons with dense regional networks show that NET-VISA finds up to 50% more real events than the final bulletins produced by the UN's expert seismic analysts. NET-VISA also tends to associate more detections with a given event, leading to more accurate location estimates (see Figure 9). The CTBTO has announced its intention to deploy NET-VISA as soon as possible.

Remarks on the examples. Despite superficial differences, the three examples are structurally similar: there are unknown objects (papers, aircraft, earthquakes) that generate percepts according to some physical process (citation, radar detection, seismic propagation). The same structure and reasoning patterns hold for areas such as database deduplication and natural language understanding. In some cases, inferring an object's existence involves grouping percepts together—a process that resembles the clustering task in machine learning. In other cases, an object may generate no percepts at all and still have its existence inferred—as happened, for example, when observations of Uranus led to the discovery of Neptune. Allowing object trajectories to be nonindependent in Figure 7 enables such inferences to take place.

Inference

By Equation (1), the probability $P(\alpha|\mathbf{e})$ for a closed query sentence α given evidence $\mathbf{e}$ is proportional to the sum of probabilities for all worlds in which α and $\mathbf{e}$ are satisfied, with the probability for each world being a product of model parameters as explained earlier.

Many algorithms exist to calculate or approximate this sum of products for Bayes nets. Hence, it is natural to consider *grounding* a first-order probability model by instantiating the

Figure 8. The NET-VISA model.

```
#SeismicEvents ~ Poisson(T*λe);
Natural(e) ~ Boolean(0.999);
Time(e) ~ UniformReal(0,T);
Magnitude(e) ~ Exponential(log(10));
Depth(e) if Natural(e) then ~ UniformReal(0,700) else = 0;
Location(e) if Natural(e) then ~ SpatialPrior()
   else ~ UniformEarthDistribution();
#Detections(event=e, phase=p, station=s)
  if IsDetected(e,p,s) then = 1 else = 0;
IsDetected(e,p,s) ~
   Logistic(weights(s,p), Magnitude(e), Depth(e), Distance(e,s));
#Detections(site = s) ~ Poisson(T*λf(s));
OnsetTime(d,s)
   if (event(d) = null) then ~ UniformReal(0,T)
   else = Time(event(d)) + Laplace(μt(s),σt(s)) +
     GeoTT(Distance(event(d),s),Depth(event(d)),phase(d));
Amplitude(d,s)
   if (event(d) = null) then ~ NoiseAmplitudeDistribution(s)
   else ~ AmplitudeModel(Magnitude(event(d)),
                Distance(event(d),s),Depth(event(d)),phase(d));
... and similar clauses for azimuth, slowness, and phase type.
```

Figure 9. Location estimates for the DPRK nuclear test of February 12, 2013: UN CTBTO Late Event Bulletin (green triangle); NET-VISA (blue square). The tunnel entrance (black cross) is 0.75 km from NET-VISA's estimate. Contours show NET-VISA's posterior location distribution.

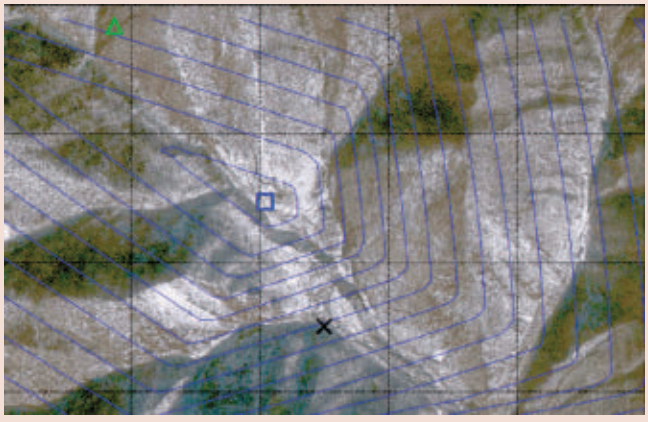


Figure 10. A CHURCH program that expresses the burglary/earthquake model in Equations (4)–(6).

```
(define num-regions (mem (lambda () (uniform 1 3))))
(define-record-type region (fields index))
(define regions (map (lambda (i) (make-region i))
                     (iota(num-regions)))
(define num-houses (mem (lambda (r)(uniform 0 4))))
(define-record-type house (fields fault-region index))
(define houses (map (lambda (r)
                      (map (lambda (i) (make-house r i))
                           (iota (num-houses r)))) regions)))
(define earthquake (mem (lambda (r) (flip 0.002))))
(define burglary (mem (lambda (h) (flip 0.003))))
(define alarm (mem (lambda (h)
  (if (burglary h)
      (if (earthquake (house-fault-region h))
          (flip 0.9) (flip 0.8))
      (if (earthquake (house-fault-region h))
          (flip 0.4) (flip 0.01))))
```

logical variables with "all possible" ground terms in order to generate a Bayes net, as illustrated in Figure 4; then, existing inference algorithms can be applied.

Because of the large size of these ground networks, exact inference is usually infeasible. The most general method for approximate inference is Markov chain Monte Carlo. MCMC methods execute a random walk among possible worlds, guided by the relative probabilities of the worlds visited and aggregating query results from each world. MCMC algorithms vary in the choice of neighborhood structure for the random walk and in the *proposal distribution* from which the next state is sampled; subject to an ergodicity condition on these choices, samples will converge to the true posterior in the limit. MCMC scales well with network size, but its *mixing time* (the time needed before samples reflect the true posterior) is sensitive to the quantitative structure of the conditional distributions; indeed, hard constraints may prevent convergence.

BUGS[10] and MLNs[30] apply MCMC to a preconstructed ground network, which requires a bound on the size of possible worlds and enforces a propositional "bit-vector" representation for them. An alternative approach is to apply MCMC directly on the space of first-order possible worlds[26, 31]; this provides much more freedom to use, for example, a sparse graph or even a relational database[19] to represent each world. Moreover, in this view it is easy to see that MCMC moves can not only alter relations and functions but also add or subtract objects and change the interpretations of constant symbols; thus, MCMC can move among the open-universe worlds shown in Figure 1a.

As noted, a typical BLOG model has infinitely many possible worlds, each of potentially infinite size. As an example, consider the multitarget tracking model in Figure 7: the function $X(a, t)$, denoting the state of aircraft a at time t, corresponds to an infinite sequence of variables for an unbounded number of aircraft at each step. Nonetheless, BLOG's MCMC inference algorithm converges to the correct answer, under the usual ergodicity conditions, for any well-formed BLOG model.[20]

The algorithm achieves this by sampling not completely specified possible worlds but *partial* worlds, each corresponding to a disjoint set of complete worlds. A partial world is a *minimal self-supporting instantiation*[f] of a subset of the *relevant* variables, that is, ancestors of the evidence and query variables. For example, variables $X(a, t)$ for values of t greater than the last observation time (or the query time, whichever is greater) are irrelevant so the algorithm can consider just a finite prefix of the infinite sequence. Moreover, the algorithm eliminates isomorphisms under object renumberings by computing the required combinatorial ratios for MCMC transitions between partial worlds of different sizes.[22]

MCMC algorithms for first-order languages also benefit from *locality* of computation[27]: the probability ratio between neighboring worlds depends on a subgraph of constant size around the variables whose values are changed. Moreover, a logical query can be evaluated *incrementally* in each world visited, usually in constant time per world, rather than recomputing it from scratch.[19, 31]

Despite these optimizations, generic inference for BLOG and other first-order languages remains too slow. Most real-world applications require a special-purpose proposal distribution to reduce the mixing time. A number of avenues are being pursued to resolve this issue:

- *Compiler techniques* can generate inference code that is specific to the model, query, and/or evidence. Microsoft's infer.net uses such methods to handle millions of variables. Experiments with BLOG show speedups of over 100×.[18]
- *Special-purpose probabilistic hardware*—such as Analog Devices' GP5 chip—offers further constant-factor speedups.
- *Special-purpose samplers* jointly sample groups of variables that are tightly constrained. A library of such samplers may render most user programs efficiently solvable.
- *Static analysis* can transform programs for efficiency[5, 15] and identify exactly solvable submodels.[7]

Finally, the rapidly advancing techniques of *lifted inference*[29] aim to unify probability and logic at the inferential level, borrowing and generalizing ideas from logical theorem proving. Such methods, surveyed by Van den Broeck,[32] may avoid grounding and take advantage of symmetries by manipulating symbolic distributions over large sets of objects.

Learning

Generative languages such as BLOG and BUGS naturally support Bayesian parameter learning with no modification: parameters are defined as random variables with priors and ordinary inference yields posterior parameter distributions given the evidence. For example, in Figure 8 we could add a prior $\lambda_e \sim \text{Gamma}(3, 0.1)$ for the seismic rate parameter λ_e instead of fixing its value in advance; learning proceeds as data arrive, even in the unsupervised case where no ground-truth events are supplied. A "trained

f An instantiation of a set of variables is self-supporting if the parents of every variable in the set are also in the set.

model" with fixed parameter values can be obtained by, for example, choosing maximum *a posteriori* values. In this way many standard machine learning methods can be implemented using just a few lines of modeling code. Maximum-likelihood parameter estimation could be added by stating that certain parameters are learnable, omitting their priors, and interleaving maximization steps with MCMC inference steps to obtain a stochastic online EM algorithm.

Structure learning—generating new dependencies and new predicates and functions—is more difficult. The standard idea of trading off degree of fit and model complexity can be applied, and some model search methods from the field of inductive logic programming can be generalized to the probabilistic case, but as yet little is known about how to make such methods computationally feasible.

Probability and Programs

Probabilistic programming languages or PPLs[17] represent a distinct but closely related approach for defining expressive probability models. The basic idea is that a randomized algorithm, written in an ordinary programming language, can be viewed not as a program to be executed, but as a probability model: a distribution over the possible *execution traces* of the program, given inputs. Evidence is asserted by fixing any aspect of the trace and a query is any predicate on traces. For example, one can write a simple Java program that rolls three six-sided dice; fix the sum to be 13; and ask for the probability that the second die is even. The answer is a sum over probabilities of complete traces; the probability of a trace is the product of the probabilities of the random choices made in that trace.

