COMMUNICATIONS OF THE ACM
CACM.ACM.ORG
09/2015 VOL.58 NO.09

Commonsense Reasoning and Commonsense Knowledge in Artificial Intelligence

Sensing Emotions

Trustworthy Hardware from Untrusted Components

Q&A with Dan Boneh

Should Conferences Meet Journals and Where? A Proposal for 'PACM'

Association for Computing Machinery

IPDPS 2016 CHICAGO

30th IEEE INTERNATIONAL Parallel and Distributed Processing SYMPOSIUM

MAY 23-27, 2016
CHICAGO HYATT REGENCY
CHICAGO, ILLINOIS USA

IPDPS 2016 will be held in Chicago, the third largest metropolis in the USA. It boasts a dual-hub airport system and 46 non-stop global flights daily. Situated along the shoreline of Lake Michigan, downtown Chicago (aka the Loop), where the Hyatt Regency is located, is an easy walk or cab ride to the attractions the city offers, including parks, beaches, shopping, museums, live entertainment, and the singular international cuisine of Chicago's diverse ethnic neighborhood eateries. Check the IPDPS Web pages for updates and information on workshops, the PhD Forum, and other events as the full program develops.

CALL FOR PAPERS

Authors are invited to submit manuscripts that present original unpublished research in all areas of parallel and distributed processing, including the development of experimental or commercial systems. Work focusing on emerging technologies is especially welcome.

Topics of interest include:

- **Parallel and distributed algorithms**, focusing on topics such as: numerical, combinatorial, and data-intensive parallel algorithms, locality-aware and power-aware parallel algorithms, streaming algorithms, parallel algorithms in specific domains such as machine learning and network science, scalability of algorithms and data structures for parallel and distributed systems, communication and synchronization protocols, network algorithms, scheduling, and load balancing.

- **Applications of parallel and distributed computing**, including computational and data-enabled science and engineering, big data applications, parallel crowd sourcing, large-scale social network analysis, management of big data, cloud and grid computing, scientific, biological and medical applications, and mobile computing. Papers focusing on applications using novel commercial or research architectures, big data approaches, or discussing scalability toward the exascale level are encouraged.

- **Parallel and distributed architectures**, including architectures for instruction-level and thread-level parallelism; petascale and exascale systems designs; novel big data architectures; special purpose architectures, including graphics processors, signal processors, network processors, media accelerators, and other special purpose processors and accelerators; impact of technology on architecture; network and interconnect architectures; parallel I/O and storage systems; architecture of the memory hierarchy; power-efficient and green computing architectures; dependable architectures; and performance modeling and evaluation.

- **Parallel and distributed software**, including parallel and multicore programming languages and compilers, runtime systems, operating systems, resource management including green computing, middleware for grids, clouds, and data centers, libraries, performance modeling and evaluation, parallel programming paradigms, and programming environments and tools. Papers focusing on novel software systems for big data and exascale systems are encouraged.

IMPORTANT DATES

October 9, 2015
Submit Abstract

October 16, 2015
Submit Paper

December 18, 2015
Author Notification

After December 18, 2015
Deadlines for Paper Submissions to Workshops

WHAT/WHERE TO SUBMIT

Authors will need to register their paper and submit an abstract by October 9, 2015 and then submit full versions by October 16, 2015. More details on submissions and instructions for submitting files are available at www.ipdps.org. All submitted manuscripts will be reviewed. Submitted papers should NOT have appeared in or be under consideration for another conference, workshop or journal.

IPDPS 2016 WORKSHOPS

IPDPS workshops, held on the first and last day of the conference, are a major part of the IPDPS week-long family of events and provide the IPDPS community an opportunity to explore special topics and present work that is more preliminary and cutting-edge or that has more practical content than the more mature research presented in the main symposium. **IPDPS 2016 is introducing a new event: Roundtable Workshops.** These will be organized and animated by a few people and will be focused on emerging areas of interest in parallel and distributed computing, especially topics that complement and "round out" the areas covered by the regular workshops. For more information, see the IPDPS Website for details.

GENERAL CHAIR

Xian-He Sun (Illinois Institute of Technology, USA)

PROGRAM CHAIR

Jeffrey K. Hollingsworth (University of Maryland, USA)

PROGRAM VICE-CHAIRS

- **Algorithms:**
  Ümit V. Çatalyürek
  (Ohio State University, USA)
- **Applications:**
  Darren Kerbyson
  (Pacific Northwest National Laboratory, USA)
- **Architecture:**
  Andrew A. Chien
  (University of Chicago, USA)
  Hank Hoffman
  (University of Chicago, USA)
- **Software:**
  Karen Karavanic
  (Portland State University, USA)

- KEYNOTES & TECHNICAL SESSIONS
- WORKSHOPS & PHD FORUM
- COMMERCIAL PARTICIPATION

Details at www.ipdps.org

SPONSORED BY:
IEEE computer society

IN ASSOCIATION WITH:
ACM SIGARCH
IEEE Computer Society Technical Committee on Computer Architecture
IEEE Computer Society Technical Committee on Distributed Processing

Previous A.M. Turing Award Recipients

1966 A.J. Perlis
1967 Maurice Wilkes
1968 R.W. Hamming
1969 Marvin Minsky
1970 J.H. Wilkinson
1971 John McCarthy
1972 E.W. Dijkstra
1973 Charles Bachman
1974 Donald Knuth
1975 Allen Newell
1975 Herbert Simon
1976 Michael Rabin
1976 Dana Scott
1977 John Backus
1978 Robert Floyd
1979 Kenneth Iverson
1980 C.A.R Hoare
1981 Edgar Codd
1982 Stephen Cook
1983 Ken Thompson
1983 Dennis Ritchie
1984 Niklaus Wirth
1985 Richard Karp
1986 John Hopcroft
1986 Robert Tarjan
1987 John Cocke
1988 Ivan Sutherland
1989 William Kahan
1990 Fernando Corbató
1991 Robin Milner
1992 Butler Lampson
1993 Juris Hartmanis
1993 Richard Stearns
1994 Edward Feigenbaum
1994 Raj Reddy
1995 Manuel Blum
1996 Amir Pnueli
1997 Douglas Engelbart
1998 James Gray
1999 Frederick Brooks
2000 Andrew Yao
2001 Ole-Johan Dahl
2001 Kristen Nygaard
2002 Leonard Adleman
2002 Ronald Rivest
2002 Adi Shamir
2003 Alan Kay
2004 Vinton Cerf
2004 Robert Kahn
2005 Peter Naur
2006 Frances E. Allen
2007 Edmund Clarke
2007 E. Allen Emerson
2007 Joseph Sifakis
2008 Barbara Liskov
2009 Charles P. Thacker
2010 Leslie G. Valiant
2011 Judea Pearl
2012 Shafi Goldwasser
2012 Silvio Micali
2013 Leslie Lamport
2014 Michael Stonebraker

ACM A.M. TURING AWARD NOMINATIONS SOLICITED

Nominations are invited for the 2015 ACM A.M. Turing Award. This is ACM's oldest and most prestigious award and is presented annually for major contributions of lasting importance to computing. Although the long-term influences of the nominee's work are taken into consideration, there should be a particular outstanding and trendsetting technical achievement that constitutes the principal claim to the award. The recipient presents an address at an ACM event that will be published in an ACM journal. The award is accompanied by a prize of $1,000,000. Financial support for the award is provided by Google Inc.

Nomination information and the online submission form are available on:
http://amturing.acm.org/call_for_nominations.cfm

Additional information on the Turing Laureates is available on:
http://amturing.acm.org/byyear.cfm

The deadline for nominations/endorsements is November 30, 2015.

For additional information on ACM's award program please visit: www.acm.org/awards/

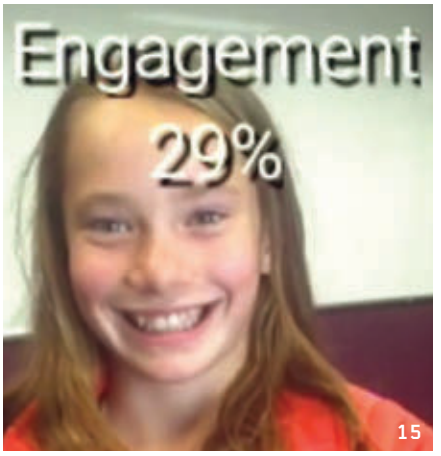
Association for Computing Machinery

COMMUNICATIONS OF THE ACM

Departments

Last Byte

News



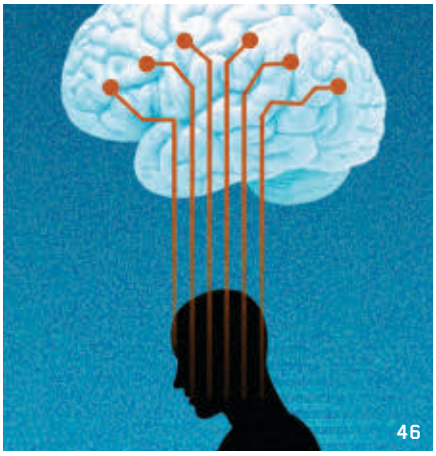
Viewpoints

Viewpoints

Association for Computing Machinery
Advancing Computing as a Science & Profession

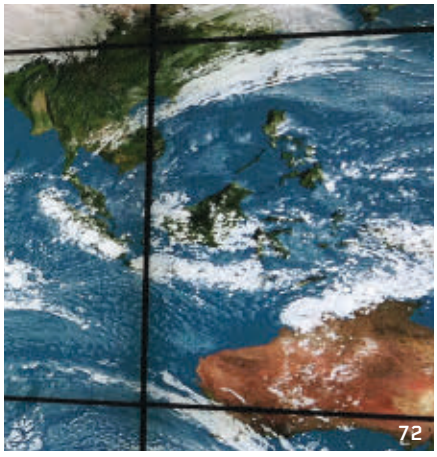
09/2015
VOL. 58 NO. 09

Practice

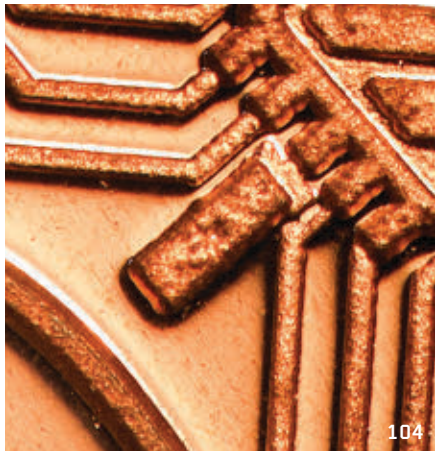


Articles' development led by acmqueue
queue.acm.org

Contributed Articles

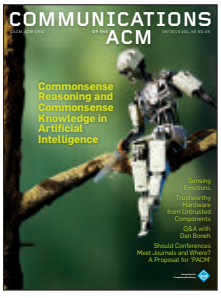


Review Articles



Watch the authors discuss their work in this exclusive *Communications* video. http://cacm.acm.org/videos/commonsense-reasoning-and-commonsense-knowledge-in-artificial-intelligence

Research Highlights



About the Cover: Major advances in AI may be plentiful in some respects, but when it comes to achievements in basic commonsense, success remains painfully slow. Ernest Davis and Gary Marcus explore the challenges of understanding human commonsense on p. 92. Cover illustration by Peter Crowther Associates.

IMAGES BY STUART BRIERS; UNM RESEARCH DATA SERVICES; ASAF ELIASON

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:10.1145/2811400

Joseph A. Konstan and Jack W. Davidson

Should Conferences Meet Journals and Where? A Proposal for 'PACM'

For many months now, the ACM Publications Board Conferences Committee has been working on a proposal that brings together conference and journal publishing.

Here, we offer some background leading up to this proposal, summarize the plan, and solicit your input. We are also pleased to present two leading computer scientists—Kathryn S. McKinley and David S. Rosenblum—who argue for and against the proposal beginning on page 43.

This proposal has been driven by strong input from the CS research community where the prevailing feeling is, despite strong review processes and selective acceptance, publishing in conference proceedings puts CS researchers at a disadvantage with respect to researchers in other scientific disciplines, where journal publication predominates, and is hence fundamentally detrimental to the CS field.

While some claim U.S.-based efforts by the National Research Council and the Computing Research Association have helped make top-quality conference papers "count" as much or more than peer-reviewed journal papers, others claim conference papers still have second-class status. Indeed, funding and merit systems in many countries do not fully recognize conference papers. Many believe the conference-centric culture of CS puts researchers at a disadvantage when competing with researchers in other science disciplines for top science awards and career progression.

The relationship between conferences and journals is complex. Many computer scientists lament the inherent flaws and limitations of conference publishing: hard deadlines, page limits, limited review and revision time, and overloaded review committees. Attending conferences comes at a cost, limiting access. And some of the initial reasons CS publishing favored conferences—the lack of CS journals, the slow speed of journal publishing, and the high cost of journal papers—have mostly disappeared. Still, many in our community argue that conferences are where the most exciting research is published; they seek to maintain the vibrant exchange that happens there. For this reason, we charged a diverse task force to explore whether the field would be better served through new models of conference-journal interaction.

What Is the Status Quo?

ACM currently offers three ways to publish conference papers in journals.

Revision of the paper by adding at least 25%–33% new content and submitting to a journal for review. Disadvantages include "citation splitting" and lag time.

Journal-first publication where the paper is reviewed by the journal and published there, but the authors are invited to present at the conference. This option is the basis of the successful HiPEAC conference, whose papers are submitted directly to ACM TACO, and is increasingly common as an option in several SIG-sponsored conferences.

Journal-integrated publication where the conference integrates its committee review process with a journal. Papers accepted by the conference-review process are published in the journal and presented at the conference. Papers that need additional rounds of review are transferred to the journal-review process. Papers accepted through the journal-review process can be presented at a later conference. This model is used by SIGGRAPH in conjunction with ACM TOG.

New Proposal: A Proceedings-Focused Journal Series

The committee proposes a fourth alternative—a journal series specifically created to publish the proceedings of ACM's highest quality conferences. Tentatively called *Proceedings of the ACM*, it would parallel ACM Transactions with a set of journals publishing high-quality research vetted by research communities through conferences. A few key features and criteria:

- A series would cover several conferences related by SIG or theme (for example, *PACM Programming Languages* or *PACM Distributed Systems*).
- All included proceedings papers would be reviewed for correctness, accuracy, completeness, and impact through a selective peer-review process. This process would require written reviews by a minimum number of three qualified reviewers. Documented policies would ensure the integrity of the review process.
- The review process must permit at least one substantive revision by the authors and review of that revision by the reviewers to determine whether their concerns are adequately met.
- Papers may not have artificial limits on length that would prevent full disclosure of references, documentation of methods, and so on. Supplemental materials would be clearly labeled as to whether they underwent peer review.
- *PACM* series may invite submissions or second and subsequent revisions outside the conference timeline and review cycle.
- Inclusion of a conference proceeding in a *PACM* series is subject to initial and periodic review by the series steering committee and the ACM Publications Board.

We want to hear from you! What do you think? Please read the arguments for and against this proposal on p. 43 and tell us what *you* think at https://www.surveymonkey.com/r/PACM2015 by Sept. 20 if possible!

Joseph A. Konstan (konstan@umn.edu) and **Jack W. Davidson** (jwd@virginia.edu) are co-chairs of the ACM Publications Board.

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

DOI:10.1145/2811280

Vinton G. Cerf

On (Computing) Artifacts

The term *artifact* has more than one interpretation. *Moiré patterns* are artifacts produced when two very similar patterns, usually straight-lined, are overlaid and one is slightly rotated. *Pixelation,* for example, occurs when resolution is lacking in a display. The interpretation here, however, focuses on the creation of man-made things (for example, software, tools, and computers).

Nobel Prize winner Herbert Simon wrote *The Sciences of the Artificial* 46 years ago. The third edition, published almost exactly 20 years ago,[a] still holds a great deal of meaning today. Computers and computer programs are *artifacts*. That is, they are man-made constructs and, in theory, they should be more understandable than natural phenomena for which there are no guaranteed laws or rules, only approximations that help us understand or predict behavior.

Computers and, especially, computer programs may be among the least well understood artifacts known to mankind. Computers themselves may have sufficiently constrained behaviors that we may be fairly confident as to how they behave (instruction sets, I/O behavior). Linking them in networks may make their aggregate behavior less clear but that is partly because this behavior is derived from *software* that seems to have no serious constraints on the variations it can manifest. Moreover, most software today is mapped from some programming language into machine language, so the mapping may not always produce results we can anticipate even if the program appears to be straightforward as expressed in its higher-level form.

Making things even more complicated, most software runs in the context of other software such as an operating system that mediates access to the resources of the computer. In a networked environment, programs that express *protocols* for communication may themselves produce anomalous (that is, unexpected) behavior because of our inability to adequately analyze the potential state space the protocols can generate. When two computers interact over the Internet, they can be thought of as two bags full of varied software that might never have interacted before and thus produce unpredictable results. When all these interactions are mixed together in the global Internet, it just seems as if there are an uncountable number of ways in which things might unfold.

If we care about predictable behavior or at least behavior that might be constrained into a predictable envelope, we are going to have to think hard about *design*. Design is one way to think about bounding behavior—"constrained by design" seems like an attractive phrase.

Users are generally not too fond of unpredictable behavior. Perhaps there are some forms of entertainment in which unpredictability has value, but for programs we rely on daily, most of us would prefer they work as advertised, for the user interfaces to remain relatively stable, and to be alerted if it appears things are not going to unfold as expected.

The more I think about software *usability*, the more I am inclined to focus on the underlying design of the software and, especially, the means by which users can communicate their functional desires to exercise that software. There are times when using so-called computer tools such as text processing programs that I wish I could go back to typewriters that put the keys where I position the paper rather than trying to coerce an uncooperative text processing program to center, indent, or place text where I want it on the page. Of course, I would lose an enormous amount of flexibility including the possibility of adjusting text font and size automatically or reformatting the document on the fly.

For any significant piece of software, design for usability takes a good deal of thought, especially when considering how its functionality is to be made apparent to a user with a visual, aural, or motor problem requiring the user interface to adapt or be adaptable to accommodate. It seems to me these considerations often do not get the attention they should in the early stages of program/application design. Stanford University has the *d.school*, which is shorthand for the Institute of Design. There are similar initiatives elsewhere; I recently discovered a related program at UC San Diego.

If we are really serious about making software accessible and usable, we are going to have to recognize this is a problem related to the design of artifacts and that our intuitions about design must be rooted in real knowledge of how software is used by people needing assistive consideration. Testing of interface ideas by people using assistive techniques regularly seems a critical part of solving this problem. Blindfolding an otherwise sighted programmer and asking her to try out her code does not produce an experience adequate to the task (although it may well be, er, eye-opening).

Just as Herb Simon recognized so many decades ago, we really need to derive deep understanding of the nature and behavior of the artifacts we create before we can successfully design them to perform in predictable or at least manageable ways. C

a Hardcover ISBN: 9780262193740 (Sept. 1996), Paperback ISBN: 9780262691918 (Sept. 1996)

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

DOI:10.1145/2809694

May the Computational Force Be with You

There are many open questions in complexity theory concerning "a deeper reality below the one we perceive" that may be relevant, even fundamental, to Yannis Papakonstantinou's Viewpoint "Created Computed Universe" (June 2015). Suppose our own familiar universe is indeed a simulation and not the best of all possible universes at that. If the exponential time hypothesis is true, the entity running the simulation might defend itself against ethics charges of simulating a non-optimal universe by pointing out it cannot find the optimal parameters for such a universe without effectively running a large number of simulations, most of which would be launched with non-optimal parameters.

Now suppose the exponential time hypothesis is false. An entity wishing to find a reasonably accurate value for the probability that the Allies would win World War II, over some given probability distribution for the initial state, could use approximate counting and a sub-exponential SAT solver to compute an approximation to this probability in a time much shorter than would be required to run enough simulations to gain the same amount of accuracy, perhaps under the assumption there are good local approximations to the simulation outcome in some sets of parameter options. It could then argue it was working at a more abstract level than such simulations that might be regarded as gratuitously violent to their simulated occupants, as they were not necessary to solve the problem of calculating the probability of victory.

A.G. McDowell, Chippenham, England

Let Programmers Copy Code Between Languages

In his letter to the editor "When Software Is Not Protected by Copyright" (June 2015) on Pamela Samuelson's Viewpoint "Copyrightability of Java APIs Revisited" (Mar. 2015), Marc K. Temin said, "...the Java APIs were new, and their creation was not dictated by compatibility requirements of other programs." While this assertion was true in one sense, as compatibility with programs was not required, the Java APIs were in fact partially designed to match standard C libraries; for example, according to Java documentation, the sole method, `compareTo`, of a Java interface called `Comparable` returns "a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object." The manual page for the `qsort` function (the function was defined in the sixth edition of Unix, released May 1975) said the last argument to `qsort` is a pointer to a function that returns "an integer less than, equal to, or greater than 0 according as the first argument is to be considered less than, equal to, or greater than the second."

The Java method `Math.atan2` returns the arctangent of its first argument *y* divided by its second argument *x*, the same calling convention used by the C library function `atan2`. The calling convention used by C is not the most natural one because the C function is intended for conversions from rectangular coordinates to polar coordinates. It would have been more natural in retrospect for *x* to be the first argument, not the second, due to the traditional ordered-pair notation for points in a two-dimensional space. Yet those who designed the corresponding Java class chose compatibility as a more important goal than correcting what is arguably a suboptimal option.

Meanwhile, the API documentation for the Java class `Formatter` says, "Formatted printing for the Java language is heavily inspired by C's `printf`," a statement not surprising to those who were at Sun Microsystems when Java was first released in 1995. Nearly every programmer at Sun used C. Keeping similar interfaces consistent between the libraries of the two languages reduces the learning curve, which was, and still is, important for gaining programmer acceptance. Consistent interfaces almost certainly reduce the number of bugs resulting from programmers switching between the two languages many times per day, perhaps due to using the Java Native Interface. It is all too easy to inadvertently use the other language's calling conventions. Regardless of legalities, reducing bugs delivers enormous benefit to society by making software more reliable and cheaper to develop. If copyright law does not allow such copying (where the main goal is to reduce complexity for programmers), then the law on software copyright as applied to APIs should be changed.

William T. Zaumen, Palo Alto, CA

Why Indian Female CS Representation Is Different

I hope Roli Varma's and Deepak Kapur's article "Decoding Femininity in Computer Science in India" (May 2015) opens a wider discussion on women's participation in CS education and career development, which is generally U.S.-centric. As someone who has worked in the IT industry, been involved in campus recruitment, taught CS undergrads, and counts many relatives pursuing the CS discipline, I would say underrepresentation of women in CS is nation- and society-specific, and certainly not universal.

Since 2000, the Indian software industry has grown significantly, and the local need for CS professionals is strong. Large companies (such as Infosys, TCS, and Wipro) continue to recruit thousands at the entry level. Other sectors of the Indian economy are nowhere near in terms of job opportunities for fresh graduates. At the same time, opportunities for studying medicine, a traditional goal for high-achieving female students in India, remains restricted and costly. Female students who achieve academically at the high school level thus increasingly opt for CS undergraduate programs.

Most Indian families do not know what constitutes a CS or IT job but assume it pays well and takes place at a desk in a nice environment and IT professionals get to travel overseas, thus influencing university students' options in case they choose engineering.

Before the software boom, girls who aimed for engineering, mainly preferred electronics or its related branches. Few girls opted for civil or mechanical engineering, as these disciplines are viewed by Indian families as leading to a physically demanding job generally not confined to a fixed schedule.

Indian universities have seen the ratio of boys and girls in undergraduate CS classes at around 60 to 40, with the same ratio continuing at the entry level in IT companies. In staffing, with more than four years' service in India-based IT companies, women show less representation, as many of them sacrifice a potential career for family.

India sees notable representation of female students on the merit list in high school exams and even in undergraduate engineering exams. Men dominate staffing in certain roles, including system architects, software engineers, and database administrators, that require deep knowledge and understanding of CS. With intake in entry-level IT positions is trending downward today and layoffs seen for the first time in years, female students are again tilting toward electronics.

Researchers should look into female student representation in CS education in countries that are similar to India. They should also compare representation of women at senior levels and in deep technology roles in CS and IT in the U.S. with other countries. Over a long career, I have seen IT education and jobs in India continue to emphasize performance, with no caste factor in any discussion, evaluation, appointment, or achievement.