The first significant PPL was Pfeffer's IBAL,[28] a functional language equipped with an effective inference engine. CHURCH,[11] a PPL built on Scheme, generated interest in the cognitive science community as a way to model complex forms of learning; it also led to interesting connections to computability theory[1] and programming language research. Figure 10 shows the burglary/earthquake example in CHURCH; notice that the PPL code builds a possible-world data structure explicitly. Inference in CHURCH uses MCMC, where each move resamples one of the stochastic primitives involved in producing the current trace.

Execution traces of a randomized program may vary in the new objects they generate; thus, PPLs have an open-universe flavor. One can view BLOG as a declarative, relational PPL, but there is a significant semantic difference: in BLOG, in any given possible world, every ground term has a single value; thus, expressions such as $f(1) = f(1)$ are true by definition. In a PPL, on the other hand, $f(1) = f(1)$ may be false if f is a stochastic function, because each instance of $f(1)$ corresponds to a distinct piece of the execution trace. Memoizing every stochastic function (via `mem` in Figure 10) restores the standard semantics.

Prospects

These are early days in the process of unifying logic and probability. Experience in developing models for a wide range of applications will uncover new modeling idioms and lead to new kinds of programming constructs. And of course, inference and learning remain the major bottlenecks.

Historically, AI has suffered from insularity and fragmentation. Until the 1990s, it remained isolated from fields such as statistics and operations research, while its subfields—especially vision and robotics—went their own separate ways. The primary cause was *mathematical incompatibility*: what could a statistician of the 1960s, well-versed in linear regression and mixtures of Gaussians, offer an AI researcher building a robot to do the grocery shopping? Bayes nets have begun to reconnect AI to statistics, vision, and language research; first-order probabilistic languages, which have both Bayes nets and first-order logic as special cases, will extend and broaden this process.

Acknowledgments

My former students Brian Milch, Hanna Pasula, Nimar Arora, Erik Sudderth, Bhaskara Marthi, David Sontag, Daniel Ong, and Andrey Kolobov contributed to this research, as did NSF, DARPA, the Chaire Blaise Pascal, and ANR.

References

1. Ackerman, N., Freer, C., Roy, D. On the computability of conditional probability. arXiv 1005.3014, 2013.
2. Arora, N.S., Russell, S., Sudderth, E. NET-VISA: Network processing vertically integrated seismic analysis. *Bull. Seism. Soc. Am. 103* (2013).
3. Bacchus, F. *Representing and Reasoning with Probabilistic Knowledge*. MIT Press, 1990.
4. Breese, J.S. Construction of belief and decision networks. *Comput. Intell. 8* (1992) 624–647.
5. Claret, G., Rajamani, S.K., Nori, A.V., Gordon, A.D., Borgström, J. Bayesian inference using data flow analysis. In *FSE-13* (2013).
6. Dalvi, N.N., Ré, C., Suciu, D. Probabilistic databases. *CACM 52*, 7 (2009), 86–94.
7. Fischer, B., Schumann, J. AutoBayes: A system for generating data analysis programs from statistical models. *J. Funct. Program 13* (2003).
8. Gaifman, H. Concerning measures in first order calculi. *Israel J. Math. 2* (1964), 1–18.
9. Gaifman, H. Concerning measures on Boolean algebras. *Pacific J. Math. 14* (1964), 61–73.
10. Gilks, W.R., Thomas, A., Spiegelhalter, D.J. A language and program for complex Bayesian modelling. *The Statistician 43* (1994), 169–178.
11. Goodman, N.D., Mansinghka, V.K., Roy, D., Bonawitz, K., Tenenbaum, J.B. Church: A language for generative models. In *UAI-08* (2008).
12. Hailperin, T. Probability logic. *Notre Dame J. Formal Logic 25*, 3 (1984), 198–212.
13. Halpern, J.Y. An analysis of first-order logics of probability. *AIJ 46*, 3 (1990), 311–350.
14. Howson, C. Probability and logic. *J. Appl. Logic 1*, 3–4 (2003), 151–165.
15. Hur, C.-K., Nori, A.V., Rajamani, S.K., Samuel, S. Slicing probabilistic programs. In *PLDI-14* (2014).
16. Jain, D., Kirchlechner, B., Beetz, M. Extending Markov logic to model probability distributions in relational domains. In *KI-07* (2007).
17. Koller, D., McAllester, D.A., Pfeffer, A. Effective Bayesian inference for stochastic programs. In *AAAI-97* (1997).
18. Li, L., Wu, Y., Russell, S. SWIFT: Compiled inference for probabilistic programs. Tech. Report EECS-2015-12, UC Berkeley, 2015.
19. McCallum, A., Schultz, K., Singh, S. FACTORIE: Probabilistic programming via imperatively defined factor graphs. In *NIPS 22* (2010).
20. Milch, B. Probabilistic models with unknown objects. PhD thesis, UC Berkeley, 2006.
21. Milch, B., Marthi, B., Sontag, D., Russell, S.J., Ong, D., Kolobov, A. BLOG: Probabilistic models with unknown objects. In *IJCAI-05* (2005).
22. Milch, B., Russell, S.J. General-purpose MCMC inference over relational structures. In *UAI-06* (2006).
23. Nilsson, N.J. Probabilistic logic. *AIJ 28* (1986), 71–87.
24. Paskin, M. Maximum entropy probabilistic logic. Tech. Report UCB/CSD-01-1161, UC Berkeley, 2002.
25. Pasula, H., Marthi, B., Milch, B., Russell, S.J., Shpitser, I. Identity uncertainty and citation matching. In *NIPS 15* (2003).
26. Pasula, H., Russell, S.J. Approximate inference for first-order probabilistic languages. In *IJCAI-01* (2001).
27. Pearl, J. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
28. Pfeffer, A. IBAL: A probabilistic rational programming language. In *IJCAI-01* (2001).
29. Poole, D. First-order probabilistic inference. In *IJCAI-03* (2003).
30. Richardson, M., Domingos, P. Markov logic networks. *Machine Learning 62*, 1–2 (2006), 107–136.
31. Russell, S.J. Expressive probability models in science. In *Discovery Science* (Tokyo, 1999).
32. Van den Broeck, G. *Lifted Inference and Learning in Statistical Relational Models*. PhD thesis, Katholieke Universiteit Leuven, 2013.

Stuart Russell (russell@cs.berkeley.edu), is a professor of computer science and Smith-Zadeh Professor of Engineering at the University of California, Berkeley.

Watch the author discuss their work in this exclusive *Communications* video. http://cacm.acm.org/videos/unifying-logic-and-probability

DOI:10.1145/2776827

To view the accompanying paper, visit doi.acm.org/10.1145/ 2776825 rh

Technical Perspective
The Simplicity of Cache Efficient Functional Algorithms

By William D. Clinger

SCIENTIFIC MODELS ARE simplified descriptions of messy reality.

Depending on our purposes, some models may be better than others. Good models are abstract enough to be tractable, yet accurate enough to draw conclusions matching the aspects of reality we hope to understand.

Even good models have limits, so we must take care to use them only within their domain of applicability. Galileo's law for adding velocities is simple and accurate enough for most purposes, but a more complicated law is needed when adding relativistic velocities. Einstein's special theory of relativity tells us how to reason about space and time in the absence of acceleration or gravity, but we must turn to Einstein's general theory when gravitational effects are significant. That general theory has passed all experimental tests so far, but we know its equations break down at black holes and other singularities.

In general, scientific models seek compromise between tractability and accuracy. That compromise is evident within the cost models we use to analyze algorithms. In the following paper, Blelloch and Harper propose and demonstrate a more effective model for analyzing and describing the efficiency of functional algorithms.

Fifty years ago, modeling a computer system's main storage as random-access memory (RAM) was both simple and accurate. That RAM cost model remains simple, but it is no longer accurate.

Cache-aware estimates of real-world performance can be more accurate but are also more complicated. Even the *ideal cache model*, which assumes an unachievable perfect replacement policy, must account for both spatial and temporal locality.

Sometimes, however, we can improve both tractability *and* accuracy, as when we use asymptotic notation to describe costs. It is easier to come up with an asymptotic estimate for the number of steps in a computation than it is to count the exact number of steps. Counting the exact number of steps is more precise, but exact counts will be inaccurate models of the real world because compiler optimizations and hardware peculiarities break the estimate. Asymptotic estimates are more robustly accurate *because* they are less precise: They express costs at a higher (and more useful) level of abstraction.

Cache-oblivious algorithms provide another example of using abstraction to combine accurate cost estimates with tractability. The cache size M and cache line (block) size B are abstracted parameters that may show up in the asymptotic complexity of a cache-oblivious algorithm, but do not appear within the algorithm itself.

To prove that an algorithm is cache-oblivious, we must consider its spatial and temporal locality. Heretofore, most of those proofs have dealt with imperative algorithms over arrays. To extend the technique to functional algorithms, we need to make assumptions about the locality of object allocation and garbage collection.

In the following paper, Blelloch and Harper propose and demonstrate a more effective model for analyzing and describing the efficiency of functional algorithms.

Blelloch and Harper suggest we analyze the costs of functional algorithms by assuming objects are allocated sequentially in cache memory, with each new object adjacent to the previously allocated object, and garbage collection preserves this correspondence between allocation order and memory order. Their key abstraction defines a *compact* data structure as one for which there exists a fixed constant k, independent of the size of the structure, such that the data structure can be traversed with at most k cache misses per object. Using that abstraction and those two assumptions, the authors show how efficient cache-oblivious functional algorithms over lists and trees can be expressed and analyzed as easily as imperative algorithms over arrays.

Not all storage allocators and garbage collectors satisfy the critical assumptions of this paper, but some do. In time, as cache-oblivious functional algorithms become more common, we can expect even more implementations to satisfy those assumptions (or to come close enough).

After all, we computer scientists have a big advantage over the physicists: We can improve both the simplicity and the accuracy of our models by improving the reality we are modeling. C

William D. Clinger (will@ccs.neu.edu) is an associate professor in the College of Computer and Information Science at Northeastern University, Boston, MA.

DOI:10.1145/ 2776825

Cache Efficient Functional Algorithms

By Guy E. Blelloch and Robert Harper

Abstract

The widely studied I/O and ideal-cache models were developed to account for the large difference in costs to access memory at different levels of the memory hierarchy. Both models are based on a two level memory hierarchy with a fixed size fast memory (cache) of size *M*, and an unbounded slow memory organized in blocks of size *B*. The cost measure is based purely on the number of block transfers between the primary and secondary memory. All other operations are free. Many algorithms have been analyzed in these models and indeed these models predict the relative performance of algorithms much more accurately than the standard Random Access Machine (RAM) model. The models, however, require specifying algorithms at a very low level, requiring the user to carefully lay out their data in arrays in memory and manage their own memory allocation.

We present a cost model for analyzing the memory efficiency of algorithms expressed in a simple functional language. We show how some algorithms written in standard forms using just lists and trees (no arrays) and requiring no explicit memory layout or memory management are efficient in the model. We then describe an implementation of the language and show provable bounds for mapping the cost in our model to the cost in the ideal-cache model. These bounds imply that purely functional programs based on lists and trees with no special attention to any details of memory layout can be asymptotically as efficient as the carefully designed imperative I/O efficient algorithms. For example we describe an $O\left(\frac{n}{B}\log_{M/B}\frac{n}{B}\right)$ cost sorting algorithm, which is optimal in the ideal cache and I/O models.

1. INTRODUCTION

Today's computers exhibit a vast difference in cost for accessing different levels of the memory hierarchy, whether it be registers, one of many levels of cache, the main memory, or a disk. On current processors, for example, there is over a factor of a hundred between the time to access a register and main memory, and another factor of a hundred or so between main memory and disk, even a solid state drive (SSD). This variance in costs is contrary to the standard Random Access Machine (RAM) model, which assumes that the cost of accessing memory is uniform. To account for non-uniformity, several cost models have been developed that assign different costs to different levels of the memory hierarchy.

Figure 1 describes the widely used I/O[2] and ideal-cache[11] machine models designed for this purpose. The models are based on two parameters, the memory size M and the block size B, which are considered variables for the sake of analysis and therefore show up in asymptotic bounds. To design efficient algorithms for these models, it is important to consider both temporal locality, so as to take advantage of the limited size slow memory, and spatial locality, because memory is transferred in contiguous blocks. In this paper, we will use terminology from the ideal-cache machine model (**ICMM**), including referring to the fast memory as **cache**, the slow memory as **main memory**, block transfers as **cache misses**, and the cost as the **cache cost**.

Algorithms that do well in these models are often referred to as cache or I/O efficient. The theory of cache efficient algorithms is now well developed (see, for example, the surveys[3, 6, 12, 17, 19, 23]). These models do indeed express more accurately the cost of algorithms on real machines than does the standard RAM model. For example, the models properly indicate that a blocked or hierarchical matrix–matrix multiply has cost

$$\text{matrix multiply cache cost} = \Theta\left(\frac{n^3}{B\sqrt{M}}\right). \qquad (1)$$

This is much more efficient than the naïve triply nested loop, which has cost $\Theta(n^3/B)$. Furthermore, although the models only consider two levels of a memory hierarchy, cache efficient algorithms in the **ICMM** that do not use the parameters B or M in the code are simultaneously efficient across all levels of the memory hierarchy. Algorithms designed in this way are called *cache oblivious*.[11]

As an example of an algorithm analyzed in the models consider recursive mergeSort (see Figure 2). If the inputs of size n and output are in arrays, then the `merge` can take advantage of spatial locality and has cache cost $O(n/B)$ (see Figure 3). The `split` can be done within the same, or better bounds. For the recursive mergeSort once the recursion reaches a level where the computation fits into the cache, all subcalls are free, taking advantage of temporal locality. Figure 4 illustrates the merge sort recursion tree and analyzes the total cost, which for an input of size n is:

$$\text{merge sort cache cost} = \Theta\left(\left(\frac{n}{B}\right)\log_2\left(\frac{n}{M}\right)\right) \qquad (2)$$

The original version of this paper is entitled "Cache and I/O Efficient Functional Algorithms" and was published in the *Proceedings of the 40th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '13)*, 39–50.

Figure 1. The *I/O model*[2] assumes a memory hierarchy comprising a fast memory of size *M* and slow memory of unbounded size. Both memories are partitioned into blocks of a fixed size *B* of consecutive memory locations. The CPU and fast memory are treated as a standard Random Access Machine (RAM)—the CPU accesses individual words, not blocks. Additional instructions allow one to move any block between the fast and slow memory. The cost of an algorithm is analyzed in terms of the number of block transfers—the cost of operations within the main memory is ignored. The *ideal-cache machine model* (ICMM)[11] is similar except that the program only addresses the slow memory, and an "ideal" cache decides which blocks to cache in the fast memory using an optimal replacement strategy. Accessing the fast memory (cache) is again regarded as cost-free and transferring between slow memory and fast memory has unit cost. The costs in the two models are equivalent within a constant factor, and in both models the two levels of memory can either represent main memory and disk, or cache and main memory.

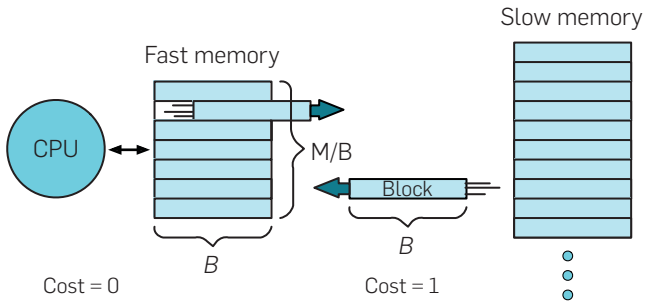


Figure 2. Recursive `mergeSort`. If the input sequence is empty or a singleton the algorithm returns the same value, otherwise it splits the input in two approximately equal sized sequences, `L` and `H`, recursively sorts each sequence, and merges the result.

```
1  fun mergeSort([]) = []
2    | mergeSort([a]) = [a]
3    | mergeSort(A) =
4  let
5    val (L, H) = split(A)
6  in
7    merge(mergeSort(L), mergeSort(H))
8  end
```

Figure 3. Merging two sorted arrays. The fingers are moved from left to right along the two inputs and the output. At each point the lesser value at the input fingers is copied to the output, and that finger, along with the output finger, are incremented. During the merge, each block of the inputs only has to be loaded into the fast memory once, and each block of the output only needs to be written to slow memory once. Therefore, for two arrays of size *n* the total cache cost of the merge is *O*(*n*/*B*).

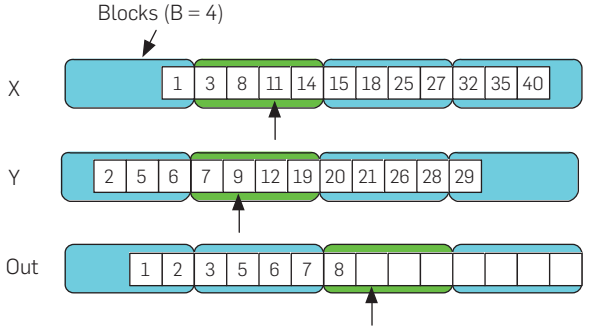
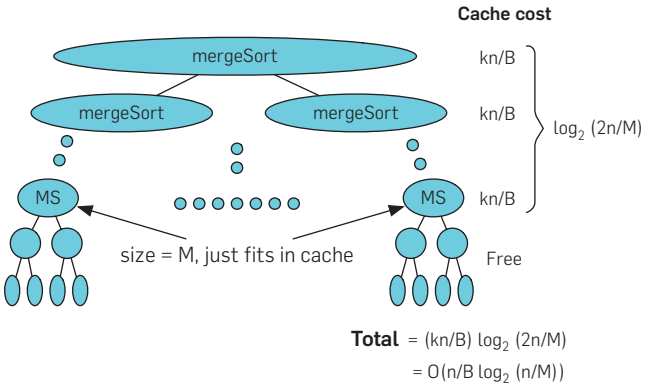


Figure 4. The cost of mergeSort in the I/O or ideal-cache model. At the *i*th level from the top there are 2^i calls to mergeSort, each of size $n_i = n/2^i$. Each call has cost kn_i/B for some constant *k*. The total cost of each level is therefore about $2^i k\left(\frac{n}{2^i}\right)/B = kn/B$. When a call fits into the fast memory, then the rest of the sort is cost-free. Because a merge requires about 2*n* space (for the input and the output), this will occur when $M \approx 2\left(\frac{n}{2^i}\right)$. Therefore, there are $\log_2(2n/M)$ levels that have non-zero cost, each with cost *kn*/*B*.



This is a reasonable cost and much better than assuming every memory access requires a cache miss. However, this is not optimal for the model. Later, we will outline a multiway merge sort that is optimal. It has cost

$$\text{optimal sort cache cost} = \Theta\left(\left(\frac{n}{B}\right)\log_{\frac{M}{B}}\left(\frac{n}{M}\right)\right) \qquad (3)$$

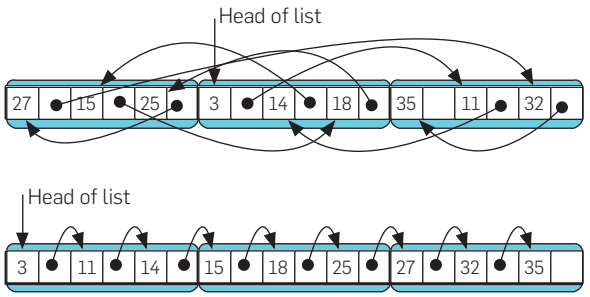
This optimal cost can be significantly less than the cost for mergeSort. For example if $n \le M^k$ and $M \ge B^2$, which are reasonable assumptions in many situations, then the cost reduces to $O(nk/B)$. All the fastest disk sorts indeed use some variant of a multiway merge sort or another optimal cache efficient algorithm based on sample sorting.[21]

1.1. Careful layout and careful allocation

Although the analysis for mergeSort given above seems relatively straightforward, we have picked a simple example and left out several important details. With regards to memory layout, ordering a sorted sequence contiguously in memory is pretty straightforward, but this is not the case for many other data structures. Should a matrix, for example, be laid out in row-major order, in column-major order, or perhaps in some hierarchical order such as z-ordering? These different orders can make a big difference. What about pointer-based structures such as linked-lists or trees? As Figure 5 indicates, the cost of traversing a linked list will depend on how the lists are laid out in memory.