NK Mehta, Bangalore, India

When Enough Is More Than Good Enough

Logan Kugler's news story "Is 'Good Enough' Computing Good Enough?" (May 2015) reminded me that such a heuristic approach to reasoning and computing is fundamental in software development. As young students of computer science and software engineering, we were taught the "absolute perfection" of our work products is an elusive goal due to the fact the development process of any meaningful algorithmic solution is both iterative and optimizing.

Although seemingly amateurish, iterative refinement of "Hello World" can actually produce a useful product. But client needs evolve, affecting the desired baseline of product features. Even when a software product's change manager says, "No more!," thus fixing the requirements, what the client ultimately needs and wants is inconsistent with what developers can deliver during a given development cycle. Although verified and validated based on an agreed feature set, the work product is inevitably less than perfect.

"Good enough?" is a question all developers should ask themselves when faced with an eternal stream of demands. They must prioritize, assess delivery requirements, and estimate the effort associated with each activity. When overwhelmed with requests, they should "punt" on some tasks and accept the consequences of not completing them. This "Aw, forget it!," or AFI, algorithm, reducing one's work burden, is necessary for preventing developer overload. When the flow of tasks is arithmetic, logic-unit computations, or Web-server requests, quality-of-service requirements might not ensure each instruction or query is processed.

The topic of computing "just when needed," or JWN, arose during an Omaha-area Java User's Group I attended earlier this year, spawned by a discussion about processing in the reactive-programming model for Web transactions. The group accepted that any server request, accompanied by a "performance contract" saying how long users expect a Web server would work on it, could let the server decide whether to ignore it. This workload-limiting alternative could greatly reduce a Web server's burden during peak traffic periods. A process in a low-priority thread could maintain the needed meta-information, including the average work and wait times of available services, for making these decisions. The parallels with work Kugler described are interesting in that the goal of processing is not "perfect" but "acceptable" performance.

Considering that research is really "re-search" and "re-evaluation" of existing ideas, the AFI and JWN concepts are indeed nothing new. Developers with similar academic training draw common conclusions, and most research is really only a rehashing of old, fundamental, abstract concepts. As a superset of AFI and JWN, the "good enough?" concept is archetypal in all areas of processing. It reduces energy consumption in a server's circuits, as well as in the brain's organic neural networks. Saying "Good enough!" not only means it exceeds "good," it even approaches "perfect completion."

Jody Sharpe, Omaha, NE

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.

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:10.1145/2800200 http://cacm.acm.org/blogs/blog-cacm

Moving Beyond the Cold War

John Arquilla assesses the need for cyber arms control.

John Arquilla
"From Russia, Without Love"
http://bit.ly/1FuuYlX
June 3, 2015

Though hardly as sexy-sounding as SMERSH, the Stalin-era Soviet counterintelligence organization—much puffed up by Ian Fleming in his James Bond novels—the designator "APT28" may refer to a Russian, or at least Moscow-linked, outfit with truly mass disruptive capabilities in cyberspace. SMERSH was an acronym for smyert shpionam (roughly, "death to spies"), while APT refers to the "advanced, persistent threat" that the FireEye cyber security firm has studied so closely—in particular its activities before and during the Russo-Ukrainian conflict. And what is known thus far about APT28 is most troubling.

In Ukraine, the group—its capabilities are clearly beyond the capacity of any individual—has launched disruptive attacks on critical infrastructure, as well as on Internet connectivity. Government and military communications have also been targeted. The key tool—but hardly the only one—is Uroburos, malware that infects and takes over machines, exfiltrates sensitive data, passes false commands, and causes shutdowns. Interestingly, the scope, scale, and specific targeting engaged in against Ukraine mirror quite closely the patterns of cyber attack witnessed during the Russo-Georgian War of 2008.

Russia is alleged to be involved in a range of other recent high-profile cyber incidents as well. Last October, the White House admitted its information systems had been hacked—with Moscow-friendly perpetrators being the prime suspects. And in January of this year, the pro-Russian group Cyber Berkut ("special police") got into German Chancellor Angela Merkel's and other German government sites, disrupting them and leaving messages criticizing German support for the Kiev government of Ukraine. These hacktivist incidents were in some respects like the cyber attacks on Estonia in 2007—but that earlier case was more far-ranging, mounting costly strikes against banking and other economic targets as well.

And it is in the realm of economic cyber war that the Russians have been concentrating of late, according to Kevin Mandia, founder of the cyber security firm Mandiant. Just before the opening of the G20 summit meeting in Brisbane, Australia last November, Mandia made the point explicitly that the Russian government was "actively condoning cyber attacks on Western retail and banking businesses." Expanding on his remarks, Mandia made clear his conclusion was based on the inference that, given strict controls over cyberspace-based activities in and from Russia, it would stretch the limits of credulity to believe Moscow was unaware of these hacks.

Circumstantial evidence abounds as well—but as yet there is nothing that would stand up in a criminal court as proving Russian culpability beyond a reasonable doubt. And so a form of covert cyber warfare, with costly economic implications, continues to unfold. Much as groups of pro-Russian insurgents—including, it seems, many actual Russian soldiers—have been fighting in eastern Ukraine, with Moscow all the while denying involvement. We live now in an age of shadow wars, real and virtual.

It is thought the Russians were engaging in covert cyber action of a very sophisticated sort as far back as the spring of 1998, when an extended series of deep intrusions into U.S. defense information systems began—and continued, the public record suggests, for quite some time. I was involved in some of the investigation into this matter, which was initially labeled "Moonlight Maze," and so can only refer readers to what has been openly reported. But the key point here is there was a strong sense, even back then when the Russians were still reeling from the effects of the dissolution of the Soviet Union, that their cyber capabilities were quite substantial.

It is ironic that, just a few years prior to the Moonlight Maze episode, the Russians asked for a meeting to be

held between a few of their top cyber people and some from the U.S. The ostensible idea being to establish a process for avoiding hostile "incidents in cyberspace." The Russian delegation was headed by a four-star admiral who probably had in mind the analogy with preventing "incidents at sea." I was on the American team, and found the Russians' ideas quite sensible—including their call to consider the possibility of crafting behavior-based forms of arms control in cyberspace. There is no way to prevent the spread of information technology that can be used for cyber warfare, but there can be agreement not to use these capabilities aggressively, or against civilian targets, etc. The Russian idea was akin to the controls that exist today over chemical and biological weapons—many nations can make them, but virtually all covenant never to use them.

I was very taken by the idea of cyber arms control, and recommended in my report on the meeting that the U.S. pursue these talks further. My recommendation was, to put it mildly, hooted down in the Pentagon, where the view was that the Russians were afraid of us and were just trying to buy time because they were so far behind. Well, that was nearly 20 years ago. If the Russians were behind then—and I sincerely doubt it—they certainly are not now. Indeed, Russia is a cyber power of the first order, like China and a few others. And, like China, its cyber security capabilities are quite robust—much more so than American cyber defenses.

Ever since that fateful first Russo-American meeting of cyber specialists, the Russians have occasionally raised the prospect of cyber arms control at the United Nations. We have routinely rebuffed such suggestions. But perhaps it is time, all these years later, to reconsider the possibility of behavior-based agreements to limit cyberwar. After all, by now we know for sure the Russians are not making such proposals out of fear.

Reader's Comments

After reading your article all I wanted is to go back to Russia and hate U.S., U.N., and others. Because they hate us (your article is proof). Your article does not increase peace in the world, does not critically analyze the situation (others do not cyber-attack?). It simply states the now-trending "Russia is bad." Even worse, the title implies if I am from Russia, then I am without love to you. Really sad to read this in the scientific journal.

—Ayrat K.

I am appalled to see such a propaganda piece in a reputable scientific journal like Communications. *I am reading* Communications *to get facts and science, but here I see only speculation, hearsay, and general FUD.*

The most troubling feature of the article is that it is apparently was written by a U.S. citizen—a country that is believed to be behind the most sophisticated cyber weapons code in the world, the country that is believed to be the only one that is known [so far] to cause a real-world damage to the other country via a cyber-attack without a formal declaration of war. I highly recommend everyone who is interested in facts about cyber-attacks to learn about Stuxnet: http://en.wikipedia.org/wiki/Stuxnet. Time will be much better spent.

I urge the editors of Communications *not to keep silent about the cyber-war and cyber-crime, but to do a real and in-depth scientific study. For example, what the experts on the software ethics would say about something like the highly sophisticated code of the "Equation Group" and its nefarious intent? Is the fact that there is, undoubtedly, a huge team of highly skilled software developers who write such code, a failure of the modern education system to include ethics into the education system, or is it an unfortunate side-effect of any scientific and engineering progress? Do the people who write it understand the evil they bring onto the world or do they "just follow the orders"?*

As a person who has dedicated his life to use his software engineering knowledge to make this world a better place, I am greatly disturbed by this.

—Roman Elizarov

It is hard to accept the "propaganda" charge, as the article makes the key point that Russia has long wanted to pursue cyber arms control—something the U.S. has blocked. That Moscow now has such superb cyber warfare capabilities—and is apparently using them—is in some respects a consequence of American intransigence in the area of behavior-based cyber arms control.

—John Arquilla

Science | DOI:10.1145/2800441 Neil Savage

Split Second

The issue of whether to add a "leap second" to square the clock with the Earth's orbit pits time specialists against IT.

IMAGE BY IWONA USAKIEWICZ/ANDRIJ BORYS ASSOCIATES

THE QUESTION, "WHAT time is it?" seems simple on its face, but it is a question that delegates to the World Radiocommunication Conference will struggle with when they meet in Geneva this November. At issue is whether to keep the world's definition of time as it is, or to alter it by removing the concept of a leap second.

On 26 occasions since 1972, clocks around the world have stopped for one second to adjust for the difference between the Earth's rotation and Coordinated Universal Time (UTC, under the French acronym). Those adjustments mean the sun remains over the Prime Meridian at noon, but they also cause a host of problems for computers worldwide. For example, the leap second that happened on June 30, 2012 crashed the software powering airline reservation systems in Australia, and also led to issues with services including Reddit, Foursquare, and Yelp.

Many IT experts would like to do away with the leap second altogether, or adjust how it is implemented so their computers can deal with it more readily. Whether the International Telecommunications Union (ITU), which governs UTC, will actually do that in November, however, is anybody's guess. The ITU has been considering the question since 2005 without resolving it.

"The fundamental problem is there's no way of making everybody happy," says Judah Levine, a physicist in the Time and Frequency Division at the National Institute of Standards and Technology, which maintains and distributes standard time signals for the U.S., but takes no official position on the leap second debate. "Pretty much the entire timing community wants to get rid of the leap second, and the reason is it's a major hassle."

Ron Beard, head of the positioning, navigation, and timing branch of the U.S. Naval Research Laboratory and international chairman of the ITU working group that covers time signal services, says more devices and systems are relying on knowing the precise time than ever before. "If you have to stop or change the clocks one second at the same moment in the entire world, that's a big step," he says. "It's not a trivial matter."

Leap seconds introduce a discontinuity that can throw computers off-kilter. Generally, a leap second is added at midnight at the end of either June or December. When a clock gets to 23 hours, 59 minutes, and 59 seconds, it does not roll over into the next day, but repeats that last second, which

can prove confusing to algorithms. Suddenly it may be impossible to tell which of two events came first, causing programs to send up error flags or simply stop working. "When you have two consecutive seconds with the same name, you lose track of causality during that time," Levine says. "During the leap second, if I tell you it's 23:59:59, you don't know if it's the first version or the second version."

A leap second is such a rare event, Levine says, that it is often done incorrectly—a program may add a second when it is not supposed to, or not add it when it is, or subtract a second instead, leading to separate systems disagreeing about what time it really is. The problems such disagreements might cause are unpredictable.

"There are a number of factories that shut down across leap seconds because they don't want a valve to turn on or turn off" at the wrong time, says Warner Losh, an independent software consultant who has worked on projects involving high-precision timing. He says pharmaceutical companies, for instance, worry that a timing glitch could lead to too much or too little of an ingredient being added to a batch of drugs, or the temperature of a process being set incorrectly. Errors can arise in other industries as well. A power fluctuation—unrelated to leap seconds—hit a Toshiba plant making flash memory chips in 2010. It lasted only 70 milliseconds, but caused the company to scrap 20% of its products.

Poul-Henning Kamp, a software developer in Denmark, worries a glitch might even prove serious enough to get someone killed. "I believe that is only a matter of time," he says. "More and more systems are integrated closer and closer, including life-support medical machinery."

One reason leap seconds are such a headache is that they are rare and unpredictable. They are decided upon by another standards body, the International Earth Rotation and Reference Systems Service (IERS). Sloshing tides acts as a brake on the Earth's rotation, and weather, earthquakes, even melting glaciers, also come into play. When the IERS determines the planet is falling behind UTC, it issues the announcement that a leap second will be added six months hence. The latest leap second was inserted on June 30, just before this story was written.

A workaround Google adopted before the 2012 adjustment is to skip the leap second altogether and instead stretch the seconds leading up to it by a few milliseconds.

Six months, Losh complains, gives IT people too little warning; they can test their programs, but there could be a rare routine or a spare that has been out of use that gets skipped. "Since leap seconds are a rather exceptional thing, that code isn't tested very often," Losh says. "It's very easy to insert problems that don't show up until a leap second happens."

Though leap seconds are generally added every 12 to 18 months, there was a seven-year period from 1998 to 2005 with no leap seconds. Even that caused a problem, when Motorola's Oncore GPS receivers decided that 256 weeks was long enough without a leap second and added one in November 2003 that nobody was expecting.

One solution might be to schedule leap seconds at regular intervals, so programmers would know of them well in advance, though that would mean the planet would be out of sync with UTC for some period. Another would be to wait much longer, then add a leap minute. No serious proposal to take such steps has been put forward, Levine says.

A workaround, which Google adopted before the 2012 adjustment, is to skip the leap second altogether and instead stretch the seconds leading up to it by a few milliseconds. Most systems can tolerate variations of a couple of hundred milliseconds without problems. Once the clocks reach midnight and the leap second passes, the computers and UTC are again in sync

ACM Member News

STRIVING TO MAKE STAR WARS TECH REAL

Zhengyou Zhang, principal researcher and manager of the Multimedia, Interaction and Communication (MIC) Group at Microsoft Research in Redmond, WA, received the IEEE Helmholtz Test of Time Award in 2013 for his 1999 paper "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations" (http://bit.ly/1KEQVlD). The paper, which defined a flexible two-dimensional camera calibration technique that estimates motion by requiring the camera "to observe only a planar pattern shown at a small number of different orientations," Zhang says, has garnered over 8,730 citations in the 15 years since its publication and the technique it describes is widely deployed (Intel and the U.S. National Aeronautics and Space Administration use Zhang's technology in the Mars rovers).

A dual citizen of France and the U.S., Zhang was born and raised near Shanghai, China. He received his undergraduate degree in electronic engineering from Zhejiang University before moving to France, where he received a master's degree in computer science, artificial intelligence, and speech recognition from the University of Nancy, and a Ph.D. in computer vision from the University of Paris-Sud.

Zhang says his parents in China inspire his research and inventions. "I'm always part of my own products and projects," Zhang says. "It serves my needs and helps other people."

His current Microsoft research involves immersive human-human telepresence, augmented reality, and multimedia technology. "Ideally, people will interact with each other as though they're face to face," Zhang says. "We want to bridge the real world and the digital world and make Star Wars technology a reality."

—Laura DiDio

without the discontinuity caused by repeating a second, but that can bring up other issues.

A second is legally defined in terms of the number of oscillations of a cesium atom in an atomic clock, and those are the seconds the ITU uses in universal time. For financial institutions that have legal obligations to timestamp documents accurately, using non-regulation seconds could get them in trouble. Additionally, some scientists use time servers, such as those maintained by NIST, to measure intervals in their experiments. To ensure experiments are repeatable, they have to know the seconds they use on one day are the same length as those on another day.

So why keep the leap second? "The resistance is mostly from, I would call them traditionalists, who feel it would be a travesty to have the sun not be on the meridian at noon," says Dennis McCarthy, retired director of time at the U.S. Naval Observatory. Just as leap days were added to the Gregorian calendar to keep the equinoxes and solstices from drifting out of their accustomed months, these traditionalists argue, so leap seconds are needed to keep the sun at the right place in the sky. "I think that's just theater," Levine says of that argument. "The rate of offset is about a minute per century, so it's going to be a long time before it's midnight at noon." Levine believes national pride, particularly British attachment to Greenwich Mean Time, which currently matches UTC, plays a role.

On the other hand, Ken Seidelmann, former director of astrometry at the U.S. Naval Observatory and an astronomy professor at the University of Virginia, says there could be religious, cultural, legal, and financial implications of redefining UTC that have not been considered. "I think it is unprecedented to eliminate a time scale that is needed, just because it is inconvenient," he says. "Once mean solar time is abandoned, how would you return to it?"

He points out that the Global Positioning System uses International Atomic Time, which does not include a leap second, but does provide offset information so that systems receiving GPS signals can translate to UTC. People who need it could simply use atomic time, he argues, while leaving UTC untouched.

McCarthy chaired a working group of the International Astronomical Union (IAU), created to see if that body wanted to take an official position on the leap second. Some astronomers argue that, if official time and the position of the Earth drift apart, it becomes more difficult to know where to point their instruments, though McCarthy says making an adjustment is fairly easy. Ultimately, the IAU decided not to take a stance. "The astronomical community, outside of a few very vocal individuals, doesn't care," McCarthy says.

He thinks most of the rest of the world will not care much either if the leap second disappears. "Noon" has not really been at "noon" in most places since time zones were introduced to straighten out the confusion in train schedules. Most people seem to adjust to Daylight Saving Time without much difficulty, and China, which is wide enough for five time zones, only has one, and manages to get by, McCarthy says.

He is not sure the leap second will be abolished in November, but he believes its eventual demise is inevitable. "We will indeed do away with the leap second," McCarthy says, "because it will just become too inconvenient to keep doing this." C

Further Reading

Finkleman, D., Allen, S., Seago, J., Seaman, R., and Seidelmann, P.K.
The Future of Time: UTC and the Leap Second, *American Scientist, 99*, 312-319 (2011)

Kamp, P-H
The One-Second War, *Communications of the ACM, 54*, 44-48 (2011)

Nelson, R.A., McCarthy, D.D., Malys, S., Levine, J., Guinot, B., Fliegel, H.F., Beard R.L., and Bartholomew, T.R.
The leap second: its history and possible future, *Metrologia*, 38, 509-529 (2011)

Beard, R.
To abolish or not to abolish the leap second? The past and future of Coordinated Universal Time, *ITU News*, (2013) http://bit.ly/1FdX5t6

Leap Seconds, The UK Public Dialogue http://leapseconds.co.uk/reports-findings-dialogue/

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

Milestones

Computer Science Awards, Appointments

CRA HONORS JAHANIAN, GATES

The Computing Research Association (CRA) recently awarded Farnam Jahanian, vice president for research at Carnegie Mellon University, the 2015 CRA Distinguished Service Award.

Farnam served as U.S. National Science Foundation assistant director for Computer and Information Science and Engineering from 2011 to 2014. He also served as co-chair of the Networking and Information Technology Research and Development (NITRD) subcommittee of the National Science and Technology Council Committee on Technology, providing overall coordination for the research and development activities of 17 government agencies.

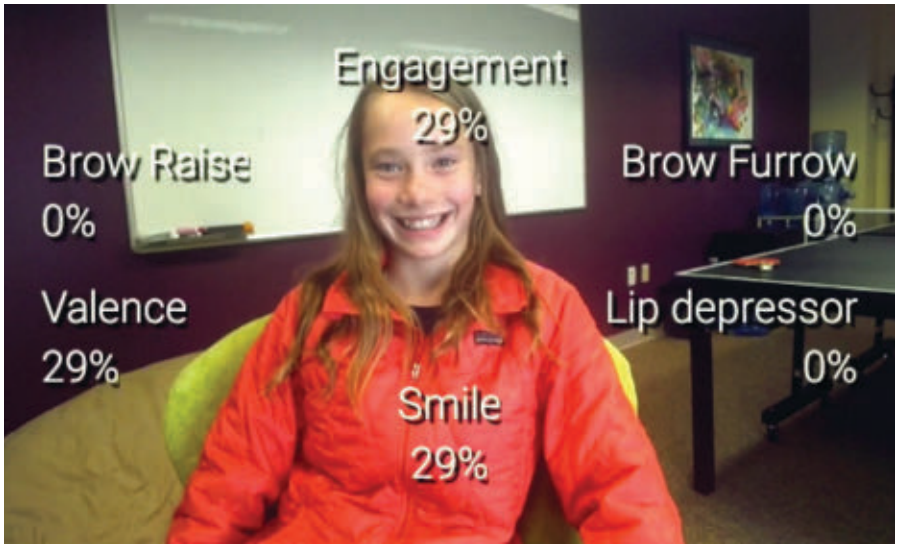
CRA also awarded Ann Quiroz Gates, chair of the department of computer science at the University of Texas at El Paso, the organization's 2015 A. Nico Habermann Award, for "outstanding contributions aimed at increasing the numbers and/or successes of underrepresented groups in the computing research community."

Gates has long been a leader in initiatives that support Hispanics and members of other underrepresented groups in the computing field, leading the Computing Alliance of Hispanic-Serving Institutions, an alliance of 13 institutions whose work has had large and sustained positive impact on recruitment, retention, and advancement of Hispanics in computing, and helping to establish the Affinity Research Group (ARG) model for research mentoring and peer support. Gates has also promoted the recruitment, retention, and advancement of female faculty at her home institution. Gates' influence has extended to other initiatives and communities, including the Society for Advancement of Hispanics/Chicanos and Native Americans in Science (SACNAS), CMD-IT, and the AccessComputing Alliance.

Technology | DOI:10.1145/2800498 Gregory Mone

Sensing Emotions

How computer systems detect the internal emotional states of users.



Quantifying clues to human emotions.

IMAGE COURTESY OF AFFECTIVA.COM

During the last Super Bowl, Emotient, a San Diego, CA-based company, hosted an unusual party. Emotient recruited 30 volunteers to a local bar to watch the game, eat, and drink. As the annual spectacle progressed on two televisions, a camera attached to each flat screen monitored the viewers. Behind the scenes, the Emotient Analytics system identified and tracked each face within the camera's view, then determined the changing emotional state of each individual over time. Whether they were amused, ambivalent, or surprised, the system matched those reactions to what was happening on screen, determining which commercials seemed to bore the viewers and which ones they liked enough to share. "We were able to predict which advertisements were likely to go viral based on facial behavior," says lead scientist Marian Bartlett.

Both the Emotient technology and a similar system from Waltham, MA-based Affectiva are products of the burgeoning field of affective computing, in which researchers are developing systems that estimate the internal emotional state of an individual based on facial expressions, vocal inflections, gestures, or other physiological cues. Affectiva and Emotient are catering to advertising and market research companies, but affective computing stands to impact a number of areas. The technique could lead to online learning systems that notice when a student is growing frustrated with a problem, healthcare applications that measure how a depressed patient is responding to a new medication, or in-home assistance robots that closely monitor the emotional state of the elderly.

The field has been around for two decades; Affectiva co-founder and Massachusetts Institute of Technology professor Rosalind Picard coined the term "affective computing" in the mid-1990s. Yet experts say a combination of improved data, increased processing power, and more robust machine learning techniques has led to significant recent advances. "Affective computing has really exploded in the last five years," says signal processing expert Shrikanth Narayanan of the University of Southern California.

Focusing on Faces

When people communicate with each other, we study each other's vocal intonations, gestures, facial expressions, and posture, each of which gives us some information about the other person's internal state—whether they are engaged, distracted, or annoyed. In some cases, the expressions are contradictory; the fact that a child is smiling, for example, might seem to indicate she is happy. Yet Narayanan's work has shown how computers trained to pick up clues in the pitch, intonation, or rhythm of a child's speech can uncover hidden emotions lurking behind that smile.

This sort of detection is natural for humans; we track multiple cues to guess what someone is really thinking or feeling. "We're implicitly integrating all these pieces of information," Narayanan says. "We're not picking one or the other."

At this point, though, the most popular affective computing systems focus on visual input, and on the face in particular. Until recently, simply recognizing and tracking faces in a crowded, dimly lit environment such as a Super Bowl party was a serious challenge. Early systems could be confused by something as pedestrian as a beard. Today's software can delineate contours in the lips, eyes, nose, and more, and it can do so even in poor lighting conditions, according to Akshay Ashtana, a facial recognition researcher at Seeing Machines in Canberra, Australia.