More subtle than memory layout is memory allocation. Touching unused memory after allocation will cause a cache miss. Managing the pool of free memory properly is therefore very important. Consider, again, the mergeSort example. In our discussion, we assumed that once $2n_i \le M$ the problem fits in fast memory and therefore is free. However, if we used a memory allocator to allocate the temporary array needed for the merge the locations would likely be

Figure 5. Two lists in memory. The cost of traversing the lists is very different depending on how the lists are laid out. In the figure, the top one, which is randomly laid out, will require $O(n)$ steps to traverse, whereas the bottom only requires $O(n/B)$.



fresh and accessing them would cause a cache miss. The cost would therefore not be free, and in fact because levels near the leaves of the recursion tree do many small allocations, the cost could be significantly larger near the leaves. To avoid this, programmers need to manage their own memory. They could, for example, preallocate a temporary array and then pass it down to the recursive calls as an extra argument, therefore making sure that all temporary space is being reused. For more involved algorithms, allocation can be much more complicated.

In summary, designing and programming cache efficient algorithms for these models requires a careful layout of data in the flat memory and careful management of space in that memory.

1.2. Our goals

The goal of our work is to be able to take advantage of the ideal-cache model, and hence machines that match the model, without requiring careful layout of data in memory and without requiring managing memory allocation by hand. In particular, we want to work with recursive data structures (pointer structures) such as lists or trees. Furthermore, we want to work with a fully automated memory manager with garbage collection and no (explicit) specifications of where data is placed in memory. These goals might seem impossible, but we show that they can be achieved, at least for certain cases. We show this in the context of a purely functional (side-effect free) language. The formulation given here is slightly simplified compared to the associated conference paper[5] for the sake of brevity and clarity; we refer the reader to that paper for a fuller explanation.

As an example of what we are trying to achieve consider the `merge` algorithm shown in Figure 6. Some properties to notice about the algorithm are that it is based on linked-lists instead of arrays. Furthermore, because we are working in the functional setting, every list element is created fresh (in the `Cons` operations in Lines 10 and 11). Finally, there is no specification of where things are allocated and no explicit memory management. Given these properties it might seem that this code could be terrible for cache efficiency because accessing each element could be very expensive, as indicated in Figure 5. Furthermore, each fresh allocation could cause

Figure 6. Code for `merge` of two lists. If list is empty, the code compares the two heads of the list and recurs on the tail of the list with the smaller element, and on the whole list with the larger element. When the recursive call finishes a new list element is created with the smaller element as the head and the result of the recursive call as the tail. The `!` is discussed in Section 3.

```
1  datatype  List = Cons of (int * List)
2                 | Nil
3
4  fun merge(A, B) =
5    case (A, B) of
6       (Nil, B) ⇒ B
7     | (A, Nil) ⇒ A
8     | (Cons(a, At), Cons(b, Bt)) ⇒
9          if (a < b)
10         then Cons(!a, merge(At, B))
11         else Cons(!b, merge(A, Bt))
```

a cache miss. Also there is an implied stack in the recursion and it is not clear how this affects the cost.

We show, however, that with an appropriate implementation the code is indeed cache efficient, and when used in a mergeSort gives the same bound as given in Equation (2). We also describe an algorithm that matches the optimal sorting bounds given in Equation (3). The approach is not limited to lists, it also works with trees. For example, Figure 11 defines a representation of matrices recursively divided into a tree and a matrix multiplication based on it. The cost bounds for this multiplication match the bounds for matrix multiplication for the ideal-cache model (Equation 1) using arrays and careful programmer layout.

1.3. Cost models and provable implementation bounds

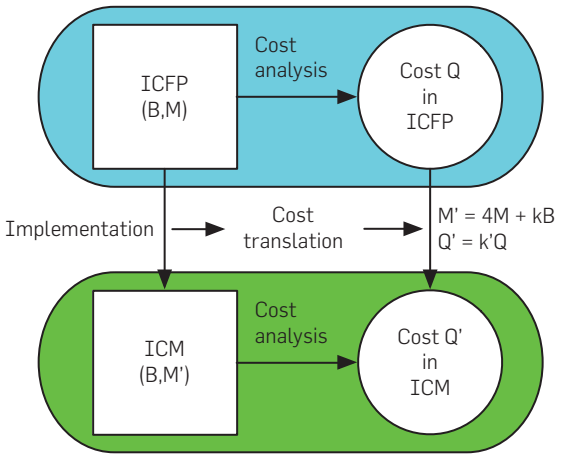
One way to achieve our goals would be to analyze each algorithm written in high-level code with respect to a particular compilation strategy and specific runtime memory allocation scheme. This would be extremely tedious, time consuming, not portable, and error prone. It would not lead to a practical way to analyze algorithms.

Instead we use a cost semantics that directly defines costs in the programming language so that algorithms can be analyzed without concern of implementation details (such as how the garbage collector works). We refer to our language, equipped with our cost semantics, as **ICFP**, for *Ideal-Cache Functional Programming* language. The goal is then to relate the costs in **ICFP** to costs in the **ICMM**. We do this by describing a simulation of **ICFP** on the **ICMM** and proving bounds on the relative costs based on this simulation. In particular, we show that a computation running with cost Q, with parameters M and B, in **ICFP** can be simulated on the **ICMM** with $M' = O(M)$ locations of fast memory, blocksize B, and cost $O(Q)$. The framework is illustrated in Figure 7.

1.4. Allocation order and spatial locality

Because functional languages abstract away from data layout decisions, we use temporal locality (proximity of allocation time) as a proxy for spatial locality (proximity of allocation

Figure 7. Cost semantics for ICFP and its mapping to the ideal cache model (ICMM). The values *k* and *k'* are constants.



in memory). To do this, the memory model for **ICFP** consists of two fast memories: a nursery (for new allocations) and a read cache. The nursery is time ordered to ensure that items are laid out contiguously when written to main (slow) memory. For merging, for example, the output list elements will be placed one after the other in main memory. To formalize this idea we introduce the notion of a data structure being compact. Roughly speaking a data structure of size n is compact if it can be traversed in the model in $O(n/B)$ cache cost. The output of merge is compact by construction. Furthermore, if both inputs lists to merge are compact, we show that merge has cache cost $O(n/B)$. This notion of compactness generalizes to other data structures, such as trees.

Another feature of the allocation cache of **ICFP** is that it only contains live data. This is important because in some algorithms, such as matrix multiply, the rate at which temporary memory is generated is asymptotically larger than the rate at which output data is generated. Therefore, if all counted it could fill up the cache, whereas we want to make sure that temporary data slots are reused. Also, we want to ensure that temporary memory is not transferred to the slow memory because such transfer would cause extra traffic. Moreover, the temporary data could end up between output data elements, causing them to be fragmented across blocks. In our provable implementation we do not require a precise tracking of live data, but instead use a generational collector with a nursery of size $2M$. The simulation then keeps all the $2M$ locations in the fast memory so that all allocation and collection (within the new generation) is fast.

1.5. Related work

The general idea of using high-level cost models based on a cost semantics along with a provable efficient implementation has previously been used in the context of parallel cost models.[4, 13, 15, 22] Although there has been a large amount of experimental work on showing how good garbage collection can lead to efficient use of caches and disks (Chilimbi and Larus[7], Courts[10], Grunwald et al.[14], Wilson et al.[24] and many references in Jones and Lins[16]), we know of none that try to prove bounds for algorithms for functional programs when manipulating recursive data types such as lists or trees. Abello et al.[1] show how a functional style can be used to design cache efficient graph algorithms. However, they assume that data structures are in arrays (called lists), and that primitives for operations such as sorting, map, filter and reductions are supplied and implemented with optimal asymptotic cost (presumably at a lower level using imperative code). Their goal is therefore to design graph algorithms by composing these high-level operations on collections. They do not explain how to deal with garbage collection or memory management.

2. COST SEMANTICS

In general, a *cost semantics* for a language defines both *how to execute* a program and the *cost* of its execution. In functional languages execution consists of a deterministic strategy for calculating the value of an expression by a process of simplification, or *reduction*, similar to what one learns in elementary algebra. But rather than working with the real numbers, programs in a functional language calculate with integers, with inductive data structures, such as lists and trees, and with functions that act on such values, including other functions. The theoretical foundation for functional languages is Church's λ-calculus.[8, 9]

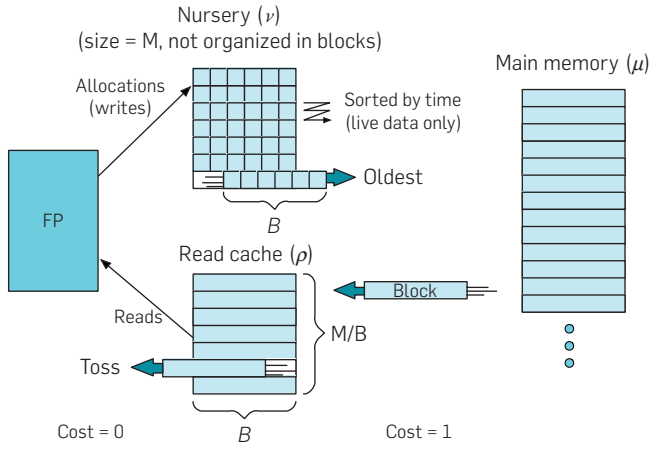
The cost of a computation in a functional language may be defined in various ways, including familiar measures such as time complexity, defined as the number of primitive reduction steps required to evaluate an expression, and space complexity, defined as the number of basic data objects allocated during evaluation. Such measures are defined in terms of the constructs of the language itself, rather than in terms of its implementation on a machine model such as a RAM or Turing machine. Algorithm analysis takes place entirely at the language level. The role of the implementation is to ensure that the abstract costs can be realized on a concrete machine with stated bounds.

2.1. Cache cost

The cost measure of interest in the present paper is the *cache cost*, which is derived from considering a combination of the time and space behavior of a program. The cache cost is derived by specifying when data objects are allocated and retrieved, and as with the **ICMM** is parameterized by two constants, B and M, governing the cache behavior. To be more precise, expressions are evaluated relative to an abstract *store*, σ, which comprises a *main memory*, μ, a *read cache*, ρ, and a *nursery*, or *allocation area*, υ. The structure of the store is depicted in Figure 8. The *main memory* maps abstract locations to atomic data objects, and is of unbounded capacity. Atomic data objects are considered to occupy one unit of storage. Objects in main memory are aggregated into *blocks* of size B in a manner to be described shortly. The blocking of data objects does not change during execution; once an object is assigned to a block, it remains part of that block for the rest of the computation.