Emotient and Affectiva benefit from this technology, since it allows them to pick out multiple faces in a single crowded scene, but before they focus their systems on those targets, they have to train them to recognize the different expressions and emotions. Both technologies are based on psychologist Paul Ekman's facial action coding system, or FACS, which details the involuntary muscle movements we generate in response to different stimuli, and how these subtle cues are linked to different emotional states.

Generating Quality Data

Emotient employed trained consultants to produce and analyze hundreds

of thousands of pictures of people spontaneously displaying emotions. Then the company trained its algorithms on even more data. To teach the system to recognize joy, for example, they fed it 100,000 images of people looking and/or feeling joyful, then another million pictures of individuals who were not quite so happy.

The Emotient Analytics technology segments each frame into a grid of 2,304 pixels. Each pixel is then analyzed in terms of its brightness, which can range from zero up to 255. Next, the system searches across that array of pixel values for higher-level features such as edges. This provides the system with what Bartlett calls the dense texture features of the image: a precise measure of the patterns of light and darkness across the entire face. "We end up with a very high-dimensional description," she says.

Once the system breaks down each image, it compares the different bodies of data—joy and not-joy. Then it uses statistical pattern recognition to figure out what joy actually looks like in the raw data. Emotient has used this general approach for years, but more recent advances in its capabilities can be traced in part to its switch to deep learning methods. "Deep learning is much more powerful at being able to pick up patterns in the data," Bartlett says.

Deep Learning

In the past, scientists had to instruct a system to look for specific features. Deep learning analyzes the raw data without such instructions, using a hierarchical approach. First the system searches for patterns at the pixel level. Then it looks across groups of pixels and picks out edges, corners, gradients, and so forth. Once these and other layers are scrutinized, and the emergent patterns are linked to specific emotions, the chance of miscategorizing or failing to recognize an expression drops significantly.

McDuff likens deep learning to the way an infant identifies connections in the real world. When babies start to see cars on the road or on the television, they do not necessarily know each of these is called an automobile, but their brains begin to associate the unique combinations of edges and shapes with each other. They start to link all of these images of cars together, even though some are two-dimensional representations on a screen, because they identify shared features at different levels. "That's what our algorithms are doing," says McDuff. "They're looking at the pixel information, and loads and loads of examples, and they're figuring out what types of pixel structures coincide with different facial expressions."

At Emotient, Bartlett says deep learning has increased the system's range. Originally, if a person turned more than 15 degrees to the side, or tilted or rolled back their head by the same amount, the system wouldn't be able to accurately estimate his or her facial actions. With deep learning, however, the system can capture expressions when the individual's face is beyond 30 degrees off-center.

The deep learning approach also scales more effectively than previous methods. Like Emotient, Affectiva has its own dataset of hundreds of thousands of hand-coded images, and this is a huge advantage with deep learning. "As you increase the amount of training data or the number of images you give the algorithm, the performance just continues to improve," McDuff says.

Applications and Multimodal Approaches

The power of this new measurement tool for market research and advertising has not been lost on businesses; both Emotient and Affectiva have signed a number of major clients. Yet affective computing experts inside and outside the commercial sphere are also pressing forward with many other applications. Bartlett, who is also a computer scientist at the University of California, San Diego, has been working with doctors and researchers to improve pain management for children. "There's a significant problem of undermanaged pain in pediatrics," she explains. Children do not always accurately express how they feel, but a facial analytics system could help find the truth. "The idea is to be able to have a system that monitors pain levels the same way you keep track of heart rate and blood pressure."

The number of applications will also expand as researchers incorporate more physiological input, going beyond facial expressions, gestures, voice, and other more obvious cues. At the University of Notre Dame, computer scientist Sydney D'Mello has conducted studies that gauge a person's emotional state based on the frequency and intensity of their keystrokes during an online learning session. McDuff has shown it is possible to gain insights into someone's emotions based on who they text, email, or call during the day, and how frequently they do so. Other scientists, such as Robert Jenke at the University of Munich, have conducted studies trying to pinpoint a subject's true emotions using electroencephelogram (EEG) electrodes.

Eventually, scientists hope the technology will incorporate multiple sensor input streams—voice, audio, gesture—to create a more accurate picture of a person's internal emotional state. This added data would be especially valuable when evaluating people from different parts of the world. In some cultures, for example, social norms dictate that people mask their emotions when in groups. Estimating the emotional state of such individuals based on visual feedback could be misleading. For now, using multiple input streams remains a major challenge, but humans stand as proof of the potential benefits. "Computers are still miles away from what a human can do in terms of intuition and understanding emotion," says McDuff. "Computers still just have the understanding of babies."

Further Reading

Picard, R.W.
Affective Computing, The MIT Press, 2000.

Calvo, R.A., D'Mello, S.K., Gratch, J., and Kappas, A. (Eds.)
The Oxford Handbook of Affective Computing, Oxford University Press, 2015.

Bartlett, M., Littlewort, G., Frank, M., and Lee, K.
Automated Detection of Deceptive Facial Expressions of Pain, *Current Biology*, 2014.

Narayanan, S., et. al.
Analysis of Emotion Recognition Using Facial Expressions, Speech and Multimodal Information, *Proceedings of the International Conference on Multimodal Interfaces*, 2004.

Emotient
https://vimeo.com/67741811

Gregory Mone is a Boston, MA-based science writer and children's novelist.

Society | DOI:10.1145/2800445 Logan Kugler

New News Aggregator Apps

How apps like Inkl and SmartNews are overcoming the challenges of aggregation to win over content publishers and users alike.

CEO Gautam Mishra describes news app Inkl as the "Spotify of news."

IMAGE BY WAYNE TAYLOR/FAIRFAX MEDIA

NEWS AGGREGATORS ARE services that pull together online content in one place for ease of viewing on mobile devices. Until recently, the market for news aggregator apps has been dominated by Google, with its Google News & Weather (http://bit.ly/1za0Ycd) for personalized news and weather content, and Google Play Newsstand (http://bit.ly/1IFTeav) for personalized curated content and topic-based collections of articles. Yahoo News Digest (https://mobile.yahoo.com/newsdigest/) has also been popular, as has Apple Newstand (http://apple.co/1TM43uC), which is being revamped into Apple News .

Other notable players include:

- **Breaking News** (http://www.breakingnews.com/)—Personalized breaking news stories.
- **BuzzFeed** (www.buzzfeed.com/)—Social content for sharing.
- **Facebook Paper** (https://www.facebook.com/paper)—An interactive iPhone app that assembles news and other stories.
- **Flipboard** (https://about.flipboard.com/)—A popular app with user-curated magazines, and 34,000 topics that can be followed.
- **Nuzzel** (http://nuzzel.com/)—Provides stories popular with users' friends on Twitter and Facebook.
- **Vox Media** (www.voxmedia.com)—Available via seven brands/channels: News (Vox.com), Technology and Lifestyle (The Verge), Sports (SB Nation), Gaming (Polygon), Food and Nightlife (Eater), Fashion and Shopping (Racked), and Real Estate and Architecture (Curbed), and many more.

Mobile news also is available via RSS feed services like Feedly, Digg, Latest News & Radion, and GReader.

There also are a number of new players in this arena.

Inkl (www.inkl.com) is an Australian startup, a subscription news service that curates high-quality content from major news outlets using story rankings by the publishers and editors themselves; it achieves a 30% open rate with its daily news email. CEO Gautam Mishra compares Inkl to popular streaming digital media platforms like Netflix, and describes it as the "Spotify of news." Inkl is not yet available in the U.S., but that could change later this year. A tiered fee structure is planned, with a monthly unlimited-access subscription, a prepaid pay-per-article model, and a free version carrying ads.

SmartNews (www.smartnews.com) describes itself as an "addictively simple news app." This free download is ranked No. 1 in both Google Play and Apple's App Store. Downloads have exceeded 10 million, with more than 1 million active users in the U.S. alone, and it now evaluates more than 10 million articles every day. SmartNews has won numerous "App of The Year" and "Best News App" awards. According to CNET, "SmartNews is an excellent free alternative to other popular news apps."

Co-founder and co-CEO Ken Suzuki developed SmartNews to try to entice Tokyo's millions of subway commuters away from playing mindless games on their smartphones into reading news stories. Suzuki believes SmartNews' rivals are not "other news apps," but "mobile games, social media, and the top apps people use on their smartphones every day. We are competing for the overall attention of the user." The beta international edition (ultimately covering more than 150 countries and a growing number of local cities) was launched earlier this year.

Despite the decline of newspapers in recent years, Pew Research Center data (http://pewrsr.ch/1OrUDlA)

SmartNews Leaders (from left) Rich Jaroslavsky, Kaisei Hamamoto, and Ken Suzuki. At right, a screenshot of SmartNews on a mobile handset, showing the colored section tabs.

shows consumers continue to have a large appetite for news, though increasingly in digital form. Indeed, digital media has recently overtaken traditional news sources in terms of trust, according to a global survey by Edelman (http://bit.ly/1I6recK). The Pew research showed news is "part of the explosion of social media and mobile devices, and in a way that could offer opportunity to reach more people with news than ever before."

According to the International News Media Association, "A quarter of the top 20 apps in the news category...are aggregator apps. More and more, publishers are understanding the need to have a presence wherever their readers or potential readers are consuming content" (http://bit.ly/1I6rUij).

Traditional news aggregators always introduce a dilemma to publishers: if users read an article in the app, they will not go to the publisher's site and the publisher will not make money as a result; yet, this procedure is necessary to reach new audiences. This is an issue highlighted by a report in *The New York Times* on the substantial decrease of traffic to publishers' home pages and the subsequent loss of engagement (http://bit.ly/1CB04qM).

Inkl has an unusual approach to this challenge, as users pay for the articles they read, supporting publishers and creating sustainable news. The payback is an ad-free format, which is its main selling point.

SmartNews has its own approach: to get the best publisher content, it created a uniquely publisher-friendly model, ensuring the first click always goes to the publisher's mobile site, so it is able to keep traffic and related revenue. If the user is offline in the readability mode (SmartView), publishers can place an ad at the bottom of that view and maintain the revenue stream.

These approaches appear to be working, as SmartNews already has over 150 publisher channels (see the full list at http://bit.ly/1aJxEUa/) and Inkl's global publisher list is also growing.

"Our algorithm is unique in that it is optimized for discovery and diversity, rather than targeted personalization."

SmartNews senior communications manager Vincent Chang says, "Our machine learning algorithm evaluates 10 million articles and signals a day to uncover the top 20 most interesting and important articles in each category--that matter right now. Our algorithm is unique in that it is optimized for discovery and diversity, instead of targeted personalization. This prevents the 'filter bubble' problem (the same topics become boring over time), ensuring you can always find something fresh and interesting each day."

The SmartNews algorithm looks at more than just social signals and what is trending; it also evaluates many other user behaviors, including whether people finish reading an article, how they share it, and which topics are in the article. The algorithm then uses natural language processing to classify each article into one of the appropriate topic channels (Lifestyle, Business, Sports, Tech, World, U.S., and so on), topics that people are actually reading (not just clicking) and that matter most to them. The algorithm also takes into account what is culturally relevant for each country, as what is trending in one country often differs from what is trending in the next.

SmartNews is designed like a newspaper, a design proven for over a century, with skimmable headlines, minimalist design, and high-quality news content, plus an offline reading mode (SmartView) for times when Internet connectivity is unreliable. The design lets the user see many stories at a glance, and easily find one to read. Looking at the app's reviews, users appear to be impressed with its interface, usability, intuitiveness, and customizability, and to like the efficient browsing and the smooth swiping between tabs.

Unlike most news aggregators that focus on personalization, SmartNews focuses on discovery. The ethos of the company is about helping users expand their interests and encounter new and fascinating things every day, while connecting them to what others are reading to encourage discussion.

The discovery of news is analogous to variety-seeking buying behavior in marketing; reading articles outside your existing preferences is like switching brands. Variety-seeking is

IMAGES COURTESY OF SMARTNEWS

exploration for users, although the risk is that they will experience low satisfaction by trying novel products. SmartNews is designed to introduce users to stories they did not know they wanted to read, rewarding them by widening their horizons without too much additional time cost.

As SmartNews uses state-of-the-art machine learning algorithms instead of human editors to determine which news to capture for its users, the app can evaluate a huge volume of culturally relevant news articles in real time. Chang says, "In the past, this would require building many newsrooms of editors in each country. It allows us to deliver the news much faster and impartially, which is especially important for political news."

Recommending news through content discovery is a decision-making problem involving uncertainty. In probability theory, this is referred to as the multiarmed bandit problem: a trade-off between the exploration of new possibilities for unknown payoff and the exploitation of known possibilities for an expected payoff. In the context of news aggregator apps, this theory relates to the trade-off between the time invested in reading or searching, and the payoff with quality, relevant, and useful information received.

There are ways to quantify the relationship between exploration/exploitation and risk-taking/avoiding for humans. While at IBM, Ricky Takahashi, one of SmartNews's data science and machine learning engineers, researched the prediction of consumer choice under uncertainty.

Most news apps use algorithms optimized for personalization—giving us more and more of what we want, with nothing to challenge our beliefs, enlarge our perception, or give us new ideas—which leads to a circular narrowing of knowledge, or a "filter bubble," a phrase coined by Eli Pariser of Upworthy. In a review of Pariser's book *The Filter Bubble* (http://www.thefilterbubble.com), American writer, consultant, and teacher Clay Shirky says, "Internet firms increasingly show us less of the wide world, locating us in the neighborhood of the familiar. The risk, as Eli Pariser shows, is that each of us may unwittingly come to inhabit a ghetto of one." Indeed, Pew data questions whether the self-selective process, combined with algorithmic feeds, is narrowing the kinds of information to which Americans are exposed.

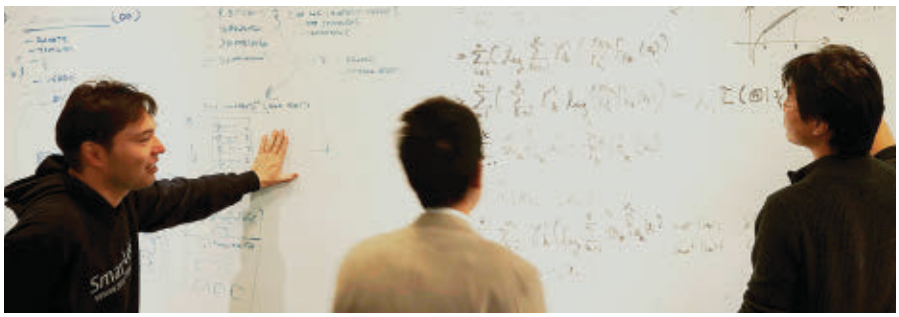
"What people think they are interested in ahead of time and what they are actually interested in can be completely different."

SmartNews is unique in that to avoid the trap of personalization and the filter bubble, it optimizes its algorithms for *serendipity* — those magical "aha" moments when you find something new and amazing, not just things you already follow and know. As Jaroslovsky explains, "what people think they are interested in ahead of time and what they are actually interested in can be completely different."

In addition to the advanced algorithm, the app uses natural language processing to gain greater understanding of articles by processing the nuance in words. The app lets a user tap into the most interesting and important stories immediately. SmartNews co-founder/co-CEO Kaisei Hamamoto says, "Our machine learning algorithm gets smarter every day, learning from millions of users what stories really matter to humanity—not just what your friends are reading."

By helping mobile audiences "discover" new articles, SmartNews helps publishers discover new audiences that may not have otherwise found them. The company wants to expand its news discovery algorithm to make it even better at delivering the most culturally relevant and interesting stories tailored to each country.

As the mobile news industry engages new users, novel ways to package news content are being explored; Apple, for example, recently announced a revamped News app as part of iOS 9 that is a little like Flipboard. The competition for the mobile news market is ramping up, which can only be a good thing for consumers and publishers alike.



Machine learning at work.

References

Lakkaraju, H., Rai, A., and Merugu, S. (2001). **Smart news feeds for social networks using scalable joint latent factor models.** ***Proceedings of the 20th international conference companion on World Wide Web.*** **ACM. http://researcher.watson.ibm.com/researcher/files/in-klakkara/smartnewsfeeds1.pdf**

May, B.C., Korda, N., Lee, A., and Leslie, D.S. (2012). **Optimistic Bayesian sampling in contextual-bandit problems.** ***Journal of Machine Learning Research*****, 13(Jun), 2069–2106. http://www.jmlr.org/papers/volume13/may12a/may12a.pdf**

Pariser, E. (2012). **The filter bubble: What the Internet is hiding from you. Penguin Books. http://www.amazon.com/The-Filter-Bubble-Internet-Hiding/dp/0241954525**

Takahashi, R., and Morimura, T. (2015). **Predicting preference reversals via Gaussian process uncertainty aversion. In** ***Proceedings of the 18th international conference on Artificial Intelligence and Statistics*****, San Diego, CA. http://jmlr.org/proceedings/papers/v38/takahashi15.html**

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

DOI:10.1145/2804228

Thomas Haigh and Mark Priestley

Historical Reflections

Innovators Assemble: Ada Lovelace, Walter Isaacson, and the Superheroines of Computing

Can computing history be both inspiring and accurate?

Consider two recent blockbuster sequels. *Avengers: Age of Ultron*, a superhero movie, enjoyed the second strongest opening weekend of all time, behind only its predecessor, *Avengers Assemble*. The fastest-selling history of computing book ever published is Walter Isaacson's *The Innovators: How a Group of Hackers, Geniuses, and Geeks Created the Digital Revolution*. Its sales fall short only in comparison to his previous book, *Steve Jobs*, which reportedly broke all records for a biography.

Avenging and innovating turn out to have a surprising amount in common. Both require one to assemble a team of superheroes who must work together to defy daunting odds and change the course of human history. Both deploy a cast of characters who have been written about for decades but are now reaching massive audiences. Both feel somewhat overstuffed, as their hugely experienced creators struggle to maintain a light touch while maneuvering a complicated narrative through a vast number of required plot points. Both highlight origin stories, as if understanding the moments at which individuals received their special powers or the circumstances in which particular technologies were first coaxed into operation will always explain their subsequent trajectories.

Isaacson's geek revolutionaries are, for the most part, entrepreneurs rather than academics. People are interested in the men behind the companies behind the gizmos of daily life, particularly if those men became spectacularly rich while exhibiting captivating flaws. Hence the wealth of books and films about Steve Jobs, Bill Gates, Mark Zuckerberg, and the early days of Google. Most of the computer science students featured in these stories dropped out part way through their degrees. Computer science has invested little effort in building and celebrating its own set of heroic role models. Individuals such as Edsger Dijkstra, Donald Knuth, and Alan Kay all have their followings but none have yet inspired a full-length biography, a statue, or a museum. Even John von Neumann has largely slipped from general awareness in the decades since his death. Perhaps computer scientists feel their discipline is doing pretty well without devoting significant energy to the construction and celebration of male heroes. Alan Turing is the exception that proves the rule here, given his gripping personal story, significant contribution to the Second World War, and crossover appeal as a gay martyr.

Isaacson, who has headed both CNN and *Time* Magazine, is one of the world's most successful and best-connected journalists. His titular promises of "geeks" and "genius" signal this is a fairly conservative retelling of computer history, discussing the invention of technologies rather than their commercialization or use. He arranges a series of vignettes along a familiar arc, leading from the Victorian dreams of Charles Babbage through the various computer inventors of the 1940s to the networking pioneers

of the 1960s, the home computer hackers of the 1970s, and a variety of Internet entrepreneurs. Isaacson interviewed famous people to enliven the more recent history, but in his earlier chapters he mines the work of historians.

There are two areas in which Isaacson departs from the classic storyline of popular computing history. Like the Avengers movies, his book suggests superheroes need to collaborate to get the job done. We historians of science and technology often complain that nobody reads our scholarly tomes. Usually we blame this on the unshakable attachment of the public to books about "the lone genius who revolutionized" something or other. Isaacson's subtitle confirms he has no problem with the "genius" part of that, but is rightly critical of the silly idea that his "hackers, geniuses, and geeks" are most innovative when deprived of resources and forced to work alone.

The other distinctive feature of Isaacson's book is his stress on the contribution of women to innovation in computing. He foregrounds a trio of coding superwomen, and centered the promotional campaign for the book on the claim that he was avenging brilliant but defenseless women who had been unjustly erased from history. Avenging is less hospitable to women. *Age of Ultron* was criticized for marginalizing its female characters, despite being directed by Joss Whedon whose *Buffy the Vampire Slayer* broke new ground with its stake-happy heroine and her lesbian witch sidekick. Disney's failure to produce an action figure of Black Widow, the only female Avenger, provoked a chorus of angry tweets.

According to Isaacson, men were good at hardware, but only women were smart enough to appreciate the need for software. The idea that women invented programming is not entirely novel, having been repeated widely enough in recent years to have become the new conventional wisdom. As we write, each of the top 10 hits returned by Googling "first computer programmer" relates to one of three topics. Hits number 1, 4, 6, 7, 8, 9, and 10 are about Ada Lovelace. Number 2 is about Grace Hopper, while number 3 is about the "women of ENIAC." Number 5 is "The Forgotten Female Programmers Who Created Modern Tech," a lengthy interview with Isaacson on NPR's flagship "Morning Edition" radio program in which he hits the trifecta.

Ada Lovelace.

PAINTING BY ALFRED EDWARD CHALON (1840)

Ada Lovelace

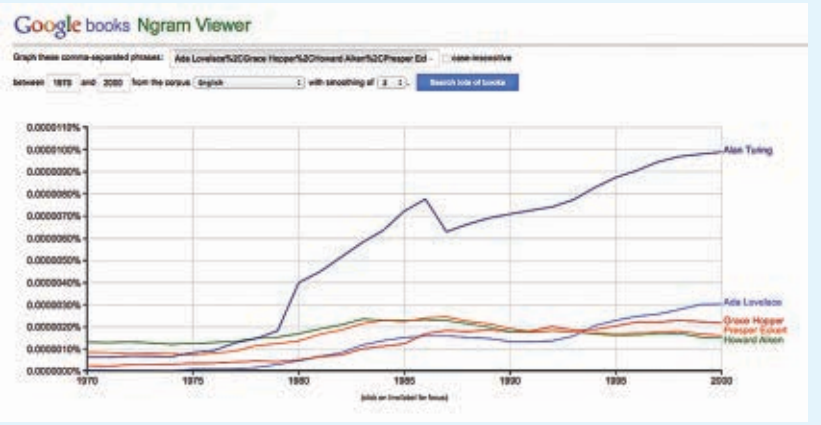
One might reasonably question whether Lovelace needed Isaacson to rescue her from obscurity. He first learned of Lovelace in 2007 when his teenage daughter wrote a college essay about her as the family summered in Aspen, apparently inspired by a "Batman comic book."[a] Isaacson might not have heard of Lovelace, but she was already more famous than any Turing Award winner, having inspired at least a dozen biographers, a major computer language, several novels, a Google Doodle, a comic book series, and a movie. Launched in 2009 by social media consultant Suw Charman-Anderson, Ada Lovelace Day has spread quickly to encompass a host of worldwide events intended to celebrate "the achievements of women in science, technology, engineering, and maths." Lovelace has become a superhero for all of STEM, not just for computing.

a http://nyti.ms/1vAolLZ

Lovelace is remembered for her collaboration with Charles Babbage to publicize the mechanical calculators he designed and attempted to build from the 1820s onward. Babbage relied substantially on the writings of others to promote his machines. The first publication describing the Analytical Engine, intended to automatically execute a sequence of mathematical operations, was an 1842 article by the Italian Luigi Menabrea, written after Babbage's visit to Turin in 1840. Lovelace translated Menabrea's French-language original, adding seven explanatory "Notes" that were more than twice the length of Menabrea's original text.[b] These included a detailed discussion of how to use the Analytical Engine to calculate the Bernoulli numbers. This work, summarized in a large table describing the calculation, has long been understood as a computer program *avant la lettre*.[c] That made Lovelace "the first programmer," an identification cemented in the 1980s when the language ADA was named in her honor.

Google's N-gram tool, based on a massive full-text database of English-language books, suggests that by the mid-1990s Lovelace's fame had already outstripped those of computer-builders such as Presper Eckert and Howard Aiken. Alan Turing, whose story is similarly romantic, enjoyed a still sharper climb in public awareness (see Figure 1).