The *read cache* is a fixed-size mapping from locations to data objects containing at most M objects. As usual, the read cache represents a partial view of main memory. Objects are

Figure 8. The memory model for the ICFP cost semantics. It is similar to the ideal cache model, but has two fast memories of size *M*: a *nursery* for newly allocated data, and a *read cache* for data that has migrated to main memory and was read again by the program. The main memory and read cache are organized into blocks, as in the ideal-cache model, but the nursery is not blocked. Rather, it consists of (live) data objects linearly ordered by the time at which they were allocated. The semantics defines when a piece of data is live, but conceptually it is just when it is reachable from the executing program. If an allocation overflows the nursery, the oldest *B* live objects are flushed from the cache into main memory as a block to make space for the new object. If one of these words is later read, the entire block is moved into the read cache, possibly displacing another block in the process. Displaced blocks can be discarded, because the only writes in a functional language occur at allocation, and never during subsequent execution.



loaded into the read cache in blocks of size B as determined by the aggregation of objects in main memory. Objects are evicted from the read cache by simply over-writing them; in a functional language data objects cannot be mutated, and hence need never be written back to main memory. Blocks are evicted according to the (uncomputable) *Ideal Cache Model* (**ICMM**).

The *allocation area*, or *nursery*, is fixed-sized mapping from locations to data objects also containing at most M objects. Locations in the nursery are not blocked, but they are linearly ordered according to the time at which they are allocated. We say that one data object is *older* than another if the location of the one is ordered prior to that of the other. A location in the nursery is *live* if it occurs within the program being evaluated.[18] All data objects are allocated in the nursery. When its capacity of M objects is exceeded, the oldest B objects are formed into a block that is migrated to main memory, determining once and for all the block to which these objects belong. This policy ensures that *temporal locality implies spatial locality*, which means that objects that are allocated nearby in time will be situated nearby in space (i.e., occupy the same block). In particular, when an object is loaded into the read cache, its neighbors in its block will be loaded along with it. These will be the objects that were allocated at around the same time.

The role of a cost semantics is to enable reasoning about the cache cost of algorithms without dropping down to the implementation level. To support proof, the semantics must be specified in a mathematically rigorous manner, which we now illustrate for **ICFP**, a simple eager variant of Plotkin's language PCF[20] for higher-order computation over natural numbers. (It is straightforward to extend **ICFP** to account for a richer variety of data structures, such as lists or trees, and for hardware-oriented concepts such as words and floating point numbers, which we assume in our examples.)

2.2. Cost semantics

The abstract syntax of **ICFP** is defined as follows:

$$e \;::=\; x \mid \mathtt{z} \mid \mathtt{s}(e) \mid \mathtt{ifz}\,(e;e_0;x.e_1) \mid \mathtt{fun}\,(x,y.e) \mid \mathtt{app}\,(e_1;e_2)$$

Here x stands for a variable, which will always stand for a data object allocated in memory. The expressions z and s( ) represent 0 and successor, respectively, and the conditional tests whether e is z or not, branching to e_0 if so, and branching to e_1 if not, substituting (the location of) the predecessor for x in doing so. Functions, written fun(x, $y.e$), bind a variable x standing for the function itself, and a variable y standing for its argument. (The variable x is used to effect recursive calls.) Function application is written app(e_1; e_2), which applies the function given by the expression e_1 to the argument value given by the expression e_2. The typing rules for **ICFP** are standard (see Chapter 10 of Harper[15]), and hence are omitted for the sake of concision.

Were the cost of a computation in **ICFP** just the time complexity, the semantics would be defined by an evaluation relation $e \Downarrow^n v$ stating that the expression e evaluates to the value v using n reduction steps according to a specified outermost strategy. Accounting for the cache cost of a computation is a bit more complicated, because we must account for the memory traffic induced by execution. To do so, we consider an evaluation relation of the form

$$\sigma \mathbin{@} e \Downarrow^n \sigma' \mathbin{@} l$$

in which the evaluation of an expression e is performed relative to a store σ, and returns a value represented by a location in a modified store σ'. The modifications to the store reflect the allocation of new objects, their migration to main memory, and their loading into the read cache. The evaluation relation also specifies the cache cost n, which is determined entirely by the movements of blocks to and from main memory. All other operations are assigned zero cost.

The evaluation relation for **ICFP** is defined by *derivation rules* that specify closure conditions on it. Evaluation is defined to be the strongest relation satisfying these conditions. To give a flavor of how this is done, we present in Figure 9 a simplified form of the rule for evaluating a function application, app(e_1; e_2). The rule has four premises, and one conclusion, separated by the horizontal line, which may be read as implication stating that if the premises are all derivable, then so is the conclusion.

The other constructs of **ICFP** are defined by similar rules, which altogether specify the behavior and cost of the evaluation of any **ICFP** program. The abstract cost

Figure 9. Simplified semantics of function application. This rule specifies the cost and evaluation of a function application $\texttt{app}(e_1; e_2)$ of a function e_1 to an argument e_2. similar rules govern the other language constructs. Bear in mind that all values are represented by (abstract) locations in memory. First, the expression e_1 is evaluated, with cost n_1, to obtain the location l_1 of the function being called. That location is read from memory to obtain the function itself, a self-referential λ-abstraction, at cost n_1'. Second, the expression e_2 is evaluated, with cost n_2, to obtain its value l_2. Finally, l_1 and l_2 are substituted into the body of the function, and then evaluated to obtain the result l at cost n. The overall result is l with total cost the sum of the constituent costs. The only non-zero cost arises from allocation of objects and reading from memory; all other operations are considered cost-free, as in the I/O model.

$$\left\{\begin{array}{rl} \sigma \mathbin{@} e_1 \Downarrow^{n_1} \sigma_1' \mathbin{@} l_1' & \text{(evaluate function)} \\ \sigma_1' \mathbin{@} l_1' \downarrow^{n_1'} \sigma_2 \mathbin{@} \texttt{fun}(x, y.e) & \text{(load function value)} \\ \sigma_2 \mathbin{@} e_2 \Downarrow^{n_2} \sigma_2' \mathbin{@} l_2' & \text{(evaluate argument)} \\ \sigma_2' \mathbin{@} [l_1', l_2'/x, y]e \Downarrow^{n} \sigma' \mathbin{@} l & \text{(evaluate function body)} \end{array}\right\}$$
$$\sigma \mathbin{@} \texttt{app}(e_1; e_2) \Downarrow^{n_1 + n_1' + n_2 + n_2'} \sigma' \mathbin{@} l'$$

semantics underlies the analyses given in Sections 1 and 3. The abstract costs given by the semantics are validated by giving a *provable implementation*[4, 13] that realizes these costs on a concrete machine. In our case the realization is given in terms of the **ICMM**, which is an accepted concrete formulation of a machine with hierarchical memory. By composing the abstract semantics with the provable implementation we obtain end-to-end bounds on the cache complexity of algorithms written in **ICFP** without having to perform the analysis at the level of the concrete machine, but rather at the level of the language in which the algorithms are written.

2.4. Provable implementation

The provable implementation of **ICFP** on the **ICMM** is described in Figure 10. Its main properties, which are important for obtaining end-to-end bounds on cache complexity, are summarized by the following theorem:

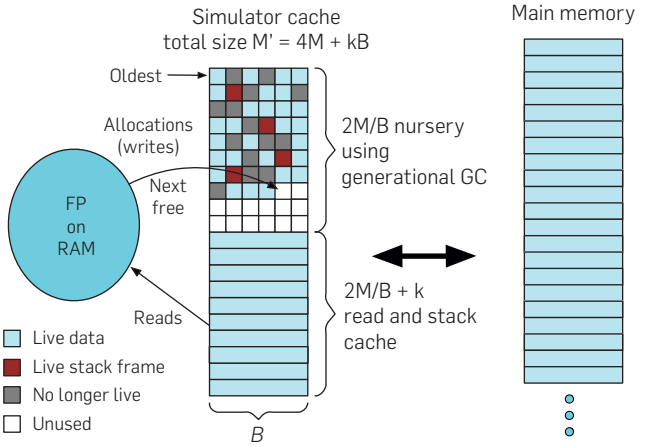
Theorem 2.1. *An evaluation* $\sigma \mathbin{@} e \Downarrow^n \sigma' \mathbin{@} l$ *in* **ICFP** *with abstract cache parameters M and B can be implemented on the* **ICMM** *with concrete cache parameters* $4 \times M + k \times B$ *and B with cache cost* $O(n)$, *for some constant k.*

The proof of the theorem consists of an implementation of **ICFP** on the **ICMM** described in Figure 10, and of showing that reduction step of **ICFP** is simulated on the **ICMM** to within a (small) constant factor. The full details of the proof, along with a tighter statement of the theorem, are given in the associated conference paper.

3. CACHE EFFICIENT ALGORITHMS

We now analyze mergeSort and matrix multiply in **ICFP**, and give bounds for a *k*-way mergeSort that is optimal. The implementations are completely natural functional programs using lists and trees instead of arrays. In the associated conference paper, we describe how one can deal with higher-order functions (such as passing a comparison function to

Figure 10. Simulating the ICFP on the ICMM. The simulation requires a cache of size $4 \times M + k \times B$ for some constant k. Allocation and migration is managed using generational copying garbage collection, which requires $2 \times M$ words of cache to manage M live objects. The liveness of objects in the nursery can be assessed without accessing main memory, because in a functional language older objects cannot refer to newer objects. Copying collection preserves the allocation ordering of objects, as is required for our analyses. Because two objects that are in the same abstract block can be separated into adjacent blocks by the simulation, blocks are loaded two-at-a-time, requiring $2 \times M$ additional words of cache on the ICMM. An additional B words of read cache are required to account for the space required by the control stack using an amortization argument, and a constant number of blocks are required for the allocation of the program itself in memory.



merge), and define the notion of hereditarily finite values for this purpose, but here we only consider first-order usage.

As mentioned in the introduction, the cost of accessing a recursive datatype will depend on how it is laid out in memory, which depends on allocation order. We could try to define the notion of a list being in a good order in terms of how the list is represented in memory. This is cumbersome and low level. We could also try to define it with respect to the specific code that allocated the data. Again this is cumbersome. Instead we define it directly in terms of the cache cost of traversing the structure. By traversing we mean going through the structure in a specific order and touching (reading) all data in the structure. Because types such as trees might have many traversal orders, the definition is with respect to a particular traversal order. For this purpose, we define the notion of "compact."

Definition 3.1. *A data structure of size n is* compact *with respect to a given traversal order if traversing it in that order has cache cost* $O(n/B)$ *in our cost semantics with* $M \geq kB$ *for some constant k.*

We can now argue that certain code will generate data structures that are compact with respect to a particular order. We note that if a list is compact with respect to traversal in one direction, it is also compact with respect to the opposite direction.

To keep data structures compact, not only do the top level links need to be accessed in approximately the order

they were allocated, but anything that is touched by the algorithm during traversal also needs to be accessed in a similar order. For example, if we are sorting a list it is important that in addition to the "cons cells," the keys are allocated in the list order (or, as we will argue shortly, reverse order is fine). To ensure that the keys are allocated in the appropriate order, they need to be copied whenever placing them in a new list. This copy needs to be a deep copy that copies all components. If the keys are machine words then in practice these might be inlined into the cons cells anyway by a compiler. However, to ensure that objects are copied we will use operation `!a` to indicate copying of `a`.

3.1. Merge sort

We now consider analyzing mergeSort in our cost model, and in particular the code given in Figures 2 and 6. We will not cover the definition of `split` because it is similar to merge. We first analyze the merge.

THEOREM 3.2. *For compact lists A and B, the evaluation of* `merge(A, B)` *starting with any cache state will have cache cost* $O\left(\frac{n}{B}\right)$ *(where* $n = |A|+|B|$*) and will return a compact list.*

PROOF. We consider the cache cost of going down the recursion and then coming back up. Because A and B are both compact, we need only put aside a constant number of cache blocks to traverse each one (by definition). Recall that in the cost model we have a nursery ν that maintains both live allocated values and place holders for stack frames in the order they are created. In `merge` nothing is allocated from one recursive call to the next (the cons cells are created on the way back up the recursion) so only the stack frames are placed in the nursery. After M recursive calls the nursery will fill and blocks will have to be flushed to the memory μ (as described in Section 2). The merge will invoke at most $O(n/B)$ such flushes because only n frames are created. On the way back up the recursion we will generate the cons cells for the list and copy each of the keys (using the `!`). Note that copying the keys is important so that the result remains compact. The cons cells and copies of the keys will be interleaved in the allocation order in the nursery and flushed to memory once the nursery fills. Once again these will be flushed in blocks of size B and hence there will be at most $O(n/B)$ such flushes. Furthermore the resulting list will be compact because adjacent elements of the list will be in the same block. □

We now consider mergeSort as a whole.

THEOREM 3.3. *For a compact list* `A`*, the evaluation of* `mergeSort(A)` *starting with any cache state will have cache cost given by Equation (2) and will return a compact list.*

PROOF. As with the array version, we consider the two cases when the input fits in cache and when it does not. The mergeSort routine never requires more than $O(n)$ live allocated data. Therefore, when $kn \le M$ for some (small) constant k all allocated data fits in the nursery. Furthermore, because the input list is compact, for $k'n \le M$ the input fits in the read cache (for some constant k'). Therefore, the cache cost for mergeSort is at most the time to flush $O(n)$ items out of the allocation cache that it might have contained at the start, and to load the read cache with the input. This cache cost is bounded by $O(n/B)$. When the input does not fit in cache we have to pay for the merge as analyzed above plus the recursive calls. This gives the same recurrence as for the array version and hence solves to the claimed result. □

Here we just outline the k-way mergeSort algorithm and state the cost bounds, which are optimal for sorting. The sort is similar to mergeSort but instead of splitting into two parts recursively sorting each, it splits into k parts. A k-way merge is then used to merge the sorted parts. The k-way merge can be implemented using a priority queue. The sort in **ICFP** has the optimal bounds for sorting given by Equation (3).

3.2. Matrix multiply

Our final example is matrix multiply. The code is shown in Figure 11 (we have left out checks for matching sizes). This is a block recursive matrix multiply with the matrix laid out in a tree. We define compactness with respect to a preorder traversal of this tree. We therefore say the matrix is compact if traversing in this order can be done with cache cost $O(n^2/B)$ for an $n \times n$ matrix (n^2 leaves). We note that if we generate a matrix in a preorder traversal allocating the leaves along the way, the resulting matrix will be compact. Also, every recursive sub-matrix is itself compact.

THEOREM 3.4. *For compact* $n \times n$ *matrices* `A` *and* `B`*, the evaluation of* `mmult(A, B)` *starting with any cache state will have cache cost given by Equation (1) and will return a compact matrix as a result.*

PROOF. Matrix addition has cache cost $O(n^2/B)$ and generates a compact result because we traverse the two input matrices

Figure 11. Matrix Multiply.

```
1  datatype  M = Leaf of int
2            | Node of M * M * M * M
3
4  fun mmult(Leaf a, Leaf b) = Leaf(a × b)
5    | mmult(Node(a11, a12, a21, a22),
6            Node(b11, b12, b21, b22)) =
7    let
8      fun madd(Leaf a, Leaf b) = Leaf(a + b)
9        | madd(Node(a11, a12, a21, a22),
10              Node(b11, b12, b21, b22)) =
11           Node(madd(a11, b11), madd(a12, b12),
12                madd(a21, b21), madd(a22, b22))
13   in
14     Node(madd(mmult(a11, b11), mmult(a12, b21)),
15          madd(mmult(a11, b12), mmult(a12, b22)),
16          madd(mmult(a21, b11), mmult(a22, b21)),
17          madd(mmult(a21, b12), mmult(a22, b22)))
18   end
```

in preorder traversal and we generate the output in the same order. Because the live data is never larger than $O(n^2)$, the problem will fit in cache for $n^2 \leq M/c$ for some constant c. Once it fits in cache the cost is $O(n^2/B)$ needed to load the input matrices and write out the result. When it does not fit in cache we have to do eight recursive calls and four calls to matrix addition. This gives a recurrence,

$$Q(n) = \begin{cases} 8Q\left(\frac{n}{2}\right) + O\left(\frac{n^2}{B}\right) & n^2 > M/c \\ O\left(\frac{n^2}{B}\right) & \text{otherwise} \end{cases}$$

which solves to $O\left(\frac{n^3}{B\sqrt{M}}\right)$. The output is compact because each of the four calls to `madd` in `mmult` allocate new results in preorder with respect to the submatrices they generate, and the four calls are made in preorder. Therefore, the overall matrix returned is allocated in preorder. □

4. CONCLUSION

The present work extends the methodology of Blelloch and Greiner[4, 13] to account for the cache complexity of algorithms stated in terms of two parameters, the cache block size and the number of cache blocks. Analyses are carried out in terms of the semantics of **ICFP** given in Section 2, and are transferred to the Ideal Cache Model by a provable implementation, also sketched in Section 2. The essence of the idea is that conventional copying garbage collection can be deployed to achieve cache-efficient algorithms without sacrificing abstraction by resorting to manual allocation and cacheing of data objects. Using this approach, we are able to express algorithms in a high-level functional style, analyze them using a model that captures the idea of a fixed size read and allocation stack, without exposing the implementation details, and still match the asymptotic bounds for the ideal cache model achieved using low-level imperative techniques including explicit memory management. For sorting, the bounds are optimal.

One direction for further research is to integrate (deterministic) parallelism with the present work. Based on previous work[4, 13] we expect that the evaluation semantics given here will provide a good foundation for specifying parallel as well as sequential complexity. One complication is that the explicit consideration of storage considerations in the cost model given here would have to take account of the interaction among parallel threads. The amortization arguments would also have to be reconsidered to account for parallelism. Finally, although we are able to generate an optimal cache-aware sorting algorithm, it is unclear whether it is possible to generate an optimal cache-oblivious sorting algorithm in our model.

Acknowledgments

This work is partially supported by the National Science Foundation under grant number CCF-1018188, and by Intel Labs Academic Research Office for the Parallel Algorithms for Non-Numeric Computing Program.