Figure 1. Google's N-gram tool view: Howard Aiken, Presper Eckert, Grace Hopper, Ada Lovelace, and Alan Turing.



Lovelace has been celebrated as much for her visionary asides as her mathematical accomplishments. She imagined a version of the Analytical Engine that would allow mechanical representation of "the fundamental relations of pitched sounds in the science of harmony" and "compose elaborate and scientific pieces of music." These remarks carry an undeniable frisson of prescience, leading science writer and television presenter Steven Johnson to the idea that Lovelace was a "time travelling" inventor who, like Leonardo da Vinci and Charles Darwin, somehow existed outside her own time. According to Johnson, "her footnote opened up a conceptual space" eventually filled by "Google queries, electronic music, iTunes, hypertext, Pixar."[d] One recent book, *A Female Genius: How Ada Lovelace, Lord Byron's Daughter, Started the Computer Age*, claims that Lovelace "foresaw the digitization of music as CDs."[8] In a promotional interview for the fashion section of the *New York Times*, Isaacson claimed "Ada Lovelace defined the digital age," arguing with a wave that seemed to "encompass all of Silicon Valley and the techies sitting around us" that "If it wasn't for Ada Lovelace, there's a chance that none of this would even exist." This hand waving is not untypical, as is the failure to distinguish between imagining something and causing that thing to come about.

Isaacson acknowledges that the extent of Lovelace's contribution to Babbage's overall project has been energetically debated. Researchers have minutely parsed the surviving records to build or challenge the case for Lovelace as a mathematical genius or as an essential contributor to Babbage's project. The progress of work on the Bernoulli example can be followed in the correspondence between the pair, complicating the popular idea of a neat division of labor in which Lovelace worked out how to use the hardware Babbage invented. Isaacson steers a middle course here, and as a result has been criticized for suggesting Lovelace was "never the great mathematician that her canonizers claim," thus denying her admittance to the club of "genius" promised in his subtitle and perhaps undercutting his own vague claims for her significance.[e]

We feel this squabbling over authorship misses the bigger point. What does it mean for Isaacson to call Lovelace an essential precondition to the existence of today's tech world?

b "Sketch of the Analytical Engine invented by Charles Babbage Esq. By L.F. Menabrea, of Turin, officer of the Military Engineers, with notes upon the memoir by the translator," *Taylor's Scientific Memoirs*, Vol. 3 (1843). Menabrea's original article is reprinted alongside Lovelace's translation and notes in *The Works of Charles Babbage, Volume 3: The Analytical Engine and Mechanical Notation*, M. Campbell-Kelly, Ed. (William Pickering, London, 1989); see http://bit.ly/18FbNZL.

c For example, in one of the earliest popular accounts of computers, *Faster Than Thought* (B.V. Bowden, Ed., Pitman, 1953). This book included a reprint of the text of the Notes, and its frontispiece was a portrait of Lovelace herself. In 1984, Dorothy Stein reproduced part of the table with the caption "The diagram or program for the computation of the first seven Bernoulli numbers" (D.K. Stein, "Lady Lovelace's Notes: Technical Text and Cultural Context," *Victorian Studies 28*, 1 (1984)), and according to Benjamin Woolley, "It is this table which is used to justify the claim that Ada was the world's first computer programmer" (B. Woolley, *The Bride of Science: Romance, Reason and Byron's Daughter* (Macmillan, 1999)). Isaacson tells us that Lovelace "figure[d] out in step-by-step detail the workings of what we now call a computer program or algorithm ... described a sequence of operations and then made a chart showing how each would be coded into the machine."

d Johnson, S. The Tech Innovators of the Victorian Age: What the Victorian Computing Pioneers Can Teach Us About Invention—and Time Travelling. *Financial Times*, (Oct. 17, 2014), showcasing material from his 2014 book *How We Got to Now* (Riverhead Books).

e http://bit.ly/1J4hHSh

Logically, it means those who created later technologies drew directly on her work with Babbage, and indeed that without them nobody would ever have thought of programming a computer. For example, Isaacson explained during his NPR interview that "Babbage's machine was never built. But his designs and Lovelace's notes were read by people building the first computer a century later." However, expert historians have been unable to find any substantial influence of Babbage on the designers of early computers.

Isaccson's claim most likely refers to Howard Aiken, a Harvard University computer pioneer who rhetorically enlisted Babbage as kind of patron saint for his project, since to the best of our knowledge no claim has ever been made that Babbage or Lovelace influenced others such as Konrad Zuse, Tommy Flowers, John V. Atanasoff, J. Presper Eckert, or John W. Mauchly. I. Bernard Cohen, who looked more closely than any other historian at Aiken's work, concluded he became aware of Babbage only after proposing to build his computer and did not learn of Lovelace's work or appreciate the importance of conditional branching for some years later. Cohen discovered that "Aiken was generally ignorant of Babbage's machines" when writing his 1937 proposal to construct what became the Harvard Mark I, having seen only brief accounts, and including "summary (and somewhat erroneous) statements about Babbage's machines." "Aiken," wrote Cohen, "praised Babbage to enhance his own stature" even though "Babbage did not play a major role in the development of Aiken's ideas." Cohen quotes Grace Hopper, who worked closely with Aiken to oversee programming of his computer, as saying she was unaware of Lovelace's work "until 10 or 15 years later."[5]

Turning to the specifics of Lovelace's notes, we challenge the common characterization of Lovelace's tabular presentation of the Bernoulli calculation as a "program." It is not clear what would constitute a program for a computer that was never fully designed, still less built, but neither Lovelace nor Menabrea (whose paper included several smaller examples reportedly supplied to him by Babbage[16]) ever produced a description of a calculation in a form detailed enough to be processed directly by the Analytical Engine. Babbage intended his computer to read and execute instructions one at a time from a deck of "operation cards." The famous table is not a specification for a card deck to compute the Bernoulli numbers. Instead, as Lovelace explained, it "presents a complete simultaneous view of all the successive changes" the various storage units of the Analytical Engine would "pass through" in computing the specific number B_7. When computing other Bernoulli numbers the engine would carry out different sequences of operations, leading to tables with more or fewer rows. The control logic to produce these variations is not present in the table itself. In modern terminology, the table might best be described as a trace of the machine's expected operation.

Considerable further work would be required to turn the calculation described by the table into a set of cards the Analytical Engine could read to calculate *all* the Bernoulli numbers. As well as the operation cards themselves, the control deck would have to include special cards to tell the machine when to "back up" the deck to repeat a sequence of operations. Babbage had not settled on a format for these cards, or even on a mechanism for reading backward. A separate deck of "variable cards" specifying the numbers to be operated on would also be needed, as an operation card could be reused at different stages of a calculation to process different data.

Lovelace was part of her own time, not a visitor from the future or a lone superhero who invented programming and created the modern world. Through Babbage and her math tutor Augustus de Morgan, she was connected to contemporary thinking about mathematics, algebra in particular, and in the Notes she applied and extended those ideas in the context of the Analytical Engine, in detailed analyses and also in more general discussion. Lovelace was particularly interested in "cycles" of repeated operations, and the Bernoulli example involved two nested loops. In her Note E she reused "some of the notation of the integral calculus" to express loops symbolically, including ones with bounds set by computed values rather than constants.[15] This notation enabled her to write a one-line expression she described as "represent[ing] the total operations for computing every [Bernoulli] number in succession"; see Figure 2.[f] This expression provides the missing control structure and, with cross-reference to the table, it could be used to generate the deck of operation cards needed to run the full calculation. It is not a complete program for the Analytical Engine as an additional expression, not provided by Lovelace, would be required to define the structure of the deck of variable cards. In fact the idea of a program as we now understand it did not yet exist (an idea we will return to in a later column). It does, however, provide an interesting and neglected parallel to later efforts to develop abstract notions to specify computational processes.

Similarly, her suggestion that the Analytical Engine might be applied to the analysis and production of music was not a proposal for digital audio recording. One of the first topics she had discussed with de Morgan was the relationship between mathematics and music, an object of investigation since the time of Plato.[17] She had observed the Analytical Engine was "an *embodying of the science of operations*, constructed with peculiar reference to abstract number as the subject of

f Lovelace's Note G.

Figure 2. Lovelace's one-line summary of the Bernoulli computation as a set of nested loops, using a novel mathematical notion of her own devising. The numbers 1–25 refer to the operations listed in the second column of her celebrated table.

$$(1\ldots7),\ (24,\ 25),\ \underset{\text{limits 1 to } n}{\Sigma(+1)^n}\{(1\ldots7),\ (8\ldots12),\ \underset{\text{limits 0 to } (n+2)}{\Sigma(n+2)}(13\ldots23),\ (24,\ 25)\}$$

those operations."[g] Her musical proposal raised the prospect of embodying the abstract science in other areas, as George Boole was shortly to do in the case of logic. Her instincts, and her tragic personal story, were similarly emblematic of their time. She was Romantic with a capital "R," befitting her status as a daughter of Byron and echoing the era's belief in centrality of tortured genius and creative passion to greatness in both arts and natural science.

Grace Hopper

The second of Isaacson's "forgotten women" is Grace Hopper. Unlike Lovelace she was well known for her computing work during her lifetime, spending decades as a popular keynote speaker and appearing frequently on national television. Since her death in 1992 a guided missile destroyer and the Grace Hopper Celebration of Women in Computing (the leading annual event of its kind) have been named after her, as have a bridge and several buildings. She has been the subject of several full-length biographies and a documentary movie on her life is reportedly in progress.

Isaacson's treatment of Hopper in *The Innovators* is generally clear and accurate, lauding her for her collaboration with Aiken and for the pioneering work on compiler and programming language design she directed at Univac during the 1950s. However, a close reading hints at the ways in which inspirational superhero stories can hide the variety of contributions women have made to the history of computing. On page 117 Isaacson credits Hopper as "the technical lead in coordinating the creation of COBOL," a hugely important standard language for business programming that was defined jointly by several computer companies. Standards committee work is a difficult thing to dramatize, which is presumably why Isaacson gives COBOL no more than a passing mention, but as historian of technology Janet Abbate recently noted, his omission slights several women who, unlike Hopper, were on the technical committee in question. Among them is Jean Sammett, who made the largest single contribution to the design of COBOL.[h] Sammet has stated that Hopper was "not the mother, creator, or developer of COBOL," an idea Hopper reportedly endorsed by calling herself the "grandmother of COBOL" rather than its creator.[10]

Sammet's remarkable career began with work on programming languages for computer builders UNIVAC, Sylvania, and IBM.[3] She was the first woman to lead ACM, from 1974 to 1976, and was active in its special interest groups for decades. She pioneered the comparative and historical study of programming languages. Sammet has not been forgotten, as a fellow of the Computer History Museum and a winner of the IEEE Computer Pioneer Award, but sits on the wrong side of a growing chasm separating the handful of women chosen for public veneration from those whose names are known only to specialists.

This is an example of what sociologist of science Robert K. Merton called the "Matthew Effect," the tendency for the most famous person involved in any collaborative project, however peripherally, to be remembered by the public as the key contributor. He named it after a passage in Matthew's Gospel, noting that "unto every one that hath shall be given, and he shall have abundance: but from him that hath not shall be taken away even that which he hath." Over time the famous become more famous, while the moderately well known are forgotten.[12] This is particularly true of those holding distinctions where the supply is deliberately limited, such as Merton's examples of the 40 "immortals" admitted to the French Academy or the winners of the Nobel Prize. It seems to us that a process of natural selection, in which a large pool of eligible pioneers compete for a handful of positions as female role models, is creating a similar dynamic.

The Women of ENIAC

Isaacson writes at length on "The Women of ENIAC," and excerpted this part of *The Innovators* in *Fortune* magazine as part of his launch campaign. ENIAC, the Electronic Numerical Integrator and Computer, is remembered by historians as the first general-purpose electronic digital computer. It was planned and built at the University of Pennsylvania from 1943 to 1945, under contract to the United States Army, which justified its support for the rather bold foray into the technological unknown by pointing to an accumulating backlog of trajectory calculations needed to deliver accurate firing tables along with new artillery pieces. ENIAC missed the war, but when completed it tackled those calculations thousands of times faster than the women who had been working with desk calculators and was configured to tackle a range of other problems, including the first computerized Monte Carlo simulations and the first numerical weather calculations.[i]

ENIAC has always been the most famous of the early computers, but what it has been known for has changed over time. After finding fame as a scientific marvel and "giant electronic brain," it quickly became a yardstick against which the size, weight, and speed of newer computers could be flatteringly compared. In the early 1970s it was at the center of the longest trial ever held in Federal Court, during which a great deal of expensive expert and legal attention was focused on its claims to be the "first computer."

In recent years, however, it has been best known as a computer programmed by women. So here, too, the notion that Isaacson's characters were "forgotten" seems rather fanciful. Unlike Lovelace and Hopper, who were already well known in the 1980s, the six "Women of ENIAC" became famous more recently. However, Jean Bartik—the best known—has been inducted as a fellow of the Computer History Museum and has won the IEEE Computer Pioneer Award. Her obituary appeared in the *New York Times*. The women's work has been celebrated in countless blogs, books,

g Lovelace's Note A, her italics.

h http://bit.ly/1go319Y

i Material on ENIAC here is sourced from our forthcoming book T. Haigh, M. Priestley, and C. Rope, *ENIAC in Action: Making and Remaking the Modern Computer.* MIT Press, Cambridge, MA, 2016.

and scholarly articles and in two documentary films.

Isaacson's retelling of their story is sourced primarily from Bartik's engaging memoir, *Pioneer Programmer* and from a number of oral history interviews.[2] He observed that while "the boys with their toys thought that assembling the hardware was the most important task, and thus a man's job" in fact "all the programmers who created the first general-purpose computer were women." This odd verbal flourish suggests that Isaacson, despite his love of innovation, does not have a very clear idea of what creating a computer involves. Programmers, however gifted, do not usually design computers and design work was already finished when the women in question were chosen for ENIAC work in the summer of 1945. We are struck that Isaacson rhetorically downgrades the contribution of the designers and engineers to "assembling the hardware." Elsewhere he spins a statement from J. Presper Eckert, the project's lead engineer, that he double-checked the calculations of his subordinates into an assertion that Eckert would "hover over the other engineers and tell them where to solder a joint or twist a wire." He seems to have no sense of computer engineering work as something different from tinkering with a soldering iron.

One startling fact was overlooked by Isaacson and almost everyone else who has written about ENIAC. The cliché that the machine was assembled by men but programmed by women is just not true. Dozens of women, employed as "wiremen," "assemblers," and "technicians" worked shifts through 1944 and 1945 to build ENIAC, threading miles of wire through the machine and soldering approximately half a million joints. Other women drew blueprints and helped with administration. While Bartik and her colleagues were given belated recognition for their work on ENIAC, more than 50 women worked on the machine in 1944 alone. Their names are preserved only in the project's accounting records and personnel forms. They are truly forgotten.

We also believe the word "programming" does not do a good job of capturing the actual work of Bartik and her colleagues. When programming developed as a distinct job, in the mid-1950s, it was generally separated from the work of operating the computer or any ancillary punched card units. Programmers sat with pencils and coding pads, writing instructions. These were punched onto cards by keypunch women, the data entry clerks of their day. Operators tended to the machine itself, preparing the computer and its peripherals for a job, loading the appropriate card and tape media, and extracting printed output.

The women of ENIAC combined elements of all three roles, but operations work was what they were hired to do and it occupied most of their time with the computer. They were chosen for their skill as human computers, carrying out manually the tasks ENIAC would process electronically. Their initial training was in punched card machine operation. ENIAC's tiny writable electronic memory (just 200 decimal digits) meant running a reasonably ambitious job involved interleaving computer steps, in which decks of cards were run though ENIAC for automatic computations, with manual steps in which the cards were punched with new values, sorted, collated, or run through the tabulator for printing.

The initial cohort of six operators had to understand how ENIAC worked, so they could set it up for a new job, which involved many hours of plugging cables and setting switches, and monitor its progress. Because of this expertise they also played an important preparatory role in many of the applications run on ENIAC, partnering with scientific and mathematical users to figure out the "set-up" needed to configure the machine to carry out the appropriate series of mathematical operations. These set-ups implemented loops, subroutines, and the other elements we now associate with programs (although they were recorded visually, as a schematic of the switch and wiring settings needed on ENIAC's various units, rather than as textual instruction codes).

Lovelace was part of her own time, not a visitor from the future or lone superhero who invented programming and created the modern world.

It is this preparatory work that has caused them to be remembered as programmers, though their contributions here are frequently exaggerated. For example, it is often claimed ENIAC programming was an afterthought that received no attention from the engineering team. The women's realization, when planning trajectory computations, that ENIAC's "master programmer" unit could be used to implement loops is sometimes taken as a breakthrough moment in programming history.

In fact, ENIAC's designers had paid considerable attention to the trajectory calculations early in the design process, culminating at the end of 1943 in the creation by Arthur Burks of elaborate set-up and timing diagrams for this problem. This work profoundly shaped ENIAC's final design, particularly its master programmer, and introduced the diagramming and planning techniques later used by Bartik and her colleagues. The delight the women expressed in later interviews in grasping techniques such as looping are familiar to later generations of computing students but do not indicate they had discovered an unanticipated use for the master programmer.

Many of the errors and omissions in Isaacson's take on this history come from the oral histories and existing books he relies on, and reflect a conventional wisdom that has developed over time. His most novel suggestion, apparently based on a misreading of Bartik's memoir, is that she contributed to discussions on

computer design held between John von Neumann and the ENIAC project engineers held at the Moore School in early 1945.[j] By April these led to the *First Draft of a Report on the EDVAC*, the first statement of the crucial ideas of modern computer architecture. While the proper balance of credit for these ideas has been much disputed, Isaacson is unique in assigning Bartik, who had not yet been recruited for ENIAC work, a role in formulating the new instruction set. This error will surely be repeated by others, further building the superhero myth.

Our intent is not to belittle the very real contributions of the ENIAC operators to computing history, but to argue that forcing them into the role of the "first programmers," as is commonly done, hides most of the reality of their actual work. Operating ENIAC, and its ancillary punched card machines, was highly skilled work that was absolutely vital to the machine's success. To this day programming accounts for a fairly small proportion of overall IT work, and the demand for system administrators, network technicians, facilities management staff, and other hands-on IT practitioners is greater than ever. By the 1960s, however, computer operations work was seen as a lower paid and lower skilled occupation—blue-collar shift work against the white-collar work of programming. Perhaps celebrating hands-on operations work would mislead impressionable young women. As Wendy Hui Kyong Chun has noted, "reclaiming these women as the first programmers...glosses over the hierarchies...among operators, coders, and analysts."[4] Remembering the ENIAC operators only as programmers implies gender matters to history but social class does not.

Conclusion

A great deal is at stake here. Women are conspicuous by their absence in computer science classrooms and in the programming teams, data centers, and computing research labs of America.[6] Computer technology has been actively claimed by some men as their exclusive domain, as in the recent "Gamergate" campaign to deter women from writing about video games. While most areas of science and engineering are gradually becoming more balanced in their gender representation, computer science has slipped backward over the past 30 years. This has prompted a great deal of hand wringing and countless blue ribbon committees, research reports, proposed interventions, and volunteer campaigns to recruit and retain more women in various areas. ACM has been heavily involved in this area. Maria Klawe, ACM president from 2002–2004, spearheaded work at Princeton and, more recently, as President of Harvey Mudd College. ACM-W, which "supports, celebrates, and advocates internationally for the full engagement of women in all aspects of the computing field" is open to all members without additional cost. In 2006, ACM ended a 40-year streak for the boys' team when it presented a Turing Award to Frances Allen. Women won again in 2008 and 2012.

History is playing an increasingly visible role in this struggle. One widely held concern is that girls are not exposed to images of female programmers or information technology professionals. Films about Jobs, Gates, Turing, and Zuckerberg encapsulate the broader tendency of mass media to present computers primarily as an object of fascination for brilliant but spectacularly awkward men. In their own lives teenage girls are less likely than boys to find peers with an interest in programming or computer technology. They have little reason to assume that software development is work they would find interesting or be good at.

The past has therefore been mined in search of women who might provide girls with inspiring counter-examples. The goal of convincing women to enter computing might sometimes seem at odds with the responsibility of historians to provide accurate and nuanced stories. In such a battle the force of numbers would not be with the historians, who find intrinsic value in the past of a field that is relentlessly focused on the future. In contrast, most people with a connection to computing encounter history only occasionally as disconnected fragments, for the time it takes to read a blog post, watch a movie, or walk through a science museum exhibit.

We run the risk of hiding a rich historical tapestry, full of thousands of little figures carrying out many vital tasks.

We believe good history is not just important to specialists, and that it will ultimately prove more inspiring and more relevant than superhero stories. Isaacson's book has some distinctive strengths and weaknesses, but we have used it here primarily as an accessible, and influential, summary of the emerging public understanding of the contribution of these superhero coders to the invention of the modern world. This has many parallels with the recent film, *The Imitation Game*, in that both are designed to flatter their audiences by making them feel superior to the sexist, homophobic, or otherwise flawed characters who conspire unsuccessfully to prevent their heroes from reaching their special destinies. Stories of this kind inflate their own importance, by pretending a certain idea, person, or invention has direct and singular responsibility for some hugely complex chain of later events that leads to our modern "digital world." Their authors are fixated on firsts and origins, and glorify flashes of abstract insight over slow and incremental technological change driven by the needs of users.

The conception of history as the deeds of a few "great men" has problems that are too fundamental to correct simply by elevating a few "great women" to the pantheon. Indeed, this strategy devalues many of the contributions made by women. Nobody has

j Isaacson's discussion, on pages 106–107, is apparently based on pages 118–119 of Bartik's memoir. She is discussing work with von Neumann in 1947 to plan a new instruction set for ENIAC, rather than 1945 design work on EDVAC.

chosen to celebrate "The Key Punch Operators Who Invented the Digital Universe" or "The Lone Computer Assembler Who Built the Modern Computer." The obsession with discovering the first coders, intended to empower women, has obscured the importance of other kinds of computer-related labor that was carried out by women for decades. The quest for "girls who code" is erasing the history of "women who operate." One of the reasons such work was seen as low status and unimportant is that it was carried out by women, and any attempt to challenge historical sexism should dispute that assumption rather than tacitly endorsing it. Scholars have recently looked more systematically at the historical participation of women in computing, most notably in Janet Abbate's *Recoding Gender*,[1] which followed the history of women in computing from the 1940s to the 1970s, and in the collection *Gender Codes* edited by Thomas Misa.[13] Nathan Ensmenger has looked at rhetoric around the gendering of programming in the 1960s and 1970s.[7] Marie Hicks and Ian Martin have both explored the work of computer operators, looking at the intersection of class and gender during the same era.[9,11] Laine Nooney has been exploring the career of Roberta Williams, cofounder of Sierra On-Line and designer of the King's Quest series of adventure games, as a different model for female success.[14]

Stories about superheroes make great entertainment, as testified to by the huge success of the Avengers films and the other movies and licensed television series based on the Marvel characters. The superhero narrative is not, however, the best way to understand history. Isaacson's celebration of Ada Lovelace, Grace Hopper, and the "Women of ENIAC" as the creators of computer programming is part of a much broader movement to make these women into recruiting sergeants for computer science. The colorful feats sometimes attributed to these women can overshadow their actual historical contributions, which were rather more nuanced and rather less dependent on superpowers. In particular, the concept of "programming" has been awkwardly projected back in time, in a way that misrepresents their accomplishments and squanders the chance to celebrate a broad range of computer-related occupations such as mathematical analysis, operations work, and management.