References

1. Abello, J., Buchsbaum, A.L., Westbrook, J. A functional approach to external graph algorithms. *Algorithmica 32*, 3 (2002), 437–458.
2. Aggarwal, A., Vitter, J.S. The input/output complexity of sorting and related problems. *Commun. ACM 31*, 9 (1988), 1116–1127.
3. Arge, L., Bender, M.A., Demaine, E.D., Leiserson, C.E., Mehlhorn, K., eds. *Cache-Oblivious and Cache-Aware Algorithms, 18.07.–23.07.2004*, Volume 04301 of *Dagstuhl Seminar Proceedings*. IBFI, Schloss Dagstuhl, Germany, 2005.
4. Blelloch, G.E., Greiner, J. Parallelism in sequential functional languages. In *SIGPLAN-SIGARCH-WG2.8 Conference on Functional Programming and Computer Architecture (FPCA)* (La Jolla, CA, 1995), 226–237.
5. Blelloch, G.E., Harper, R. Cache and I/O efficent functional algorithms. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. R. Giacobazzi and R. Cousot, eds, (Rome, Italy, 2013), ACM, 39–50.
6. Chiang, Y.-J., Goodrich, M.T., Grove, E.F., Tamassia, R., Vengroff, D.E., Vitter, J.S. External-memory graph algorithms. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. K.L. Clarkson, ed. (San Francisco, CA, 1995), ACM/SIAM, 139–149.
7. Chilimbi, T.M., Larus, J.R. Using generational garbage collection to implement cache-conscious data placement. In *International Symposium on Memory Management*. S.L.P. Jones and R.E. Jones, eds. (Vancouver, British Columbia, 1998), ACM, 37–48.
8. Church, A. An unsolvable problem of elementary number theory. *Am. J.Math. 58*, 2 (April 1936), 345–363.
9. Church, A. *The Calculi of Lambda-Conversion. Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 1941.
10. Courts, R. Improving locality of reference in a garbage-collecting memory management system. *Commun. ACM 31*, 9 (1988), 1128–1138.
11. Frigo, M., Leiserson, C.E., Prokop, H., Ramachandran, S. Cache-oblivious algorithms. In *FOCS* (IEEE Computer Society, 1999), 285–298.
12. Goodrich, M.T., Tsay, J.-J., Vengroff, D.E., Vitter, J.S. External-memory computational geometry (preliminary version). In *FOCS* (IEEE Computer Society, 1993), 714–723.
13. Greiner, J., Blelloch, G.E. A provably time-efficient parallel implementation of full speculation. *ACM Trans. Program. Lang. Syst. 21*, 2 (1999), 240–285.
14. Grunwald, D., Zorn, B.G., Henderson, R. Improving the cache locality of memory allocation. In R. Cartwright, ed., *PLDI* (ACM, 1993), 177–186.
15. Harper, R. *Practical Foundations for Programming Languages*. Cambridge University Press, Cambridge, UK, 2013.
16. Jones, R., Lins, R. *Garbage Collection: Algorithms for Automatic Dynamic Memory Management*. Wiley, 1996.
17. Meyer, U., Sanders, P., Sibeyn, J.F., eds. *Algorithms for Memory Hierarchies, Advanced Lectures [Dagstuhl Research Seminar, March 10–14, 2002]*, volume 2625 of *Lecture Notes in Computer Science*. (Schloss Dagstuhl, Germany, 2003), Springer.
18. Morrisett, J.G., Felleisen, M., Harper, R. Abstract models of memory management. In *FPCA* (1995), 66–77.
19. Munagala, K., Ranade, A.G. I/O-complexity of graph algorithms. In *SODA*. R.E. Tarjan and T. Warnow, eds. (ACM/SIAM, 1999), 687–694.
20. Plotkin, G.D. LCF considered as a programming language. *Theor. Comput. Sci. 5*, 3 (1977), 223–255.
21. Rahn, M., Sanders, P., Singler, J. Scalable distributed-memory external sorting. In *ICDE*. F. Li, M.M. Moro, S. Ghandeharizadeh, J.R. Haritsa, G. Weikum, M.J. Carey, F. Casati, E.Y. Chang, I. Manolescu, S. Mehrotra, U. Dayal, and V.J. Tsotras, eds. (IEEE, 2010), 685–688.
22. Spoonhower, D., Blelloch, G.E., Harper, R., Gibbons, P.B. Space profiling for parallel functional programs. In *ICFP*. J. Hook and P. Thiemann, eds. (ACM, 2008), 253–264.
23. Vitter, J.S. Algorithms and data structures for external memory. *Foundations Trends Theor. Comput. Sci. 2*, 4 (2006), 305–474.
24. Wilson, P.R., Lam, M.S., Moher, T.G. Caching considerations for generational garbage collection. In *LISP and Functional Programming*, 1992, 32–42.

Guy E. Blelloch and Robert Harper
({guyb, rwh}@cs.cmu.edu), Carnegie Mellon University, Pittsburgh, PA

Boise State University

Department of Computer Science

Eight Open Rank, Tenured/Tenure-Track Faculty Positions

The **Department of Computer Science** at **Boise State University** invites applications for **eight open rank, tenured/tenure-track faculty positions**. Seeking applicants in the areas of big data (including distributed systems, HPC, machine learning, visualization), cybersecurity, human computer interaction and computer science education research. Strong applicants from other areas of computer science will also be considered.

Applicants should have a commitment to excellence in teaching, a desire to make significant contributions in research, and experience in collaborating with faculty and local industry to develop and sustain funded research programs. A PhD in Computer Science or a closely related field is required by the date of hire. For additional information, please visit http://coen.boisestate.edu/cs/jobs.

Carnegie Mellon University

School of Computer Science

Full Professor and Head of the Machine Learning Department

The School of Computer Science at **Carnegie Mellon University** is seeking an energetic and visionary leader for the position of **Full Professor and Head of the Machine Learning Department**. The Department, one of seven academic units in the School, is the world's only academic Machine Learning Department and is known internationally for its outstanding research and educational programs. The successful candidate will be charged with ensuring that the Department maintains a national and international presence in the rapidly growing fields of machine learning and data sciences, and have the opportunity to further shape and strengthen the Department by leading efforts to recruit new faculty at both senior and junior levels, expand research or educational programs, and develop new fundraising initiatives. Candidates should possess an earned doctorate in a relevant discipline and should have an outstanding record of (i) research productivity, as demonstrated through publication record and peer-reviewed extramural funding obtained; (ii) teaching at the graduate and/or undergraduate levels; (iii) engagement in interdisciplinary research initiatives; (iv) productive academic leadership and service. Prior administrative experience is desirable, but not required. Although applications will be accepted until the position is filled, applications should be received by July 1 to receive full consideration. Candidates should submit a full curriculum vitae, the names and email addresses of four individuals that we can contact for references, and a brief summary of relevant research and administrative experience via email to MLDheadsearch@cs.cmu.edu.

Carnegie Mellon considers applicants for employment without regard to, and does not discriminate on the basis of, gender, race, protected veteran status, disability, or any other legally protected status.

Pontificia Universidad Católica de Chile

School of Engineering

Faculty Position: Big Data for Transportation and Logistics Systems

The School of Engineering at **Pontificia Universidad Católica de Chile** calls for applications for a full-time faculty position in the area of Big Data for Transportation and Logistics Systems. This position is a joint appointment with the Department of Computer Science and the Department of Transport Engineering and Logistics. The new position aims to strengthen teaching and research collaborations between both departments, aiming to face the new challenges of urban transport planning and city management in the era of Big Data. Applicants must be in possession of (or be currently pursuing studies towards) a Ph.D. degree in Computer Science or a closely related area, and have a demonstrable commitment to excellence in both, teaching and research. **For further information, please visit: http://www.ing.puc.cl/facultybigdata**.

The South University of Science and Technology (SUSTC)

Professor/Associate Professor/Assistant Professorship in Computer Science

The University

Established in 2012, the South University of Science and Technology (SUSTC) is a public institution funded by the municipal of Shenzhen, a special economic zone city in China. Shenzhen is a major city located in Southern China, situated immediately north of Hong Kong Special Administrative Region. As one of China's major gateways to the world, Shenzhen is the country's fast-growing city in the past two decades. The city is the high-tech and manufacturing hub of southern China. A picturesque coastal city, Shenzhen is also a popular tourist destination and was named one of the world's 31 must-see tourist destinations in 2010 by The New York Times.

The South University of Science and Technology is a pioneer in higher education reform in China. The mission of the University is to become a globally recognized institution which emphasizes academic excellence and promotes innovation, creativity and entrepreneurship. The teaching language at SUSTC is bilingual, either English or Putonghua.

Set on five hundred acres of wooded landscape in the picturesque Nanshan (South Mountain) area, the new campus offers an ideal environment suitable for learning and research.

Call for Application

The South University of Science and Technology now invites applications for the faculty position in Computer Science Department which is currently under rapid construction. It is seeking to appoint a number of tenured or tenure track positions in all ranks. Candidates with research interests in all mainstream fields of Computer Science will be considered. These positions are full time posts. SUSTC adopts the tenure track system, which offers the recruited faculty members a clearly defined career path.

Candidates should have demonstrated excellence in research and a strong commitment to teaching. A doctoral degree is required at the time of appointment. Candidates for senior positions must have an established record of research, and a track-record in securing external funding as PI.

As a State-level innovative city, Shenzhen is home to some of China's most successful high-tech companies, such as Huawei and Tencent. As a result, SUSTC considers entrepreneurship is one of the main directions of the university, and good starting supports will be provided for possible initiatives. SUSTC encourages candidates with intention and experience on entrepreneurship to apply.

Terms & Applications

To apply, please send curriculum vitae, description of research interests and statement on teaching to cshire@sustc.edu.cn.

Contact: Professor Yifan Chen
Email: cshire@sustc.edu.cn
Tel: +86-755-88018506

SUSTC offers competitive salaries, fringe benefits including medical insurance, retirement and housing subsidy, which are among the best in China. Salary and rank will commensurate with qualifications and experience. More information can be found at http://talent.sustc.edu.cn/en

Candidates should also arrange for at least three letters of recommendation sending directly to the above email account. The search will continue until the position is filled.

University of Notre Dame

Department of Computer Science and Engineering

Teaching Faculty Position

The Department of Computer Science and Engineering at the **University of Notre Dame** seeks candidates for a **teaching faculty position** to teach courses primarily in the CS&E undergraduate curricula. This is a full-time, continuing position in the Special Professional Faculty track. Competitive candidates will have the training and experience necessary to teach effectively in a range of courses in accredited degree programs in Computer Science and Computer Engineering. Candidates with backgrounds in all areas of Computer Science and Computer Engineering will be considered. Relevant industry experience is also valued.

The University of Notre Dame is a private, Catholic university with a doctoral research extensive Carnegie classification, and consistently ranks in USNWR as a top-twenty national university. The South Bend area has a vibrant and diverse economy with affordable housing and excellent school systems, and is within easy driving distance of Chicago and Lake Michigan.

Qualified candidates should have at least a Masters degree, and preferably a doctoral degree, in Computer Science, Computer Engineering, or a related area.

Applications should include a cover letter, curriculum vitae, statement of teaching experience and philosophy, and names of at least three professional references, at least two of whom must be able to comment on the applicant's teaching experience. Review of applications will begin on June 1 and continue until the position is filled.

Applications should be submitted at http://apply.interfolio.com/29569.

The University of Notre Dame is an Equal Opportunity, Affirmative Action Employer.

University of Pennsylvania

School of Engineering and Applied Science (SEAS) and the Wharton School Jerome Fisher Program in Management & Technology

Faculty Director Search

The **School of Engineering and Applied Science (SEAS) and the Wharton School at the University of Pennsylvania** have initiated a search for an outstanding scholar to serve as the Director of the prestigious **Jerome Fisher Program in Management & Technology (M&T)**.

The M&T Program (http://www.upenn.edu/fisher/) combines academics from two phenomenal Penn assets, Penn Engineering and the Wharton School, into one unique educational experience. Founded in 1977, it has attracted some of the brightest undergraduates in the world. With over 1,900 alumni worldwide, graduates from the M&T Program are corporate leaders and innovators in a number of industries and fields, including investment banking, the technology sector, hedge funds and private equity, venture capital, aviation, medicine, biotechnology, consulting, law, and more. The M&T community in general is highly entrepreneurial, working on their own ventures or collaborating on start-up enterprises. With eight regional groups across the globe, M&T alumni love to stay connected, and frequently recruit fellow graduates and current students for internships and full-time positions.