More generally, we run the risk of hiding a complex historical tapestry, full of thousands of little figures carrying out many vital tasks, behind a gaudy poster depicting a small team of superheroes. The quest for "genius" devalues the vital contributions of millions who are merely creative, intelligent, hard-working, and lucky enough to be in the right place at the right time. Superhero stories have little time for ordinary humans, who exist only to be endangered or rescued. Reducing the story of women in computing to the heroics of a handful of magical individuals draws attention away from real human experience and, counterproductively, suggests that only those with superhuman abilities need apply. C

References

1. Abbate, J. *Recoding Gender: Women's Changing Participation in Computing.* MIT Press, Cambridge, MA, 2012.
2. Bartik, J.J. *Pioneer Programmer: Jean Jennings Bartik and the Computer that Changed the World.* Truman State University Press, Kirksville, MO, 2013.
3. Bergin, T.J. Jean Sammet: Programming language contributor and historian, and ACM president. *IEEE Annals of the History of Computing 31*, 1 (Jan.–Mar. 2009), 76–85.
4. Chun, W.H.K. *Programmed Visions: Software and Memory.* MIT Press, Cambridge, MA, 2011, 34.
5. Cohen, I.B. *Howard Aiken: Portrait of a Computer Pioneer.* MIT Press, Cambridge, MA, 1999, 61–72.
6. Cohoon, J.M. and Aspray, W., Eds. *Women and Information Technology: Research on Underrepresentation.* MIT Press, Cambridge, MA, 2006.
7. Ensmenger, N. *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise.* MIT Press, Cambridge, MA, 2010.
8. Essinger, J. *A Female Genius: How Ada Lovelace, Lord Byron's Daughter, Started the Computer Age.* Gibson Square, London, 2014, 174.
9. Hicks, M. Only the clothes changed: Women computer operators in British computing and advertising, 1950–1970. *IEEE Annals of the History of Computing, 32*, 4 (Oct.–Dec. 2010), 5–17.
10. Lohr, S. *Go To: The Story of the Math Majors, Bridge Players, Engineers, Chess Wizards, Maverick Scientists and Iconoclasts—The Programmers Who Created the Software Revolution.* Basic Books, New York, 2001, 52.
11. Martin, I. Structuring information work: Ferranti and Martins Bank, 1952–1968. *Information & Culture 47*, 3 (2012), 312–339.
12. Merton, R.K. The Matthew effect in science. *Science 159*, 3810 (1968), 56–63.
13. Misa, T.J. *Gender Codes.* IEEE Computer Society Press, Hoboken, NJ, 2010.
14. Nooney, L. A pedestal, a table, a love letter: Archaeologies of gender in videogame history. *Game Studies 13*, 2 (2013).
15. Priestley, M. *A Science of Operations: Machines, Logic and the Invention of Programming.* Springer, 2011, 41–44.
16. Stein, D.K. *Ada: A Life and a Legacy.* MIT Press, 1985, 92.
17. Toole, B.A., Ed. *Ada, the Enchantress of Numbers: A Selection from the Letters of Lord Byron's Daughter and Her Description of the First Computer.* Strawberry Press, Sausalito, CA, 1992, 245.

Further Reading

Abbate, Janet.
Recoding Gender: Women's Changing Participation in Computing. **MIT Press, Cambridge, MA, 2012. This starts with ENIAC and Colossus and moves forward as far as the 1970s. As a small book on a huge topic it focuses on particular examples, but these are well chosen and give a sense for broader changes in the IT workplace.**

Bartik, Jean Jennings.
Pioneer Programmer: Jean Jennings Bartik and the Computer that Changed the World. **Truman State University Press, Kirksville, MO, 2013. Bartik's memoir gives a lively and idiosyncratic view of the ENIAC story.**

Beyer, Kurt W.
Grace Hopper and the Invention of the Information Age. **MIT Press, Cambridge, MA, 2009. The most detailed account to date of Hopper's work with Aiken at Harvard and at Univac, putting her achievements into the broader context of early computing.**

Fritz, W. Barkley.
The Women of ENIAC. *IEEE Annals of the History of Computing 18*, 3 (Fall 1996), 13–28. Full of material Fritz elicited from the women themselves, this article played a crucial role in bringing their contributions to broader awareness.

Padua, Sydney.
The Thrilling Adventures of Lovelace and Babbage: The (Mostly) True Story of the First Computer. **Pantheon, 2015. This charming, funny, and heavily footnoted graphic novel turns Babbage and Lovelace into literal superheroes. It is no less accurate than many other popular retellings of the story but has the good grace to admit that much of its action takes place in a parallel universe.**

Stein, Dorothy.
Ada: A Life and Legacy. **MIT Press, Cambridge, MA, 1985. Published 30 years ago, but still the most reliable biography of Ada Lovelace.**

Swade, Doron. `
The Difference Engine: Charles Babbage and the Quest to Build the First Computer. **Viking Penguin, New York, 2001. This wonderfully engaging book has two halves. One describes Babbage's own computing career; the other the ultimately successful project led by Swade to complete his plans to build the "difference engine," the earlier and less complicated of Babbage's two planned computers.**

Thomas Haigh (thaigh@computer.org) is an associate professor of information studies at the University of Wisconsin, Milwaukee, and immediate past chair of the SIGCIS group for historians of computing.

Mark Priestley (m.priestley@gmail.com) is an independent researcher into the history and philosophy of computing.

DOI:10.1145/2804244

Joe Karaganis and Jennifer Urban

Law and Technology
The Rise of the Robo Notice

Examining the conflicting claims involving the use of automated tools in copyright-related notice-and-takedown procedures.

IMAGE BY MARSIL

Most Internet professionals have some familiarity with the "notice and takedown" process created by the 1998 U.S. Digital Millennium Copyright Act (the DMCA). Notice and takedown was conceived to serve three purposes: it created a cheap and relatively fast process for resolving copyright claims against the users of online services (short of filing a lawsuit); it established steps online services could take to avoid liability as intermediaries in those disputes—the well-known DMCA "safe harbor"; and it provided some protection for free speech and fair use by users in the form of "counter notice" procedures.

The great virtue of the notice and takedown process for online services is its proceduralism. To take the most common example, if a service reliant on user-generated content follows the statutory procedures, acts on notices, and otherwise lacks specific knowledge of user infringement on its site (the complicated "red flag" knowledge standard), it can claim safe harbor protection in the event of a lawsuit. Services can make decisions about taking down material based on substantive review and their tolerance for risk. They may also adopt technologies or practices to supplement notice and takedown, though the law makes no such demands beyond a requirement for repeat infringer policies. The resulting balance has enabled a relatively broad scope for innovation in search and user-generated-content services. As one entrepreneur put it in our recent study of these issues, notice and takedown was "written into the DNA" of the Internet sector.

This basic model held for about a decade. In the last five or six years, however, the practice of notice and takedown has changed dramatically, driven by the adoption of automated notice-sending systems by rights holder groups responding to sophisticated infringing sites. As automated systems became common, the number of takedown requests increased dramatically.

For some online services, the numbers of complaints went from dozens or hundreds per year to hundreds of thousands or millions. In 2009, Google's search service received less than 100 takedown requests. In 2014, it received 345 *million* requests. Although Google is the extreme outlier, other services—especially those in the copyright 'hot zones' around search, storage, and social media—saw order-of-magnitude increases. Many others—through luck, obscurity, or low exposure to copyright conflicts—remained within the "DMCA Classic" world of low-volume notice and takedown.

This split in the application of the law undermined the rough industry consensus about what services did to keep their safe harbor protection. As automated notices overwhelmed small legal teams, targeted services lost the ability to fully vet the complaints they received. Because companies exposed themselves to high statutory penalties if they ignored valid complaints, the safest path afforded by the DMCA was to remove all targeted material. Some companies did so. Some responded by developing automated triage procedures that prioritized high-risk notices for human review (most commonly, those sent by individuals).

Others began to move beyond the statutory requirements in an effort to reach agreement with rights holder groups and, in some cases, to reassert some control over the copyright disputes on their services. These "DMCA+" measures included the use of content filtering systems, such as YouTube's ContentID and Audible Magic. They included special takedown privileges for "trusted" senders; expanded profiling of users; restrictive terms of service; modifications to search functionality; and other approaches that stretched beyond the DMCA's requirements.

The strength of these measures depended on a variety of factors, but most importantly the service's leverage with rights holders. Smaller services sometimes ceded control of takedown to rights holder groups, in some cases via direct back-end administrative access. Larger services with more leverage sought to retain control but, in some cases, adopted blocking or content filtering systems as part of their risk assessments. Rights holder groups worked to turn incremental concessions into perceived industry norms that could filter back into law through litigation.

As automated systems became common, the number of takedown requests increased dramatically.

Today, for online services with significant copyright exposure (particularly search, cloud storage, music, and video services), the DMCA's procedural balance is, practically speaking, obsolete. The notice and takedown process has been widely supplanted by practices that shift the balance of protection from users to rights holders, and that hold out the statutory remedies as, at best, last resorts for user and services. For services outside the major copyright conflict zones, on the other hand, notices remain relatively infrequent and DMCA Classic process is still generally the rule, built around the substantive, human review of notices.

One major concern in our work is to understand the balance of power in this divided ecosystem. Does the adoption of DMCA+ measures by large players drive out DMCA Classic over time—forcing an ever-wider range of sites to manage floods of automated notices or adopt content filtering and other strategies? Some of the smaller services we spoke with (in interviews and surveys conducted in 2014) clearly expressed such concerns. From their perspective, traditional DMCA compliance appears fragile—dependent on rights holder forbearance or inattention that could change at any time.

A second important question is whether automated notices do, in fact, reliably target infringing material. As rights holders are quick to point out, automated systems overwhelmingly target either file-sharing sites that can be plausibly described as "dedicated to infringement" (in the current enforcement lingo), or search engine results that lead to those sites. For rights holders, such targeting generally justifies the minor sins of misidentification associated with these systems, such as the targeting of dead links or pages. The likelihood of damage to important protected expression, in these cases, is relatively low. But the same practices are also brought to bear against a wider array of sites, including those with predominantly non-infringing uses and active notice and takedown policies. In these cases, the risks of collateral damage are higher.

The accuracy of automated notices has become a flashpoint in copyright enforcement debates, marked on one side by rights holder claims to reliability and procedural safeguards (such as human spot checks) and, on the other side, by a steady stream of anecdotal examples of bad takedowns. One goal of our work is to examine these conflicting claims. To this end, we undertook a close examination of a representative sample of the 108 million takedown requests sent to Google Search during a six-month period in 2013. Google, it is worth noting, is both a pioneer of automated triage and a major investor in human review. On average, it takes down 97.5% of requests. This percentage proved, in our surveys, to be on the low end for companies that receive large numbers of automated notices.

Our work suggests the situation is, predictably, complex. Nearly all of the 1,800 requests in our sample targeted file-sharing sites—in principle, the easy targets with low potential for non-infringing content. In practice, approximately 1 in 25 (4.2%) of these requests were fundamentally flawed—simply targeting the wrong content. Nearly one-third of requests (28.9%) had characteristics that raised concerns about the validity of the claim. This number included 7.3% targeting content that could make a reasonable claim to one or more fair use criteria and 10.6% that led to dynamic search results or aggregator pages that made identifying the targeted content difficult or impossible. (This form of identification has

The accuracy of automated notices has become a flashpoint in copyright enforcement debates.

not fared well in recent court cases). Despite the resources Google invests in this process, almost none of these notices, statistically speaking, receive substantive examination. Although the practices are not directly comparable, several DMCA Classic services (which often do look closely at the targeted content) reported rejecting much larger proportions of notices, in some cases exceeding 50%.

Automation has transformed notice and takedown from a targeted remedy for specific acts of infringement into a large-scale strategy for limiting access to infringing content. At one level, this looks like a fruitless enterprise: an endless game of whack-a-mole against a resilient and multi-jurisdictional file-sharing community. Yet, as the big services grow bigger and are pushed beyond the DMCA, important parts of the user-generated-content Internet ecosystem shift toward rights holder control.

And so the process continues to escalate. This year may see well over one billion DMCA takedown requests, further marginalizing the procedural protections afforded by the DMCA. The adoption of content filtering by video and music services such as YouTube, SoundCloud, and Vimeo represents a profound shift in the direction of enforcement—no longer basing takedown on the rights holder's identification of unauthorized uses after the fact, but on the a priori filtering of user activity in collaboration with rights holders. For many DMCA Classic services, such filtering represents new and potentially unsustainable costs and a threat to the wide latitude for expression on which their user communities are built. For the DMCA+ services, filtering is a response to a complex array of pressures, from staying on the right side of perceived enforcement norms, to exercising some control in keeping content up, to creating—in the case of ContentID—a new payment model for artists.

There is no easy way around these dilemmas. Stronger liability for reckless or malicious notice use might be a good step in curbing the worst notice practices, which can include deceptive or predatory behavior. But such changes are currently a dead letter in U.S. copyright politics. Nor is there appetite on the part of the Internet companies to revisit the core DMCA compromises, for fear of undermining the general principle of safe harbor protection. And so despite Congressional interest in copyright reform more generally, there would seem to be little chance of significant policy change in this area. The rights holder companies are slowly winning on enforcement and the largest Internet companies have become powerful enough to fend off changes in law that could threaten their core business models. The ability of large companies to bear the costs of DMCA+ systems, moreover, has become a source of competitive advantage, creating barriers to entry on their respective terrains. This will not be news to those watching the business consolidation of Web 2.0. It is bad news, however, for those who think both copyright and freedom of expression are best served by clear statutory protection and human judgment regarding their contexts and purposes. In important parts of the Internet sector, robots have taken those jobs and they are not very good at them. C

Joe Karaganis (joe.karaganis@gmail.com) is vice president at The American Assembly, a public policy institute based at Columbia University.

Jennifer Urban (jurban@law.berkeley.edu) is assistant clinical professor of law at the University of California, Berkeley, and director of Berkeley Law's Samuelson Law, Technology & Public Policy Clinic.

The authors, with Brianna Schofield, have produced a study of notice and takedown available at http://takedownproject.org/.

DOI:10.1145/2804246

Dorothea Kleine

Global Computing
The Value of Social Theories for Global Computing

Conceptual toolkits for projects that work.

IMAGE BY BILLIONPHOTOS.COM

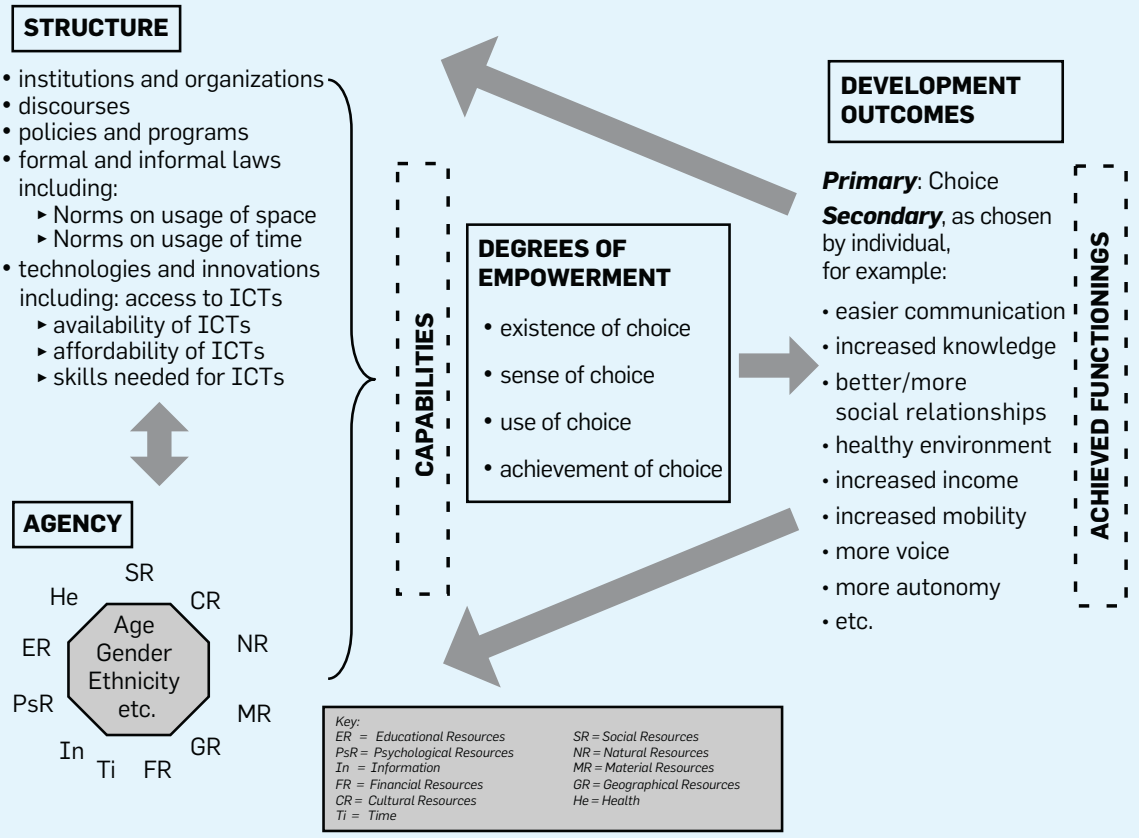
If you want to use technology to help people improve their lives, in whatever community or country, you need to understand not only the technology but the people as well. This simple truth has brought computer scientists together with people like me—a social scientist. Theory helps: you may recall finite state machines from your computer theory course. In this column, I explain how social theories can help.

For instance, Actor-Network-Theory (ANT), developed by Michel Callon, Bruno Latour,[5] John Law,[6] and others in the 1980s (for more see Latour[4]), views socio-technical systems as networks of actors (actors include humans, technologies, and formal processes) situated in some relationship to each other. These actors affect each other based upon their linking relations. We can use ANT to analyze, for example, how a new software user interface "acts" within its network to enable or prevent actions of other actors (including human users) in the network. Design intentions do not always equal usage outcomes. For instance, consider an e-government system designed to stop corruption by "acting" in a network of actors, some human and some non-human. Depending on how other actors respond, corruption may indeed stop. However, if users find ways of subverting or circumventing the software, corruption might continue along altered channels.

ANT is just one social theory, which is being applied in the field of information and communication technologies for development (ICT4D).[a] Another theoretical approach used in ICT4D, and one that I have been working with, is the Capability Approach (CA). The CA sees development as something more than economic growth, and urges practitioners to design projects and technologies based on a broader concept of human development. Amartya Sen, the Nobel-prize winning economist who co-founded this approach, defines development as the "process of expanding the real freedoms that people enjoy." He argues that development should focus on the "freedom of people to lead the lives they have reason to value and to enhance the real choices they have."[7] This pluralistic and holistic vision of development includes social, cultural, political, and economic expressions of well-being. People are to be in-

a ICT4D examines how ICTs might improve the lives of disadvantaged or marginalized people, many in the Global South of Asia, Africa, and Latin America.

The Choice Framework.



volved in decisions about their lives in a participatory way, not just provided for in a one-size-fits-all and top-down mode of "delivering development."

Applying social theory to practical ICT4D work is not easy and process diagrams can help depict social activities in ways that aid this application. I have translated the Capabilities Approach into a Choice Framework[3] by building on work by Alsop and Heinsohn[1] and the DFID Sustainable Livelihood Framework.[2] This framework traces development processes and examines where technologies can have an impact (see the accompanying figure). In this way social theory can help technologists design and develop systems based on a better analysis of the wishes of and the constraints experienced by the people they design for (or with).

For example, a person may value health information to determine what to do when a family member is ill. Technologists may be interested in developing systems that enable this. We use the figure to analyze specific circumstances and personal characteristics: gender, age, ethnicity, class, and available resources. We can ask about educational resources (including levels of print literacy), health and disability status, financial resources, what information they already have, available material resources (including IT equipment), geographical resources (including relative distance to other places), and access to cultural and natural resources. Social resources include who can advise and help. Psychological resources include curiosity, willingness to learn something new, confidence, and tenacity needed to take full advantage of the technical innovation designed for (or with) that person. An analysis of the intended users' resources helps identify challenges for the design process.

An analysis of the intended users' resources helps identify challenges for the design process.

Each person has *individual agency* based upon the portfolio of available resources. In the next logical step, the person then confronts a *social structure* to navigate in order to make choices. Institutions and organizations, such as the government health department or non-governmental organizations working in the health sector, might be in a position to provide health information. Public discourses in media and in the person's com-

munity are another element of the social structure. There might be formal and informal norms, which are often gendered in ways that affect women's ability to act. For example, in traditional societies unevenly distributed childcare duties might make women available only in the evening, while social expectations might prohibit women from walking alone or leaving home in the evenings. I have encountered open public access computing points that were seen by the community as a legitimate space for both women and men to congregate, and they can act as empowering spaces allowing women to socialize outside the home. I have also seen cybercafés evidently designed for young men, with an emphasis on violent games and suggestive advertising. In the traditional setting of the village, unwritten social norms would make it impossible for women to use the services such a cybercafé offers.

All of these context aspects must be considered before we assist in achieving a development outcome through access to health information. This Choice Framework can be applied to three design options:

▸ **Technology A:** An extensive online health information database in relevant languages approved by the health ministry and accessed on a public computer either in a local leader's office or a community healthcare facility. This could work but based on the preceding analysis we can anticipate users will need to be literate and women may find it difficult to find time to come to the public access point. Further, if the community healthcare officer is female and the local leader is male (as is likely), then social norms would make it easier for women to visit the healthcare office. Thus it would be good to have the computer there. Having a space where women can legitimately congregate might create additional empowerment effects when women can meet other women, share health information, and potentially develop locally relevant content of their own.

▸ **Technology B:** A recorded health message system on simple feature phones. While good to deal with illiterate users, the amount of information conveyed might be less. Also, it no longer relies on a communal access point but delivers information to devices for individual users. While less time-intensive, it might not empower women through a shared access space the way Technology A would.

Different strategies can be applied to bring social science expertise to project teams.

▸ **Technology C:** A tablet that allows access to online content, but requires more Internet connectivity (often via mobile broadband). Content might be richer than on the feature phone, but use might require literacy. The empowerment effect of the shared space may also be missed if the device is used in an individualized, rather than a collective way.

The Choice Framework provides a map of social factors that must be taken into account when considering these options. Which resources will be required for each option, and which resources might be affected positively or negatively? Which structural conditions are in place? Whether an ICT4D project succeeds depends on whether, after the technological change, better choices exist. But that is not enough, users must have a sense of the choices that are open to them through the technology and they need to make use of their choices. Ultimately, given the choices made, is the user then able to achieve what they wanted, which here was to get better access to health information?

Social theories such as the Capabilities Approach can be used by interdisciplinary teams to understand their work in context. Mixed teams including computer scientists can use translational devices such as the Choice Framework to make social theory much more concrete. The Choice Framework offers an analytical map of factors and forces governing social change, and allows innovators to identify where leverage might be applied to widen user control and choice in ways they value. These are examples of how social theory can be applied fruitfully in ICT4D or any user-focused design.

Different strategies can be applied to bring social science expertise to project teams. I recommend that larger projects engage multidisciplinary teams. Where that is not an option, buddy systems between social and computer science researchers can cause both sides to explore theoretical approaches beyond their disciplines. This is something that can be learned early on—for example, the Master's program in Practising Sustainable Development (ICT4D specialism) at Royal Holloway, University of London, consciously creates cohorts of students from different academic and professional backgrounds and offers a multidisciplinary curriculum.

ICT4D is committed to effective positive change. For this to work, a multidisciplinary spirit is imperative. Computer and social scientists should draw on the best theories to reach ethical decisions on technology, to deepen their analysis, and to conduct projects that work. C

References

1. Alsop, R. and Heinsohn, N. *Measuring Empowerment in Practice—Structuring Analysis and Framing Indicators.* Washington, D.C., 2005.
2. DFID. *Sustainable Livelihoods Guidance Sheets.* Department for International Development, London, U.K., 1999.
3. Kleine, D. *Technologies of Choice: ICTs, Development and the Capabilities Approach.* MIT Press, Cambridge, MA, 2013.
4. Latour, B. *Reassembling the Social: An Introduction to Actor-Network-Theory.* Oxford University Press, Oxford, U.K., 2005.
5. Latour, B. *Science in Action: How to Follow Scientists and Engineers Through Society.* Open University Press, Milton Keynes, U.K., 1987.
6. Law, J., Ed. *Power, Action, and Belief: A New Sociology of Knowledge.* Routledge, London, 1986.
7. Sen, A. *Development as Freedom.* Oxford University Press, Oxford, U.K., 1999.

Dorothea Kleine (Dorothea.Kleine@rhul.ac.uk) is the director of the multidisciplinary ICT4D Centre at Royal Holloway, University of London (www.ict4dc.org) where she leads the Masters in Practising Sustainable Development (ICT4D specialism). Her Choice Framework is further laid out in articles and in her book *Technologies of Choice: ICTs, Development and the Capabilities Approach*, MIT Press, 2013.

DOI:10.1145/2804248

Peter J. Denning

The Profession of IT
Automated Education and the Professional

Technology boffins argue the new technologies of intelligent personal learning environments will put universities out of business. Will the purported successor, an automated global virtual university, be up to the task of professional education?

THREE NEW BOOKS proclaim a coming disruption for higher education: *The End of College* by Kevin Carey,[1] *College Disrupted* by Ryan Craig,[2] and *Hire Education* by Michelle Weise and Clayton Christensen.[8] They tell variations of the following story. The common model for a university encompasses three purposes: practical training, research, and liberal education. Trying to serve all three well has driven up costs, tuitions, employer dissatisfaction, student debt, and student disillusionment to unsustainable levels. Intense competition among universities for ranking and prestige further exacerbates these issues. The new technologies of massive open online courses (MOOCs) and online competency-based modules (OCBMs) provide numerous online alternates at much lower cost. They will draw students away and cause many university bankruptcies in the next decade. A highly automated, low-cost, global, virtual "University of Everywhere" will emerge from the disruption.

Lew Perelman, who first foresaw competency-based learning technologies in his book *School's Out*,[7] recently commented: "What will really disrupt academia is not mass production of impersonal teaching but mass access to personalized learning, plus employment selection based on demonstrated competencies, not academic credentials. That is the fate that now faces academia."[a]

The disruptive threats are real and education leaders should be taking them seriously. However, these authors reflect an unwarranted faith in education technology. The envisioned automated university is unlikely to satisfy many education goals and is likely to shortchange professionals.

The faith in technology behind these predictions is based on recent advances in artificial intelligence and data analytics, which open possibilities we could hardly envision as recently as a decade ago. The headline-hogging MOOCs are the lesser threat; they are turbocharged platforms for traditional classroom courses scaled up several orders of magnitude for large student numbers. The under-the-radar OCBMs

a Perelman, L. MOOCs: Symptom not cause of disruption. ACM *Ubiquity* (Aug. 2014). A very good account of how the powers of different groups influence the shape of the disruption; http://ubiquity.acm.org/article.cfm?id=2591680.

IMAGE BY WAVEBREAK MEDIA

are the greater threat because they use new designs to explicitly help students develop job-related competencies and they offer their own credentialing systems including professional certificates and badges. The promoters of these new technologies claim these main benefits:

- Mass personal learning environments will detect and adapt to individual student learning styles by gathering and analyzing large amounts of student data, exploiting research in deep machine learning, cognitive science, and pedagogy.
- Technology-enabled collaboration and telepresence will amplify student success rates toward learning objectives.
- New credentialing systems will accurately report how well a student is able to perform at specific jobs or in specific environments.
- It will become possible to design OCBM-builders that generate the software for a particular OCBM in response to given learning objectives.
- These environments will provide a pathway to mastery.

Is computing technology up to delivering these claims? Can these technologies actually provide a path to mastery? Is the imagined post-disruption world's University of Everywhere what we really want from education?

Personal Learning Environments and Expert Systems

Artificial intelligence grew from a belief from the 1940s that a large enough network of electronic components could function like a brain: thinking, understanding, being self-conscious, and substituting for experts. This led to claims that machines could carry on natural conversations, make scientific discoveries, prove mathematical theorems, automate many cognitive tasks, understand world trends, and beat humans at chess. Except for chess, none of those claims was ever fully realized, and even the chess claim weakened when laptop-equipped chess player teams discovered they could regularly beat the grandmaster machines.

In 1972, Hubert Dreyfus, a philosopher at UC Berkeley, published a famous but controversial book *What Computers Can't Do*[5] in which he questioned the popular claim that machines would become experts in various human domains. He argued we infer expertise from the skills people exhibit when performing in a domain. He identified six levels of skill: beginner, advanced beginner, competent, proficient, expert, and master. Each is a higher level of embodiment than the previous. Embodiment means the skill is ingrained into our bodies (not just our brains) through immersive practice until it becomes so automatic that we are not aware we are doing the practice.

Is the imagined post-disruption world's University of Everywhere what we really want from education?

Dreyfus argued that beginners and advanced beginners determine actions by following rules. Competent people, however, perform most of their actions automatically and resort to rule following only when confronted with a new situation. He argued that computing machines, which follow rules, cannot attain proficient or higher levels because human performance at those levels cannot be characterized as following rules. It took many years before Dreyfus's controversial argument was vindicated. Although some expert systems performed competently, no one thought they performed as a true human expert.

In his more recent book *On the Internet*[4] Dreyfus evaluates the capability of the Internet to host personalized learning environments. He concludes that a personal learning environment itself is an expert system that, like other expert systems, cannot rise above the level of competence at what it does—teaching. Telepresence and massive data do not add much. To say machines can now store and retrieve enormous patterns just increases the size of the rules a machine can follow but does not grant the machine new non-rule-following powers such as we embody in our biologies.

Dreyfus examines what master teachers do that machines cannot do. For example, master teachers report they can tell when a student is learning by looking for a "dawning look of understanding" in the student's eye; no one knows how teachers do this, and no one knows how to build a machine of which a student says "It looked at me with understanding." More to the point, master teachers foster learning environments in which students develop proficiency, expertise, and mastery: the traditional methods of apprenticeship, conversation, immersion, mentoring, and coaching cannot be replicated by machines. With a team of colleagues, Dreyfus recently released a movie, *Being in the World*,[b] which shows six masters from diverse fields and proposes language that allows us to talk about what they do and how they became masters. It is difficult to go away from viewing this movie with any impression that any automated learning environment can possibly cultivate mastery.

Even though personal learning environments cannot be master teachers, these environments are still a major economic threat to existing university ways. There is a huge market of people wishing to become advanced beginners or competent in well-defined domains, such as network administration or basic programming, where criteria for competence are precise and testable. Personalized learning environments can do many useful things to support this goal: bring materials such as books and videos to students, provide simulators and virtual environments to host experiments, administer tests, tailor interactions to support students where their tests indicate they are weak, and provide tools for communicating with other students. Many students report a well-designed MOOC or OCBM is more engaging and pleasurable than a typical classroom lecture course.

A practical implication of Dreyfus's argument is that designing OCBMs will never be easy and will resist automation. I can testify to this from personal experience. In 1993, in the early days of the Web, Daniel Menascé and

b The video for *Being in the World* is available from http://beingintheworldmovie.com

I at George Mason University set out to make OCBMs (then called "tutorial modules") in computer science available on the Web from our new Hyperlearning Center.[c] For each topic, we devised the learning objectives and displayed them as concept maps. We designed and built a system called Hyperlearning Meter that would generate online tests customized for each student. We carefully designed questions that would reveal whether or not the student understood a key idea. We developed a programming language to describe a test question template and a test generator that would fill in parameters of a template and present an actual question. When a student asked for a test, the generator would select a random sample of templates from the question database and present a test to the student by randomly selecting numerical parameters for the variables in each one.

We spent an enormous amount of time designing individual questions and accumulating a large enough database to enable customization of tests to individual students. We drew deeply on our extensive experience as computer science teachers to design concept maps, anticipate the ways students would veer off track, and design tests to detect when they had. We had to employ our expert knowledge on each and every question. There was no way we could automate the process of generating question templates. The same is true today.

Post-Disruption Education

It is more likely the new personal learning technologies will replace parts of universities rather than the whole institution. Universities will use automated learning environments for beginners and advanced beginners, and offer more mentored learning for embodied skills beyond competent.

Completely automated environments will be very attractive for beginners and advanced beginners, but will impose important limitations in return for their lower costs. The system's orientation toward efficiency would encourage students to take only the courses or certificates they need and "test out" of others, foregoing the opportunity to explore unknown domains in preparation for personal development and innovation. The system's use of standards would force many students into conformity with the standards rather than to develop their individual talents and selves. Students would have few opportunities to join mentored conversations of exploration of new domains where there is yet no knowledge, no concept map, and no criteria of performance.

Much of the motivation for automated education comes from the practical training model itself.

Much of the motivation for automated education comes from the practical training model itself. Automated environments can offer a faster, cheaper path to basic competence in known areas where jobs are available. But new areas are constantly emerging in short periods, often less than the four-year time at a university. To keep up, professionals will need the skills of detecting emerging areas, appropriating their interpretations and practices, navigating from where they are to the new areas they choose, and mobilizing their networks to shape new offers in those areas.[3] Where will they get these skills?

Here are glimpses of what education can offer to enable us to attain those skills in our constantly shifting, highly connected, technology-accelerated world[6]:

- Skills of detecting, navigating, appropriating, offering, and mobilizing.
- An understanding of how social power—political, economic, ideological, military, and trust—works so that we can detect coming changes and develop personal power to navigate and shape them.
- Cultivation of our "self"—who we are, what we stand for, what our identity in the world is, and how we project into the world.
- An understanding of how moods and emotions affect people's willingness to move and how to orchestrate moods conducive to the changes we seek.
- An opportunity to engage in ongoing conversations exploring new worlds, as part of preparing for innovations and further developing our selves.
- Access to mentors and education offers to help in times of transition, such as loss of job, learning a new skill set, or adapting to retirement.
- Access to all levels of performance up through mastery in our field.

These concerns call for practices, skills, and sensibilities that technologies such as MOOCs and OCBMs cannot provide. People yearn for them but do not know how to ask for them. Educators do not have the language to discuss them and design education that meets them. A future environment that includes masters, mentors, coaches, and students, supported by technologies, might be able to do the job.

My conclusion is that well-designed automated systems may help newcomers become entry-level professionals, but will be unable to provide professional development beyond competence. Automation does not provide an opportunity to develop the "self," an ability to explore unfamiliar worlds of which we lack knowledge, or a path to mastery. C

References

1. Carey, K. *The End of College.* Riverhead Books, 2015.
2. Craig, R. *College Disrupted.* Palgrave Macmillan, 2015.
3. Denning, P. Emergent innovation. *Commun. ACM 58*, 6 (June 2015), 28–31; DOI: 10.1145/2753147.
4. Dreyfus, H. *On the Internet.* Routledge, 2001.
5. Dreyfus, H. *What Computers Can't Do.* Harper & Row, 1972.
6. Flores, F. Report of Consejo Nacional de Innovación para la Competitividad. *Surfing Towards the Future: Chile on the 2025 Horizon* (2013); http://chile-california.org/surfing-towards-the-future-into-counselor-fernando-floress-vision-for-innovation.
7. Perelman, L. *School's Out.* Avon, 1993.
8. Weise, M. and Christensen, C. *Hire Education: Mastery, Modularization, and the Workforce Revolution.* Chistenseninstitute.org, 2014; http://www.christenseninstitute.org/publications/hire/#sthash.qIpP29Fw.dpuf.

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

c The website of the Hyperlearning center (1993–2000), which began as the Center for the New Engineer at George Mason University, is archived at http://denninginstitute.com/pjd/oldcne-home-archive.html.

DOI:10.1145/2699405

Jeffrey D. Ullman

Viewpoint
Experiments as Research Validation: Have We Gone Too Far?

Reconsidering conference paper reviewers' requirements for experimental evidence.

I RECENTLY SUBMITTED a paper to a conference, and when I got the reviews back, I noticed the review form had a question referees are required to answer, about whether the experiments were well carried out, with choices like "believable" and "not believable." The reviewers had a bit of trouble with that question, because my paper had no experiments; it was a paper about computational complexity of MapReduce algorithms. Two of the reviewers said the nonexistent experiments were not believable, which is wrong—you have to see something to disbelieve it. I did not write this Viewpoint to complain about being mistreated; in fact the paper was accepted. However, the existence of this question on the review form convinced me there is something seriously wrong with how computer-science research is being evaluated today.[a]

There was a time when experimental evidence was not assumed necessary for publication. In fact, just the opposite was true: you needed to provide mathematical analysis of your conclusions, often in the form of

IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

a Prior to publication of this Viewpoint, I learned this item on the review form had been changed, and now includes other validation as an alternative to experiments. That change is exactly what I had hoped for when writing this Viewpoint.

proofs that your algorithms did what you claimed and "big-oh" analysis of their performance. For example, long ago we discovered algorithms to sort in $O(n \log n)$ time. These algorithms were analyzed formally, using an appropriate model for the main memory of a computer. We did not have to run the algorithms and plot their running time to know they were better than the obvious $O(n^2)$ algorithms. Experiments were only necessary to address questions such as how many elements must there be before Quicksort really beats Bubblesort. And of course when you ran out of main memory, the model no longer applied and you had an unpredicted increase in running time as you suddenly needed to move data to and from disk. Yet the basic $O(n \log n)$ idea still applies even when you use secondary storage.

Even in a far less fundamental setting than sorting algorithms, we expected to see a mathematical analysis rather than experiments. For example, if you had an idea for a better disk-scheduling algorithm, you would define a dozen or more parameters that represented the physical setup: block size, number of tracks, speed of head movement, and so on. You would develop the formula for performance of your algorithm in terms of these parameters and compare that formula with similar formulas for alternative algorithms.

So how did we get from a world where ideas were evaluated by mathematical analysis to a world where reviewers expect to see experimental results in absolutely every paper? In the time when sorting algorithms were a central issue for computer scientists—in the early to mid-1960s—if you were at a major corporation or school, you would have access to one computer for the whole institution, probably an IBM 7090. If you wanted to do the experiments to show an $O(n \log n)$ algorithm ran faster than an $O(n^2)$ algorithm, you would have to wait two hours for each of the many runs needed to develop the evidence. And your institution probably had much better things to do with the one computer they owned than to let you fool around demonstrating again what you had already demonstrated mathematically.

By the 1980s, however, it was possible for each department, or even each research group to have its own time-shared computer, and by the end of the 1980s, each researcher had one. Experiments could be performed almost instantaneously. This change in what was feasible led many subfields of computer science to start writing papers that included experiments. I am going to address principally research papers in the database community, because it is that area with which I am most familiar. However, there are many fields of CS in which a similar evolution in style has occurred, even if the details differ from field to field. To get an idea of the scope of the transformation, I decided to look at three SIGMOD conference proceedings, from 1970, 1990, and 2010. I examined the published papers to find the fraction that had some sort of experimental evidence for their work. In 1970, the conference was actually called SIGFIDET (File Description and Translation). It had 24 papers, only about half of which seemed to be research papers per se. The others looked more like committee reports. However, only one of the 24 had anything that looked like an experiment. Interestingly, the exception was the famous paper by Bayer and McCreight on B-trees, so draw from that whatever conclusion you like. In 1990, of the 37 research papers, 14 had experiments. By 2010, the SIGMOD conference had grown, so I got lazy and sampled every third research paper. Of the 25 papers I looked at, every one had some form of experiment. I also sampled the industrial papers. Only two out of six had experiments, possibly because companies are often reluctant to release performance measures for their own systems.

It (reliance on experiments) encourages the writing of papers that really should not be written at all, because they are so incremental and specialized that their use in practice is unlikely.

There are many reasons to view the addition of experimental evidence as an improved research methodology. The Viewpoint by Michael Mitzenmacher in this issue of *Communications* addresses arguments in favor of experimental evidence in the context of theoretical papers. In addition, experimental evidence makes sense in many situations, such as the following:

- Sometimes, the problem being solved is much less straightforward to state than "sort n numbers." If so, mathematical expression of an algorithm's performance may involve many different parameters or may hardly be parameterizable at all. Experiments can help quantify performance better than a complex formula, even if the experiments describe performance on only a tiny subset of the cases covered by the formula.
- Other times, the worst-case analysis of an algorithm does not represent the performance on average cases or typical cases. Experiments showing results on selected representative cases can be more informative than worst-case analysis. It may even be the case that Algorithm A beats Algorithm B in the worst case, but loses on typical cases.
- For some problems, there are well thought out benchmarks the community believes represent typical instances of the problem. Experiments that report results on these benchmarks are accepted as realistic representations of performance.
- The value of an algorithm may depend not only on the big-oh running time, but on the constant factor; the latter usually can be determined only by experiment.
- There are appropriate concerns that an analysis may omit critical parameters of a problem. By running an

experiment, we are at least assured the ideas are useful in at least one situation.

However, now it appears the pendulum has swung way too far, to the point where experiments are considered the only way to validate ideas. It is time to restore the balance. Experiments should be used when appropriate, such as in the cases just mentioned. However, ideas that require analysis rather than experiments should be handled appropriately, rather than "justified" by inappropriate and meaningless experiments.

Many good ideas stand on their own, without the need for experiments. Consider the two database ideas that have won the Turing award:[b] the relational model (Ted Codd) and two-phase locking (I know Jim Gray won for many contributions, but this one is, I think, the centerpiece). Neither paper was about experiments. Should we have rejected a paper that said "let's organize data into relations" just because there was no experiment to prove its queries were executable more efficiently? Would we want to reject a paper on two-phase locking because it did not measure experimentally the space required by locking tables?

The argument against reliance on experiments as the only way to justify ideas is not just that some papers do not need experiments. There is real harm being done by the excessive focus on experiments as validation. First, experiments are conducted on particular data or under particular assumptions. They rarely tell you what happens in other situations. In contrast, a proper formal analysis of the resources required by an algorithm applies generally. An interesting case in point was suggested by one of the referees, who points out that while cache-aware algorithms (for example, for join) made sense when they were published, today's hardware is smart enough that these algorithms are not needed, and experiments would demonstrate that fact. While true, we cannot be sure there will not be another evolution of hardware that makes cache-aware algorithms again important, perhaps at another level of the memory hierarchy. By writing a paper with the formal analysis, the ideas are relevant in many different environments, including those not contemplated when the ideas were first formulated.

We should not accept experiments as a substitute for a more careful and general analysis.

But there is a more damaging effect of taking experimental results too seriously. It encourages the writing of papers that really should not be written at all, because they are so incremental and specialized that their use in practice is unlikely. There are many areas of research where the nature of the data can vary greatly, and performance of different algorithms will vary with the data. Think of multidimensional indexes, or clustering, for example. In research areas of this kind, it is very easy to find a special form of data and invent an algorithm that works well in this narrow special case. You then run your experiments on data for which your algorithm is best suited and compare it with others, which—surprise, surprise—do not work as well. But were you to run your algorithm on the common cases, or random cases, you would do less well or not well at all. It does not matter; you can still publish yet another paper about yet another algorithm for doing this or that.

Suppose we expected that a paper describing algorithms should measure their performance in some manner. One possible manner is experimentation, which should be a fair comparison with the best-known algorithms, since simply measuring the running time of your own algorithm says nothing about how good it is. A second way to justify the importance of your ideas is with a big-oh analysis of the worst- or average-case running time. A third way is to offer a proof of optimality of your algorithm, at least according to one relevant measure such as time, space, communication, or whatever measure you can justify as the "cost" of executing the algorithm.

It is time to recenter the pendulum. So I propose that, as reviewers of submissions to conferences or journals, we should start by asking whether the value of the proposed ideas have been analyzed for the general case or at least some well-defined realistic subset of all possible cases. We should not accept experiments as a substitute for a more careful and general analysis, unless there really is no way to parameterize the input space suitably. And we should not accept experiments on contrived, specialized data under almost any circumstances. As authors, we should stop thinking of experiments as a substitute for analysis and deep understanding of why our algorithms work better than others that have been proposed. A little self-censorship might be a benefit to the community as well. Not every algorithm that works in some narrow window has to be published. And of course questions for reviewers about experiments should be replaced by a broader question about whether the value of the work has been justified in some appropriate manner. C

Jeffrey D. Ullman (ullman@gmail.com) is the Stanford W. Ascherman Professor of Computer Science (Emeritus) at Stanford University, Stanford, CA.

b This Viewpoint was written before the 2014 Turing Award to Mike Stonebraker was announced. However, the same arguments could be applied to most of the achievements in the database arena upon which that award was based.

DOI:10.1145/2699413

Michael Mitzenmacher

Viewpoint
Theory Without Experiments: Have We Gone Too Far?

Seeking a better understanding of computing through a mixture of theory and appropriate experimental evidence.

I RECENTLY SUBMITTED a paper to a conference, and when I got the reviews back, I noticed a reviewer had asked me specifically to take out the experiments (in this case, simulations) that I had presented partly as motivation, and partly to demonstrate my results. The conference was, unsurprisingly, a *theory* conference; according to the reviewer the experiments were not only unnecessary, they detracted from the paper.

Jeffrey Ullman's Viewpoint in this issue of *Communications* observes that many computer science areas are overly focused on experiments as validation for their work. I do not disagree; he raises several great points. My own experience on several program committees for systems conferences, as well as an author for papers for such conferences, is that experiments are, with very rare exception, a de facto requirement. How can you know if it is really a good idea, the thinking goes, unless there are experiments to back it up? Indeed, in some settings, even simulations are called into question; experiments should be done in real, deployed systems, as simulations can oversimplify what goes on in the real world. As Ullman points out, there are gaping problems with this framework. In particular, we run the risk of losing strong ideas because they do not fall

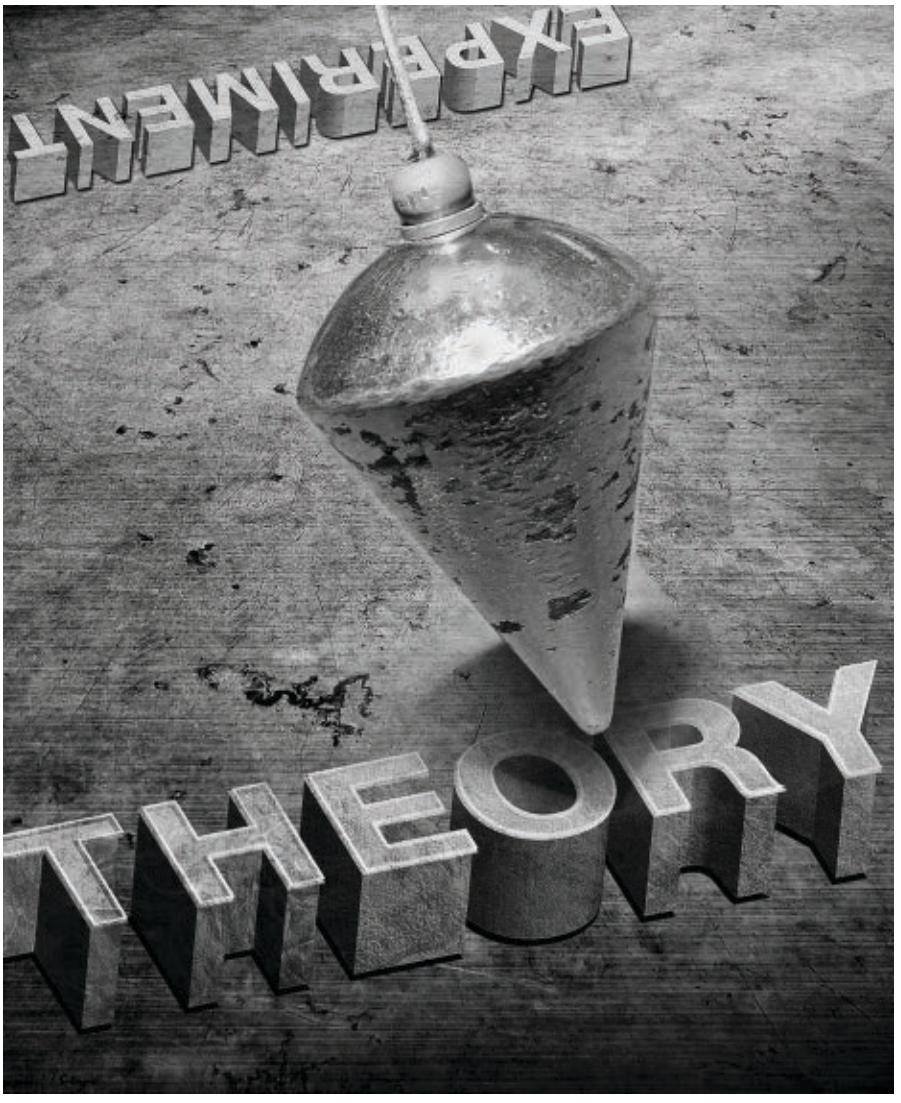


IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

into a framework where it is natural to build a system around them.

But when I put on my theory hat, I find similar fervor, except the sign has changed. In many theory conferences, and perhaps most so in the top ones, experiments are rarely seen. While they are not expressly forbidden, anecdotally I am not the only one who has received reviews where the very idea of including experimental results was called into question, even for algorithmic work. In my opinion, this mind-set represents a dramatic change from when I started out doing research, although I believe the shift has occurred slowly and gradually over time, so that people have not really noticed. And I worry that, in the theory community, the pendulum has now swung too far against experiments. In part, I simply believe that many theoretical results can be greatly enhanced by including proper experimentation, both in settings where the work is naturally tied to practice and where it is not. I also worry the theory community is separating itself further from the rest of computer science, to its own detriment.