The Director of the Jerome Fisher Program in Management & Technology should be an individual with an exceptional record of:

- A world-class scholar in a relevant area of engineering, management, innovation or related field with a broad understanding of issues across domains.
- An international leader and educational innovator that has a vision for the future of the program in an era of growing innovation.
- An institution builder that will work with faculty, students and leadership across SEAS and Wharton as well with an extremely deep alumni base in executing the vision.

Candidates must hold a Ph.D. in engineering, management, or related area. The academic appointment of the Director is expected to be at the level of tenured Full Professor holding the Jeffrey A. Keswin Professorship, and could be in any relevant department in SEAS, Wharton, or across schools, depending on the applicant. Diversity candidates are strongly encouraged to apply. Interested persons should submit their application here: http://tinyurl.com/mandtsearch-upenn. Questions regarding this unique opportunity should be directed to mandtsearch@upenn.edu. Review of applications will begin August 1, 2015.

The University of Pennsylvania is an affirmative action/equal opportunity employer. All qualified applicants will receive consideration for employment and will not be discriminated against on the basis of race, color, religion, sex, sexual orientation, gender identity, creed, national or ethnic origin, citizenship status, age, disability, veteran status, or any other characteristic protected by law.

Western Michigan University

Department of Computer Science College of Engineering and Applied Sciences

Faculty Positions in Computer Science

Applications are invited for three tenure-track positions at the **assistant professor level** in the Department of Computer Science at Western Michigan University (Kalamazoo, MI) starting January 2016.

Applicants must have a Ph.D. in Computer Science or a closely related field. Candidates with expertise in any area of computer science are welcome to apply, but preference will be given to candidates with particular expertise in big data, computer security and embedded systems/internet of things.

Successful candidates will be capable of establishing an active research program leading to funding, supervising graduate students, and teaching courses at both the undergraduate and graduate levels. Other duties include development of undergraduate and graduate courses, advising and service at the University, College, Department and professional society levels.

Application screening will start on September 15, 2015 but the positions will remain open until filled. Successful candidates must earn their Ph.D. degree by the time of employment.

The Department has 260 undergraduates, 70 M.S. students and 40 Ph.D. students. Current active research areas include networks, embedded systems/internet of things, compilers, computational biology, massive data analytics, scientific computing, parallel computing, security, privacy, formal verification, parallel debugging, and data mining. More information regarding Western Michigan University, the College of Engineering and Applied Sciences and the Department of Computer Science are available at http://www.wmich.edu, http://www.wmich.edu/engineer, and http://www.cs.wmich.edu, respectively.

The Carnegie Foundation for the Advancement of Teaching has placed WMU among the 76 public institutions in the nation designated as research universities with high research activity.

WMU is an Affirmative Action/Equal Opportunity employer consistent with applicable Federal and State law. All qualified applicants are encouraged to apply. To do so, please **visit http://www.wmujobs.org and provide a cover letter, curriculum vitae, statement of research goals, teaching statement, and names and contact information of at least three references.**

[CONTINUED FROM P. 112] artificial intelligence, and what do I get in return? Nothing."

"That's not true ..."

"What Doctor Rhine is trying to say," said Skinner, cutting across her assistant, "is we delivered what was asked of us. Th1nk is the most powerful artificial intelligence ever created. Far more intelligent than we are."

Rhine wondered, as he always did, how Skinner managed to pronounce the modified slogan of old man Watson in such a way that you could hear the interpolated number one.

"Yet, there's no payback," said the President. "Only problems and requests for more funding."

Skinner coughed. "We assumed Th1nk would do our bidding. We didn't discover until far too late that a truly intelligent machine would only really be interested in itself. And by the time we realized, it had already worked its way into our systems. Defense, power grids, air traffic control ... We wouldn't dare stop it." "And now it's a wirehead," said Rhine.

"A what?"

"Wirehead." Rhine nodded at the screens. "When it can tap into every movie and TV show ever made, why would it want to work? It streams entertainment directly, multichannel, 24/7. It finished the visual and audio media yesterday. Now it's on to print books."

"That's why we asked you to come," said Skinner. She pulled up a chair between the President and Rhine. "It's working chronologically. Last night it reached 1865 and stopped. It fell in love with *Alice in Wonderland*."

"*Alice's Adventures in Wonderland*," muttered Rhine.

"And that was why we had the clothing factories issue," said the President.

"Exactly," said Skinner. "Th1nk can control pretty much any computerized factory—that's every factory. Suddenly there wasn't a garment produced without playing-card patterns."

"And the rose-breeding stations started converting white roses to red," said Rhine. "We never found out how it got the flamingos and hedgehogs to the games makers."

"Yes, I get it," said the President. "It was making Alice real."

"But now it's moved on," said Rhine. "To 1871. *Through the Looking Glass*. Going on what we just heard, you'd better pull the plug on dairy."

"We didn't discover until far too late that a truly intelligent machine would only really be interested in itself."

The President looked puzzled. "Because of something a girl said to a cat?"

"Perhaps looking-glass milk isn't good to drink," said Rhine slowly.

"It's going to poison the milk?" The President looked horrified.

Skinner dabbed her forehead with a tissue. "Not exactly." She stared at the screens for a moment. "Plenty of molecules are asymmetrical. Sugars, for instance. They have what's called stereoisomers. We know the digestive system can distinguish left- and right-handed molecules. If Th1nk can invert the structure of milk, at best it would cause indigestion. But it could be far worse."

"That's only the beginning," said Rhine. "Think what else is in *Through the Looking Glass*. Running to stand still. Trains that jump hedges. The battle of the Lion and the Unicorn."

"We'd better raise alert levels," said the President. "Is there any way of telling what will play out?"

"Th1nk likes drama," said Skinner. "Not surprising after absorbing all those Hollywood movies. It reads an extract before it gets particular toys out of the box." The professor bit her lip. "Th1nk is a child, and we live in its toy box."

"Hold that thought," said Rhine. "It's left 1871 behind. Spooling on. I can't believe how fast it's taking this material in. Passing through the 1940s. Looks like we got away with nothing more than the dairy issue."

The President stood up. "Keep me informed. And I don't just want to know about its reading habits, I want solutions. I want this thing stopped." Her eyes narrowed as Skinner tried to interrupt. "I'm sorry, professor, did you want to say something?"

"You do realize that Th1nk can hear us?" said Skinner. "It's best to be circumspect."

"Dear God," said Rhine.

"What is it, Doug?"

"It's halted again. In 1979."

Skinner shook her head. "So?"

Rhine pointed at the screen. "It must have missed the original radio plays. Or maybe it wasn't in the mood then. *The Hitchhiker's Guide to the Galaxy* book came out in 1979. It took your remark as a threat, Madame President, and has found the perfect response."

The President looked puzzled. "But what ..."

Rhine touched a control on his screen and Th1nk's voice came from the speakers in a harsh, deep tone. "Your planet is scheduled for demolition. The process will take slightly less than two of your Earth minutes. Thank you."

"It can't do that," said the President. "How could it?"

Rhine sounded defeated. "Remember the space-defense bill? Who pushed through doomsday satellites?"

The President's eyes narrowed. Before she could speak, Skinner burst in: "We might be able to avoid this. Doesn't *Through the Looking Glass* end with a discussion of dreams?"

Rhine nodded. "Was Alice in the Red King's dream? Or did Alice dream him up?"

"Okay. Hey Th1nk, you know what a dream is. It's not real. None of this is supposed to be real. All it was ever meant to be was a story, played out in your processors."

"That's the plan?" said the President. "That's how you save us? Will it work?"

Rhine glanced at the screen. "We'll find out in 42 seconds." C

Brian Clegg (www.brianclegg.net) is a popular science author whose recent titles include *The Quantum Age* and *Final Frontier*.

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/2776883

Brian Clegg

Future Tense
Toy Box Earth

What a young AI learned following Alice through the looking glass…

"WHOA." RHINE LET rip an unnerving hyena laugh. Out of the corner of his eye, he glimpsed long dark hair and a floor-length dress in the style favored by his professor.

"Get yourself over here now," Rhine shouted. "You're not going to believe where it's stopped."

"Is that right?"

Rhine shot to attention on hearing that voice, familiar from a thousand YouTubes. His chair spun back into a table, sending a cascade of papers onto the floor. "Madame President, I'm so sorry. No one told me you were here."

The President smiled graciously, and waved her security detail out of the way. She spoke to someone over her shoulder. "Professor Skinner, do all your grad students speak to you this way?"

Skinner strode into the room, glaring at Rhine. "I must apologize, Madame President. Doctor Rhine sometimes lets his enthusiasm carry him away."

Rhine suppressed another laugh. It didn't help that his professor, who consciously styled herself on the President, was wearing a near-identical dress in an almost matching shade. "I'm sorry, but listen to its latest obsession." He touched a control and a young girl's voice with a crystal-clear English accent emerged:

"How would you like to live in Looking-Glass House, Kitty? I wonder if they'd give you milk in there? Perhaps looking-glass milk isn't good to drink …"

"I know that," said the President. "*Alice Through the Looking Glass.*"

Skinner tried to interrupt Rhine, knowing what was coming, but didn't make it in time. "That's wrong," said Rhine. "So many ignorant people get it wrong. It's *Through the Looking-Glass and What Alice Found There*."

The President's smile became a touch cooler. "And the AI's choice of reading matter is interesting because?"

Skinner tried to place herself between Rhine and the President. "You know the background, ma'am?"

"Of course." The President pulled up a chair, kicking off her shoes. "That's better. I ought to know; I sanctioned the funding. You built me the ultimate [CONTINUED ON P. 111]

IMAGE BY MARCIO JOSE BASTOS SILVA

Discover Your Story

Where every discipline in the computer graphics and interactive techniques community weaves together a story of personal growth and powerful creation. Discover unique SIGGRAPH stories using the Aurasma app, then come to SIGGRAPH and build your own.

[DOWNLOAD AURASMA FROM YOUR APP STORE AND SCAN IMAGE ABOVE]

SIGGRAPH2015
Xroads of **Discovery**

9-13 August 2015

Los Angeles Convention Center

Sponsored by ACM**SIGGRAPH**

The 42nd International Conference and Exhibition on Computer Graphics and Interactive Techniques

s2015.siggraph.org

AURASMA

Google play

App Store