To be clear, I understand not all theory papers require experiments. For example, almost all papers providing lower bounds would not need them, nor would many purely mathematical papers. But I think they are more useful than many people believe. A short, incomplete list of reasons to have experiments includes the following.

Demonstrating the potential for practice. This first reason to provide experimental results is arguably the important one: to show the potential utility of ideas for practice. I would argue that ideally a significant amount of theoretical work would make its way, directly or indirectly, into computer science applications. Providing experimental results—or, possibly, even code—should ease the path for practitioners. The best running times and the best results need not be expected; those can be improved later, and theoreticians should not be expected to be the top coders. However, even showing the ideas can be implemented and run successfully in reasonable time will encourage system builders to examine them more closely, and more readily lead to their adoption.

Writing about the experiments can help others understand how theoretical results were developed.

Improving real-world computer applications is not the *only* goal of theoretical computer science (TCS), but it should be a major goal. Otherwise, TCS runs the risk of isolating itself from the rest of the field. Other computer scientists may come to view theoretical work as largely unhelpful or unrelated to their goals, which would seem to relegate TCS to a subfield of mathematics rather than of computing. While some researchers may think that would be fine, I think the vibrant activity of TCS stems in large part from its connection to the real-world problems from the rest of computer science.

Offering verification of one's work. The conception that experiments help validate one's work can be uncomfortable for those with a mathematical bent. The proofs of theorems constitute the results, and they supposedly speak for themselves. But, as we all know, proofs can contain subtle errors. I once asked a student to run a simulation for a published paper studying a problem on Markov chains, where the authors proved a result about some value at equilibrium—I do not recall what, but say it was 0.212.... The student's simulation showed a result of 0.106.... It turns out a factor of 2 had somehow not been canceled somewhere along the way, and this mistake had not been noticed by the authors, and through several reviews. While in context this was not really a serious error, there are many other examples I know of where some simple simulation experiments would have caught problems before publication. And in some cases, the errors were much more serious.

While such computations generally cannot completely verify that a theoretical result is correct, they can dem-

Calendar of Events

September 1–3
ACM SIGPLAN International Conference on Functional Programming,
Vancouver, BC,
Sponsored: ACM/SIG,
Contact: Kathleen S. Fisher,
Email: kathleen.fisher@gmail.com

September 1–4
HT '15: 26th ACM Conference on Hypertext and Social Media,
Guzelyurt, TRNC Cyprus
Sponsored: ACM/SIG,
Contact: Yeliz Yesilada,
Email: yeliz.yesilada@manchester.ac.uk

September 7–11
MobiCom'15: The 21th Annual International Conference on Mobile Computing and Networking,
Paris, France,
Sponsored: ACM/SIG,
Contact: G. Pau,
Email: giovanni.pau@lip6.fr

September 7–11
Ubicomp '15: The 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing,
Osaka, Japan,
Sponsored: ACM/SIG,
Contact: Kenji Mase,
Email: mase@is.nagoya-u.ac.jp

September 13–14
SAP '15: ACM Symposium on Applied Perception 2015,
Tübingen, Germany,
Sponsored: ACM/SIG,
Contact: Laura Cristina Trutoiu,
Email: auract@gmail.com

September 16–20
Ninth ACM Conference on Recommender Systems,
Vienna, Austria,
Sponsored: ACM/SIG ,
Contact: Hannes Werthner
Email: hannes.werthner@ec.tuwien.ac.at

September 21–22
NANOCOM' 15: ACM The Second Annual International Conference on Nanoscale Computing and Communication,
Boston, MA,
Sponsored: ACM/SIG,
Contact: Sasitharan Balasubramaniam
Email: sasi.bala@tut.fi

onstrate that the results are not obviously wrong. In many cases, they are relatively simple to set up. As a reader, I appreciate seeing that this check has been performed.

Understanding whether results are tight or not. In many algorithmic papers, while both upper and lower bounds on the running time of algorithm might be presented, the two bounds might not be the same order asymptotically. Where does the true bound lie? Again, experiments might not be able to provide an answer—perhaps the worst case is not seen in natural examples—but they can provide guidance or lead to conjectures on where the true running time lies.

Even if one has the right asymptotics, theoretical analysis is usually very loose in considering the constant factors. In some cases, constant factors from proofs can be so large that it seems the algorithm would never be practical. But even if the analysis is not tight, the algorithm might still be suitable for practice. More generally, there might be a big difference between a constant factor of 10 and a constant factor of 100. An order of magnitude may not matter in the asymptotic analysis of an algorithm, where theorists typically ignore constant factors in the running time, but it can make a significant difference in other contexts. Experiments can help give insight into such performance issues.

Approximation algorithms provide another example of where experiments can provide insight on the tightness of theoretical analysis. For approximation algorithms, the output provides a solution that is close to optimal, and theoretical analysis bounds how far off the answer can be. As with running time, experiments can provide guidance as to how close the theoretical analysis is to the true approximation bound. Of course, experiments can also show how useful the approximation algorithm will be in practical situations.

Spurring insight into further or future problems. Experimental results can push the imagination into new directions. In several recent papers, my co-authors and I executed simulations, and the insights we gained from them led us to improve and refine our theoretical results. In this case, the results helped us *before* publishing the paper. But it stands to reason that some readers, in seeing experimental results in the final paper, might themselves see something interesting that might lead them to consider an important related or new question. Some people might argue that including such simulations is not needed when writing the paper, as the simulations have done their job, and the theory can and should stand on its own. I disagree. Writing about the experiments can help others understand how the theoretical results were developed, develop experiments of their own when needed, and start exploring extensions or related problems.

Theoreticians inclined toward more practical problems have found some relief by publishing outside of theoretical conferences.

Several of the preceding arguments make the case that experiments in theoretical papers can be useful even when the work is not necessarily motivated by practice. But practical significance is, still, very important. A scientific community ultimately reveals, through its decisions on such things as paper acceptances and awards, how it values empirical results and practical significance (real or potential). I think, today, the TCS community undervalues these, and I do not see this as the best course forward for TCS in the long run.

Theoreticians inclined toward more practical problems have found some relief by publishing outside of theoretical conferences, in venues that appear to be more interested in this kind of work. Conferences such as the World Wide Web conference (WWW), the International Conference of Machine Learning (ICML), and the IEEE Conference on Computer Communications (INFOCOM) take a number of papers that mix theory and experimental results, to great effect. Such conferences can provide a more direct way of connecting theoreticians with interested practitioners. It is not clear this approach is the best solution to the question of how to treat such research, as they reinforce the separation of the larger theory community from our more practical colleagues. But for now these conferences provide outlets for some research that combines theory and practice.

There are also positive signs that many theorists wish to stay connected with practice. Workshops regularly appear that emphasize the utility of empirical work to theoretical computer science and try to bridge the theory-practice divide.[a] Arguably, however, these workshops cannot do enough on their own to counteract larger trends.

To the extent that various subcommunities harden their views toward experiments, thinking of them as requirements as Ullman describes or potentially as just distractions in more theoretical areas, this rigidity and lack of openness does not seem healthy for computer science. The relatively rapid cross-fertilization of ideas between theory and practice has historically been a great strength of computer science research. Increased specialization as our field matures may make such cross-fertilization more difficult. But the goal of gaining a deeper understanding of computing through a suitable mix of both theoretical understanding and well-designed experiments should be a goal we all work toward together. C

a Recent examples include workshops on specific themes such as Satisfiability solvers (http://www.birs.ca/events/2014/5-day-workshops/14w5101), as well as workshops on more general themes such as Algorithms in the Field (https://sites.google.com/site/algorithmsinthefield/) or going Beyond Worst Case Analysis (http://theory.stanford.edu/~tim/bwca/bwca.html).

Michael Mitzenmacher (michaelm@eecs.harvard.edu) is a professor of computer science in the School of Engineering and Applied Sciences at Harvard University, Cambridge, MA, and was area dean of computer science from July 2010 to June 2013.

Kathryn S. McKinley/David S. Rosenblum

Point/Counterpoint
The Pros and Cons of the 'PACM' Proposal

On p. 5 of this issue, ACM Publications Board co-chairs Joseph A. Konstan and Jack W. Davidson introduce a proposal that would interweave conference and journal publishing. Here, computer scientists Kathryn S. McKinley and David S. Rosenblum argue for and against the proposal.

DOI:10.1145/2811406

Point: Kathryn S. McKinley

Make no mistake, computer science research is a resounding success. Its research advances, peer reviewed and published in conferences, created new billion-dollar industries, changing how we do science, business, government, entertain ourselves, and communicate.

While many universities accommodate the rigorous, conference publication culture of CS, others do not. Our practices differ from other sciences that only peer review articles in journals, using conferences for unreviewed talks, posters, and articles.

I recommend the ACM Publications Board proposal because it articulates best principles for research publication—rigorous peer reviewing and no page restrictions—yet is flexible enough to accommodate our conference culture. Adopting it will further improve our process and recognize our archival quality conferences as such, harmonizing computer science with the international science community.

Computer scientists use a conference reviewing process with annual deadlines to combine peer review with timely publication (typically five to seven months from submission to publication). Compared to journal reviewing, I believe our conference process has distinct advantages that this proposal maintains.

"I recommend the proposal because it articulates best principles for research publication, yet is flexible enough to accommodate our conference culture."

ACM conferences are organized by Steering Committees, with general and program chairs and SIG representatives. Program chairs generally serve once for one year. They select Program Committee (PC) members, who each review 10 to 30 submissions and act as editors, choosing accepted papers. Because of increasing numbers of submissions, many conferences have an additional committee with fewer responsibilities. They each review 1 to 10 papers and do not attend the PC meeting. When well deployed, these reviewers deliver highly expert reviews. The most important job of program chairs and journal editors is the same—to find expert reviewers, going beyond the committee and associate editors as necessary. While individual reviews are blinded, the committees are publicly acknowledged. Reviewers are known to each other, increasing accountability.

Because reviewers and leadership change regularly and every submission is reviewed, no one editor or associate editor exerts influence and scientific biases for many years or by desk rejecting submissions. Consequently, conferences are more likely to include diverse problems and approaches.

A disadvantage of conference reviewing is page-limiting submissions to control reviewer workload, which may cause omission of material, such as proofs and methodology. With this proposal, reviewers can judge page-limited submissions, but require additional material, reviewing it or not, as appropriate. Concision is a virtue that reviewers may also require. With one to three months to revise accepted papers (and more for rejected submissions), I think the result will be articles with appropriate content without overburdening reviewers.

Conferences often produce more reviews than journals, providing authors

richer feedback from diverse points of view. Submissions to conferences, such as SIGGRAPH, SOSP, ICSE, and OOPSLA, receive four to six reviews in rounds—papers that remain under consideration are assigned more reviewers. This process limits reviewer workload, while accommodating the growing numbers of submissions many conferences are experiencing.

The conference process fosters our research communities; a benefit I believe is underappreciated. For instance, in-person PC meetings structure research discussions on problem selection, contributions, approaches, methodology, and other scientific values. Committees include and train fresh Ph.D's. Every researcher, from a first-time PC member to the most famous in the field, has an equal opportunity to express scientific viewpoints. A common experience after your first PC meeting, which I distinctly remember, is that young researchers feel respected. What better way to welcome researchers to our scholarly community?

Program committee meetings also build cross-institution relationships at all levels. Reviewing alone in your office does not foster community.

One remaining issue is citations. Many ACM conferences have very highly regarded and established brands. Unfortunately, the ACM Digital Library ambiguously specifies how to cite some of them. For example, see the Export Formats for PLDI, ISCA, and older SIGMOD papers. They have two DOI identifiers, one conference and one SIG Notices. ISI indexes SIG Notices, so some authors prefer them. This duplication is a historical accident due to SIGs that made conference proceedings issues of their SIG Notices as a member benefit, producing duplicate, ambiguous, and non-branded bibliographic entries. Worse, some conference names were not standardized from year to year. I strongly recommend that ACM work with the SIGS to develop citation formats that ensure the bibliographic record avoids confusion, the new citation clearly features conference brand names, and consistently apply this citation format retrospectively and thenceforth. C

Kathryn S. McKinley (mckinley@microsoft.com) is a principal researcher at Microsoft Research, Redmond, WA.

DOI:10.1145/2811410

Counterpoint: David S. Rosenblum

ACM HAS BEEN a longtime leader and innovator in the ecosystem of computing publications. ACM produces journals and proceedings of the highest quality, and it continually finds ways to address the publishing needs and concerns of the computing research community, most recently with its innovative Author Rights framework. It is therefore with no small amount of trepidation that I write to argue against the creation of a new proceedings-focused journal series.

In their editorial, Konstan and Davidson identify two fundamental problems motivating the need for a new journal series. First, the merit and funding systems of some countries do not accord conference publications the status they deserve. Second, conference publication has hard deadlines, page limits and limited review, and it exhausts the reviewing capacity of the best researchers. I concede these problems exist to some extent. I also concede the proposed journal series, by virtue of being journals, might help circumvent some of the problems. However, I feel the proposed journal series fails to attack the root causes of these problems.

With respect to the first problem, I feel it is counterproductive for our community to distort itself merely in order to live up to a false image devised by non-experts who refuse to understand and accept the way our community works. (As Emerson wrote, "to be yourself in a world that is constantly trying to make you something else is the greatest accomplishment.") Konstan and Davidson rightly note the NRC and CRA have been very successful in educating U.S. institutions (and many international institutions, I would add) on the importance of conference publications.[a] Thus, rather than create a new journal series, a more natural way to attack this problem is for ACM, as the world's leading professional society in computing, to lobby vociferously on the community's behalf about the importance of conference publications.

"I feel it is counterproductive for our community to distort itself merely in order to live up to a false image devised by non-experts who refuse to understand and accept the way our community works."

In addition, I feel it is overly simplistic to say improper recognition of conference papers is a problem of countries. The reality is far more complicated, and it relates less to countries than it does to the relative quality differential between institutions, the degree of enlightenment among institutional administrators, the additional expectations placed on researchers at some institutions (such as explicit annual publication quotas), and the level of readiness (and unreadiness) of authors to publish to ACM's high standard of quality.

Regarding the second problem, several subcommunities have been experimenting successfully with approaches to marrying conferences with

a In fact, I would argue we find ourselves in our present predicament of dying journals and overloaded conferences in part because of that success.

journals in a way that mitigates some of the problems of conference publication. SIGGRAPH/ACM TOG and HiPEAC/ACM TACO are two prominent examples in the ACM family, and ACM TOSEM and IEEE TSE are embarking on a "journal-first" initiative in conjunction with some of the major software engineering conferences. These examples show the problem can be addressed with *existing* journals, making it unclear what additional benefits this new journal series would bring. Even so, tying journal publication to annual conference cycles arguably still encourages an annual cycle of submissions from authors (particularly authors operating under quotas), even when the related journal and conference do away with fixed deadlines (as in the case of VLDB).

One big problem not mentioned by the co-chairs is the continued proliferation of publication venues in our community. A new journal series will only further exacerbate this problem, without any obvious compensating benefit.

But all these problems are closely bound up with many additional drivers of the difficulties we face, including increasing pressure on researchers to produce, and increasing demands from research-funding agencies to plan for and quantify impact before new research even begins. Conferences now routinely receive numerous submissions from the same author in the same year (with one author having submitted 11 papers to a recent software engineering conference), which begs the question of just who exactly is complaining that conferences are too deadline-driven!

I am a strong believer in the power of incentives. As a community, we need to step back, return to first principles, and decide for ourselves what behavior we desire, and then develop appropriate incentives to produce that behavior. Do we want to encourage more journal publication? Do we want to devalue the esteem associated with conferences and conference reviewing? These are things we need to examine deeply before creating yet another series of journals that need to be nurtured, grown, and sustained. C

David S. Rosenblum (david@comp.nus.edu.sg) is a professor of computer science and Dean of the School of Computing at the National University of Singapore.

DOI:10.1145/2767151

Article development led by acmqueue
queue.acm.org

Old questions being answered with both AI and HCI.

BY SPENCE GREEN, JEFFREY HEER, AND CHRISTOPHER D. MANNING

Natural Language Translation at the Intersection of AI and HCI

THE FIELDS OF artificial intelligence (AI) and human-computer interaction (HCI) are influencing each other like never before. Widely used systems such as Google Translate, Facebook Graph Search, and RelateIQ hide the complexity of large-scale AI systems behind intuitive interfaces. But relations were not always so auspicious. The two fields emerged at different points in the history of computer science, with different influences, ambitions, and attendant biases. AI aimed to construct a rival, and perhaps a successor, to the human intellect. Early AI researchers such as McCarthy, Minsky, and Shannon were mathematicians by training, so theorem-proving and formal models were attractive research directions. In contrast, HCI focused more on empirical approaches to usability and human factors, both of which generally aim to make machines more useful to humans. Many attendees at the first CHI conference in 1983 were psychologists and engineers. Presented papers had titles such as "Design Principles for Human-Computer Interfaces" and "Psychological Issues in the Use of Icons in Command Menus," hardly appealing fare for mainstream AI researchers.

Since the 1960s, HCI has often been ascendant when setbacks in AI occurred, with successes and failures in the two fields redirecting mindshare and research funding.[14] Although early figures such as Allen Newell and Herbert Simon made fundamental contributions to both fields, the competition and relative lack of dialogue between AI and HCI are curious. Both fields are broadly concerned with the connection between machines and intelligent human agents. What has changed recently is the deployment and adoption of user-facing AI systems. These systems need interfaces, leading to natural meeting points between the two fields.

Nowhere is this intersection more apropos than in natural language processing (NLP). Language translation is a concrete example. In practice, professional translators use suggestions from machine aids to construct final, high-quality translations. Increasingly, human translators are incorporating the output of machine translation (MT) systems such as Google Translate into their work. But how do we go beyond simple correction of machine mistakes? Recently, research groups at Stanford, Carnegie Mellon, and the European CasmaCat consortium have been investigating a human-machine model like that shown in Figure 1.

For the English input "Fatima dipped the bread," the baseline MT

ILLUSTRATION BY STUART BRIERS


a
a
s
wd
g
k
h
q
Ф
ق
Д
Х
Л
m
ك
و
t
Ё
अ
o
r
ش
В
ج
Q
у

system proposes the Arabic translation غمس فاطمة الخبز, but the translation is incorrect because the main verb غمس (in red) has the masculine inflection. The user corrects the inflection by adding an affix ت, often arriving at a final translation faster than she would have on her own. The corrections also help the machine, which can update its model to produce higher-quality suggestions in future sessions. In this positive feedback loop, both humans and machines benefit, but in complementary ways. To realize this *interactive machine translation system*, both interfaces that follow HCI principles and powerful AI are required.

What is not widely known is this type of system was first envisioned in the early 1950s and developments in translation research figured significantly in the early dialogue between AI and HCI. The failed dreams of early MT researchers are not merely historical curiosities, but illustrations of how intellectual biases can marginalize pragmatic solutions, in this case a human-machine partnership for translation. As practicing AI and HCI researchers, we have found the conversation today has many of the same features, so the historical narrative can be instructive. In this article, we first recount that history. Then we summarize the recent breakthroughs in translation made possible by a healthy AI-HCI collaboration.

A Short History of Interactive Machine Translation

Machine translation as an application for digital computers predates computational linguistics and artificial intelligence, fields of computer science within which it is now classified. The term artificial intelligence first appeared in a call for participation for a 1956 conference at Dartmouth College organized by McCarthy, Minsky, Rochester, and Shannon. But by 1956, MT was a very active research area, with the 1954 Georgetown MT demonstration receiving widespread media coverage. The field of computational linguistics grew out of early research on machine translation. MT research was oriented toward cross-language models of linguistic structure, with parallel theoretical developments by Noam Chomsky in generative linguistics exerting some influence.[21]

The stimuli for MT research were the invention of the general-purpose computer during World War II and the advent of the Cold War. In an oft-cited March 1947 letter, Warren Weaver—a former mathematics professor, then director of the Natural Sciences division at the Rockefeller Foundation—asked Norbert Wiener of the Massachusetts Institute of Technology (MIT) about the possibility of computer-based translation:

Recognizing fully...the semantic difficulties because of multiple meanings, etc., I have wondered if it were unthinkable to a computer which would translate...one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."

Wiener's response was skeptical and unenthusiastic, ascribing difficulty to the extensive "connotations" of language. What is seldom quoted is Weaver's response on May 9th of that year. He suggested a distinction between the many combinatorial possibilities with a language and the smaller number that are actually used:

It is, of course, true that Basic [English] puts multiple use on an action verb such as get. But even so, the two-word combinations such as get up, get over, get back, etc., are, in Basic, not really very numerous. Suppose we take a vocabulary of 2,000 words, and admit for good measure all the two-word combinations as if they were single words. The vocabulary is still only four million: and that is not so formidable a number to a modern computer, is it?

("Basic English" was a controlled language, created by Charles Kay Ogden as a medium for international exchange that was in vogue at the time.)

Weaver was suggesting a distinction between theory and use that would eventually take root in the empirical revolution of the 1990s: an imperfect linguistic model could suffice given enough data. The statistical MT techniques described later are in this empirical tradition.

Use Cases for Machine Translation

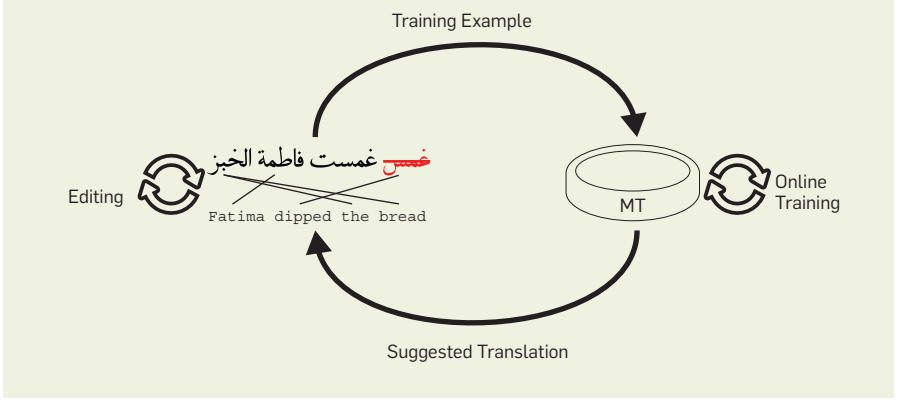
By 1951 MT research was under way, and Weaver had become a director of the National Science Foundation (NSF). An NSF grant—possibly under the influence of Weaver—funded the appointment of the Israeli philosopher Yehoshua Bar-Hillel to the MIT Research Laboratory of Electronics.[19] That fall Bar-Hillel toured the major American MT research sites at the University of California–Los Angeles, the RAND Corporation, U.C. Berkeley, the University of Washington, and the University of Michigan–Ann Arbor. He prepared a survey report[1] for presentation at the first MT conference, which he convened the following June.

That report contains two foundational ideas. First, Bar-Hillel anticipated two use cases for "mechanical translation." The first is *dissemination*:

One of these is the urgency of having foreign language publications, mainly in the fields of science, finance, and diplomacy, translated with high accuracy and reasonable speed...[1]

The dissemination case is distinguished by a desired quality threshold. The other use case is assimilation:

Figure 1. Interactive language translation.



Another is the need of high-speed, though perhaps low-accuracy, scanning through the huge printed output.[1]

Bar-Hillel observed the near-term achievement of "pure MT" was either unlikely or "achievable only at the price of inaccuracy." He then argued in favor of *mixed MT*, "a translation process in which a human brain intervenes." As for where in the pipeline this intervention should occur, Bar-Hillel recommended:

...the human partner will have to be placed either at the beginning of the translation process or the end, perhaps at both, but preferably not somewhere in the midst of it ...[1]

He then went on to define the now familiar terms *pre-editor*, for intervention prior to MT, and *post-editor* for intervention after MT. The remainder of the survey deals primarily with this pre- and post-editing, showing a pragmatic predisposition that would be fully revealed a decade later. Having established terms and distinctions still in use today, Bar-Hillel returned to Israel in 1953 and took a hiatus from MT.[21]

In 1958 the U.S. Office of Naval Research commissioned Bar-Hillel to conduct another survey of MT research. That October he visited research sites in the U.S. and Britain, and collected what information was publicly available on developments in the Soviet Union. A version of his subsequent report circulated in 1959, but the revision published in 1960 attracted greater attention.

Bar-Hillel's central argument in 1960 was that preoccupation with "pure MT"—his label for what was then called fully automatic high quality translation (FAHQT)—was "unreasonable" and that despite claims of imminent success, he "could not be persuaded of their validity." He provided an appendix with a purported proof of the impossibility of FAHQT. The proof was a sentence with multiple senses (in italics) in a simple passage that is difficult to translate without extra-linguistic knowledge ("Little John was looking for his toy box. Finally he found it. *The box was in the pen*"). Some 54 years later, Google Translate cannot translate this sentence correctly for many language pairs.

Bar-Hillel outlined two paths forward: carrying on as before, or favoring some "less ambitious aim." That less ambitious aim was mixed MT:

As soon as the aim of MT is lowered to that of high quality translation by a machine-post-editor partnership, the decisive problem becomes to determine the region of optimality in the continuum of possible divisions of labor.[2]

Increasingly, human translators are incorporating the output of machine translation systems such as Google Translate into their work. But how do we go beyond simple correction of machine mistakes?

Bar-Hillel lamented "the intention of reducing the post-editor's part has absorbed so much of the time and energy of most workers in MT" that his 1951 proposal for mixed MT had been all but ignored. No research group escaped criticism. His conclusion presaged the verdict of the U.S. government later in the decade:

Fully automatic, high quality translation is not a reasonable goal, not even for scientific texts. A human translator, in order to arrive at his high quality output, is often obliged to make intelligent use of extra-linguistic knowledge which sometimes has to be of considerable breadth and depth.[2]

By 1966 Bar-Hillel's pessimism was widely shared, at least among research backers in the U.S. government, which drastically reduced funding for MT research as recommended by the ALPAC report. Two passages concern post-editing, and presage the struggles that researchers in decades to come would face when supplying humans with machine suggestions. First:

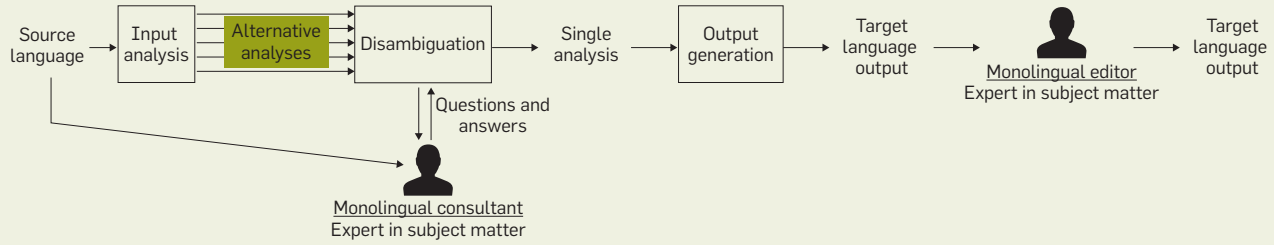
...after 8 years of work, the Georgetown University MT project tried to produce useful output in 1962, they had to resort to post-editing. The post-edited translation took slightly longer to do and was more expensive than conventional human translation. [27]

Also cited was an article by Robert Beyer of the Brown University physics department, who recounted his experience post-editing Russian-English machine translation. He said:

I must confess that the results were most unhappy. I found that I spent at least as much time in editing as if I had carried out the entire translation from the start. Even at that, I doubt if the edited translation reads as smoothly as one which I would have started from scratch.[3]

The ALPAC report concluded that two decades of research had produced systems of little practical value that did not justify the government's level of financial commitment. Contrary to the popular belief the report ended MT research,

Figure 2. The MIND system.



it suggested constructive refocusing on "means for speeding up the human translation process" and "evaluation of the relative speed and cost of various sorts of machine-aided translation."[27] These two recommendations were in line with Bar-Hillel's earlier agenda for machine-assisted translation.

The Proper Role of Machines

The fixation on FAHQT at the expense of mixed translation indicated a broader philosophical undercurrent in the first decade of AI research. Those promoting FAHQT were advocates—either implicitly or explicitly—of the vision that computers would eventually rival and supplant human capabilities. Nobel Laureate Herbert Simon famously wrote in 1960 that "Machines will be capable, within twenty years, of doing any work that a man can do."[29] Bar-Hillel's proposals were in the spirit of the more skeptical faction, which believed machine augmentation of existing human facilities was a more reasonable and achievable goal.

J.C.R. Licklider, who exerted considerable influence on early HCI and AI research,[15] laid out this position in his 1960 paper "Man-Computer Symbiosis,"[24] which is now recognized as a milestone in the introduction of human factors in computing. In the abstract he wrote that "in the anticipated symbiotic partnership, men will set the goals, formulate the hypotheses, determine the criteria, and perform the evaluations." Computers would do the "routinizable work." Citing a U.S. Air Force report that concluded it would be 20 years before AI made it possible "for machines alone to do much thinking or problem solving of military significance," Licklider suggested that human-computer interaction research could be useful in the interim, although that interim might be "10 [years] or 500." Licklider and Bar-Hillel knew each other. Both participated in meetings coincident with the 1961 MIT Centennial (also present were McCarthy, Shannon, and Wiener, among others), where Bar-Hillel directly posed the question, "Do we want computers that will compete with human beings and achieve intelligent behavior autonomously, or do we want what has been called man-machine symbiosis?"[16] He went on to criticize the "enormous waste during the last few years" on the first course, arguing it was unwise to hope for computers that "autonomously work as well as the human brain with its billion years of evolution." Bar-Hillel and Licklider also attended a cybernetics symposium in 1967[17] and a NATO workshop on information science in 1973.[9] The question of how much to expect from AI remained central throughout this period.

Licklider's name does appear in the 1966 ALPAC report that advocated reduction of research funding for FAHQT. After narrating the disappointing 1962 Georgetown post-editing results, the report says two groups nonetheless intended to develop post-editing "services." But "Dr. J.C.R. Licklider of IBM and Dr. Paul Garvin of Bunker-Ramo said they would not advise their companies to establish such a [post-editing] service."[27]

The finding that post-editing translation takes as long as manual translation is evidence of an interface problem. Surely even early MT systems generated some words and phrases correctly, especially for scientific text, which is often written in a formulaic and repetitive style. The question then becomes one of human-computer interaction: how best to show suggestions to the human user.

Later, the human-machine scheme would be most closely associated with Douglas Engelbart, who wrote a lengthy research proposal—he called it a "conceptual framework"—in 1962.[11] The proposal was submitted to Licklider, who was at that time director of the U.S. Advanced Research Projects Agency (ARPA). By early 1963, Licklider had funded Engelbart's research at the Stanford Research Institute (SRI), having told a few acquaintances, "Well, he's [Engelbart] out there in Palo Alto, so we probably can't expect much. But he's using the right words, so we're sort of honor-bound to fund him."[32]

"By augmenting the human intellect," Engelbart wrote, "we mean increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to problems." Those enhanced capabilities included "more-rapid comprehension, better comprehension,... speedier solutions, [and] better solutions."[11] Later on, he described problem solving as abstract symbol manipulation, and gave an example that presaged large-scale text indexing like that done in Web crawling and statistical machine translation:

What we found ourselves doing, when having to do any extensive digesting of journal articles, was to type large batches of the text verbatim into computer store. It is so nice to be able to tear it apart, establish our own definitions, and substitute, restructure, append notes, and so forth, in pursuit of comprehension.[11]

He noted that many colleagues were already using augmented text manipulation systems, and that once a text was entered, the original reference was rarely needed. "It sits in the archives like an orange rind, with most of the real juice squeezed out."[11]

Martin Kay and the First Interactive MT System

By the late 1960s, Martin Kay and colleagues at the RAND Corporation began designing a human-machine translation system, the first incarnation of which was called MIND.[5] Their system (Figure 2), which was never built, included human intervention by monolingual editors during both source (syntactic) analysis and target generation (personal communication with Martin Kay, Nov. 7, 2014).

Figure 2 shows the MIND system.[5] Monolingual pre-editors disambiguate source analyses prior to transfer. Monolingual post-editors ensure target fluency after generation.

MIND was consistent with Bar-Hillel's 1951 plan for pre-editors and post-editors. Kay went further with a 1980 proposal for a "translator's amanuensis," which would be a "word processor [with] some simple facilities peculiar to translation."[22] Kay's agenda was similar in spirit to Bar-Hillel's "mixed MT" and Engelbart's human augmentation:

I want to advocate a view of the problem in which machines are gradually, almost imperceptibly, allowed to take over...First they will take over functions not essentially related to translation. Then, little by little, they will approach translation itself.

Kay saw three benefits of user-directed MT. First, the system—now having the user's attention—would be better able to point out uncertain translations. Second, cascading errors could be prevented since the machine would be invoked incrementally at specific points in the translation process. Third, the machine could record and learn from the interaction history. Kay advocated collaborative refinement of results: "the man and the machine are collaborating to produce not only a translation of a text but also a device whose contribution to that translation is being constantly enhanced."[22] These three benefits would now be recognized as core characteristics of an effective mixed-initiative system.[6,18]

Kay's proposal had little effect on the commercial "translator workbenches" developed and evaluated during the 1980s,[20] perhaps due to limited circulation of his 1980 memo (which would not be published until 1998[23]). However, similar ideas were being investigated at Brigham Young University as part of the Automated Language Processing (ALP) project. Started in 1971 to translate Mormon texts from English to other languages, ALP shifted emphasis in 1973 to machine-assisted translation.[30] The philosophy of the project was articulated by Alan Melby, who wrote that "rather than replacing human translators, computers will serve human translators."[26] ALP produced the Interactive Translation System (ITS), which allowed human interaction at both the source analysis and semantic transfer phases.[26] But Melby found that in experiments, the time spent on human interaction was "a major disappointment," because a 250-word document required about 30 minutes of interaction, which is "roughly equivalent to a first draft translation by a human translator." He drew several conclusions that were to apply to most interactive systems evaluated over the following two decades:

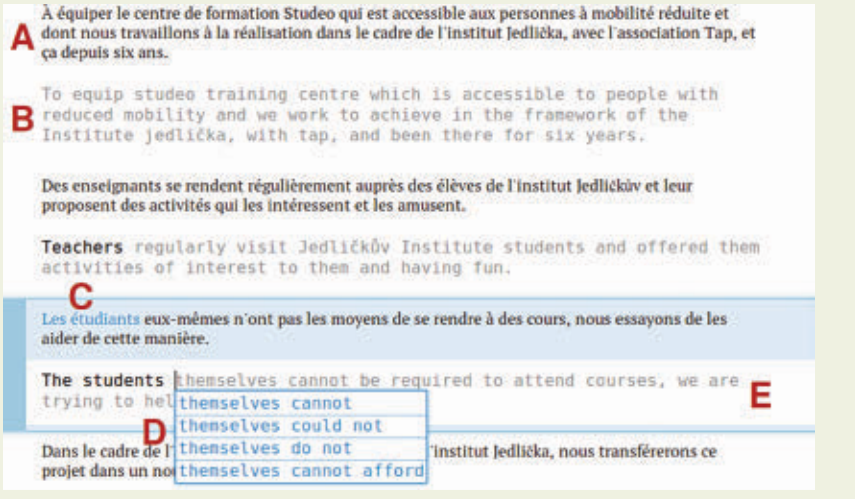
1. ITS did not yet aid the human translator enough to justify the engineering overhead.
2. Online interaction requires specially trained operators, further increasing overhead.
3. Most translators do not enjoy post-editing.

ALP never produced a production system due to "hardware costs and the amount and difficulty of human interaction."[30]

Kay and Melby intentionally limited the coupling between the MT system and the user; MT was too unreliable to be a constant companion. Church and Hovy in 1993 were the first to see an application of tighter coupling,[8] even when MT output was "crummy." Summarizing user studies dating back to 1966, they described post-editing as an "extremely boring, tedious and unrewarding chore." Then they proposed a "superfast typewriter" with an autocomplete text prediction feature that would "fill in the rest of a partially typed word/phrase from context." A separate though related aid would be a "Cliff-note" mode in which the system would annotate source text spans with translation glosses. Both of these features were consistent with their belief that a good application of MT should "exploit the strengths of the machine and not compete with the strengths of the human." The autocomplete idea, in particular, directly influenced the TransType project,[12] the first interactive statistical MT system.

A conspicuous absence in the published record of interactive MT research since the 1980s is reference to the HCI literature. HCI as an organized field came about with the establishment of ACM SIGCHI in 1982 and the convening of the first CHI conference in 1983.[14] *The Psychology of Human-Computer Interaction*, by Card, Moran, and Newell, was also published that year.[7] It is now recognized as a seminal work in the field which did much to popularize the term HCI. Several chapters analyze text editing interactions, drawing con-

Figure 3. The Predictive Translation Memory interface.



clusions that apply directly to bilingual text editing, that is, translation. But we are aware of only two MT papers[4,31] among the thousands in the Association for Computational Linguistics Anthology (up to 2013) that cite an article included in the proceedings of CHI from 1983–2013. (There may be more, but the number is remarkably small.)

In retrospect, the connection between interactive MT and early HCI research is obvious. Kay, Melby, and Church had all conceived of interactive MT as a text editor augmented with bilingual functions. Card et al. identified text editing as "a natural starting point in the study of human-computer interaction," and much of their book treats text editing as an HCI case study. Text editing is a "paradigmatic example" of HCI for several reasons: the interaction is rapid; the interaction becomes an unconscious extension of the user; text editors are probably the most heavily used computer programs; and text editors are representative of other interactive systems.[7] A user-centered approach to translation would start with text entry and seek careful bilingual interventions, increasing the level of support through user evaluation, just as Bar-Hillel and Kay suggested many decades ago.

A user-centered approach to translation would start with text entry and seek careful bilingual interventions, increasing the level of support through user evaluation, just as Bar-Hillel and Kay suggested many decades ago.

Recent Breakthroughs in Interactive MT

All this is not to say fruitful collaboration is absent at the intersection of AI and HCI. The landmark work of Horvitz and colleagues at Microsoft established mixed-initiative design principles that have been widely applied.[18] Bar-Hillel identified the need to find the "region of optimality" between human and machine; Horvitz's principles provide design guidance (distilled from research experiences) for finding that region. New insights are appearing at major human/machine conferences such as UbiComp and HCOMP. And the explosion of data generated by companies has inspired tools such as Tableau and Trifacta, which intelligently assist users in aggregating and visualizing large datasets. However, language applications have largely escaped notice until recently.

When we began working on mixed-initiative translation in 2012, we found that even post-editing had a mixed experimental record. Some studies found it increased translator productivity, while others showed the classic negative results. At CHI 2013, we presented a user study on post-editing of MT output for three different language pairs (English to Arabic, French, and German). The between-subjects design was common to HCI research yet rare in NLP, and included statistical analysis of time and quality that controlled for post-editor variability. The results showed that post-editing conclusively reduced translation time *and* increased quality for expert translators. The result may owe to controlling sources of confound overlooked in previous work, but it may also come from the rapid improvement of statistical MT, which should cause users to revisit their assumptions. For example, to avoid bias, subjects were not told that the suggestions came from Google Translate. However, one subject commented later that

Your machine translations are far better than the ones of Google, Babel and so on. So they were helpful, but usually when handed over Google-translated material, I find it way easier and quicker to do it on my own from unaided.

One of Horvitz's 12 principles is that a mixed-initiative system should learn by observing the user. Recall the top of Figure 1, in which final translations are returned to the MT system for adaptation. Recent improvements in online machine learning for MT have made this old idea possible. Denkowski et al.[10] were the first to show users can detect a difference in quality between a baseline MT system and a refined model adapted to post-edits. The adapted suggestions required less editing and were rated higher in terms of quality than the baseline suggestions. Updating could occur in seconds rather than in the hours-long batch procedures conventionally applied.

These quantitative successes contrast with the qualitative assessment of post-editing observed in many studies: that it is a "boring and tedious chore."[8] Human translators tend not to enjoy correcting sometimes fatally flawed MT output. Previously, we showed richer interactive modes have been built and evaluated, but until none improved translation time or quality rela-

tive to post-editing, a mode considered as long ago as the 1962 Georgetown experiment.

Last year we developed Predictive Translation Memory (PTM, Figure 3), which is a mixed-initiative system in which human and machine agents interactively refine translations. The initial experience is similar to post-editing—there is a suggested machine translation—but as the user begins editing, the machine generates new suggestions conditioned on user input. The translation is collaboratively refined, with responsibility, control, and turn-taking orchestrated by the user interface. The NLP innovations that make this possible are fast search and online parameter learning. The interface design is informed by Horvitz's mixed-initiative guidelines, fundamentals of graphical perception, and the CHI 2013 user study results.

In a user study with professional translators, we found that PTM was the first interactive translation system to increase translation quality relative to post-edit.[13] This is the desired result for the dissemination scenario in which human intervention is necessary to guarantee accuracy. Moreover, we found that PTM produced better training data for adapting the MT system to each user's style and diction. PTM records the sequence of user edits that produce the final translation. These edits explain how the user generated the translation in a machine-readable way data that has not bee available previously. Our current research is investigating how to better utilize this rich data source in a large-scale setting. This is the movitation for one of Horvitz's best-known recommendations for mixed-initiative system design: minimizing the cost of poor guesses about action and timing.[18]

Conclusion

We have shown a human-machine system design for language translation benefits both human users—who produce higher-quality translations—and machine agents, which can refine their models given rich feedback. Mixed-initiative MT systems were conceived as early as 1951, but the idea was marginalized due to biases in the AI research community. The new results were obtained by combining insights from AI and HCI, two communities with similar strategic aims but surprisingly limited interaction for many decades. Other problems in NLP such as question answering and speech transcription could benefit from interactive systems not unlike the one we have proposed for translation. Significant issues to consider in the design of these systems are:

- Where to insert the human efficiently in the processing loop.
- How to maximize human utility even when machine suggestions are sometimes fatally flawed.
- How to isolate and then improve the contributions of specific interface interventions (for example, full-sentence suggestions vs. autocomplete phrases) in the task setting.

These questions were anticipated in the translation community long before AI and HCI were organized fields. New dialogue between the fields is yielding fresh approaches that apply not only to translation, but to other systems that attempt to augment and learn from the human intellect. C

Related articles on queue.acm.org

AI Gets a Brain
Jeff Barr and Luis Felipe Cabrera
http://queue.acm.org/detail.cfm?id=1142067

The Future of Human-Computer Interaction
John Canny
http://queue.acm.org/detail.cfm?id=1147530

A Conversation with Jeff Heer, Martin Wattenberg, and Fernanda Viégas
http://queue.acm.org/detail.cfm?id=1744741

References

1. Bar-Hillel, Y. The present state of research on mechanical translation. *American Documentation 2, 4 (1951), 229–237.*
2. Bar-Hillel, Y. The present status of automatic translation of languages. *Advances in Computers 1* (1960), 91–163.
3. Beyer, R.T. Hurdling the language barrier. *Physics Today 18*, 1 (1965), 46–52.
4. Birch, A. and Osborne, M. Reordering metrics for MT, (2011).
5. Bisbey, R. and Kay, M. The MIND translation system: a study in man-machine collaboration. Technical Report P-4786, Rand Corp, 1972.
6. Carbonell, J. AI in CAI: An artificial-intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems 11*, 4 (1970), 190–202.
7. Card, S.K., Moran, T.P. and Newell, A. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, 1983.
8. Church, K. W. and Hovy, E. Good applications for crummy machine translation. *Machine Translation 8* (1993), 239–258.
9. Debons, A. and Cameron, W.J., Eds. *Perspectives in Information Science*, Vol. 10. NATO Advanced Study Institutes Series. Springer, 1975.
10. Denkowski, M., Lavie, A, Lacruz, I. and Dyer, C. Real time adaptive machine translation for post-editing with cdec and TransCenter. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, (2014).
11. Engelbart, D.C. *Augmenting human intellect: A conceptual framework.* Technical report, SRI Summary Report AFOSR-3223 (1962).
12. Foster, G., Langlais, P. and Lapalme, G. TransType: text prediction for translators. In *Proceedings of ACL Demonstrations*, (2002), 93-94.
13. Green, S., Wang, S., Chuang, J., Heer, J. Schuster, S. and Manning, C.D. Human effort and machine learnability in computer aided translation. In *Proceedings of EMNLP*, (2014), 1225¬–1236.
14. Grudin, J. AI and HCI: Two fields divided by a common focus. *AI Magazine 30*, 4 (2009, 48–57.
15. Grudin, J. A moving target—the evolution of human-computer interaction. *Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications* (3rd edition). J.A. Jacko, Ed. CRC Press, 2012.
16. Hauben, M. and Hauben, R. *Netizens: On the History and Impact of Usenet and the Internet.* IEEE Computer Society Press, Los Alamitos, CA, 1997.
17. Hauben, R. Heinz von Foerster, Margaret Mead and JCR Licklider and the conceptual foundations for the Internet: The early concerns of cybernetics of cybernetics. Presentation in Berlin Germany (Nov. 16, 2003).
18. Horvitz, E. Principles of mixed-initiative user interfaces. In *Proceedings of CHI* (May 1999), 15-20.
19. Hutchins, J. From first conception to first demonstration: The nascent years of machine translation, 1947–1954: A chronology. *Machine Translation 12*, 3 (1997), 195–252.
20. Hutchins, J. The origins of the translator's workstation. *Machine Translation 13* (1998), 287–307.
21. Hutchins, J. Yehoshua Bar-Hillel: A philosopher's contribution to machine translation. *Early Years in Machine Translation: Memoirs and Biographies of Pioneers.* W.J. Hutchins, Ed. John Benjamins, 2000.
22. Kay, M. The proper place of men and machines in language translation. Technical Report CSL-80-11, 1980, Xerox Palo Alto Research Center (PARC).
23. Kay, M. The proper place of men and machines in language translation. *Machine Translation 12*, 1/2 (1998). 3–23.
24. Licklider, J.C.R. Man-computer symbiosis. *IRE Transactions on Human Factors in Electronics* HFE1 1 (1960), 4–11.
25. Melby, A.K. Creating an environment for the translator. In *Proceedings of the Third Lugano Tutorial.* M. King, ed. (Switzerland, Apr. 2–7, 1984). Edinburgh University Press, 124–132.
26. Melby, A.K., Smith, M.R. and Peterson, J. ITS: Interactive translation system. In *Proceedings of the Seventh International Conference on Computational Linguistics.* (1980).
27. Pierce, J.R. (ed.) Languages and machines: computers in translation and linguistics. National Research Council Publication 1416. National Academy of Sciences, Washington, D.C., 1966.
28. Sanchis-Trilles, G., Alabau, V., Buck, C., Carl, M., Casacuberta F. and García-Martínez, M., et al. Interactive translation prediction versus conventional post-editing in practice: A study with the CasMaCat workbench. *Machine Translation 28*, 3/4 (2014), 1–19.
29. Simon, H. A. *The New Science of Management Decision.* Harper, N.Y., 1960.
30. Slocum, J. A survey of machine translation: its history, current status, and future prospects. *Computational Linguistics 11*, 1 (1985), 1–17.
31. Somers, H. and Lovel, H. Computer-based support for patients with limited English. *EAMT Workshop on MT and Other Language Technology Tools.* (2003).
32. Waldrop, M.M. *The Dream Machine: J.C.R. Licklider and the Revolution That Made Computing Personal.* Viking, N.Y. 2001.

Spence Green is a co-founder of Lilt, a provider of interactive translation systems.

Jeffrey Heer is an associate professor of computer science and engineering at the University of Washington, where he directs the Interactive Data Lab.

Christopher D. Manning is a professor of computer science and linguistics at Stanford University.

DOI:10.1145/2776756

Article development led by acmqueue
queue.acm.org

Testing a distributed system can be trying even under the best of circumstances.

BY PHILIP MADDOX

Testing a Distributed System

DISTRIBUTED SYSTEMS CAN be especially difficult to program for a variety of reasons. They can be difficult to design, difficult to manage, and, above all, difficult to test. Testing a normal system can be trying even under the best of circumstances, and no matter how diligent the tester is, bugs can still get through. Now take all of the standard issues and multiply them by multiple processes written in multiple languages running on multiple boxes that could potentially all be on different operating systems, and there is potential for a real disaster.

Individual component testing, usually done via automated test suites, certainly helps by verifying that each component is working correctly. Component testing, however, usually does not fully test all of the bits of a distributed system. Testers need to be able to verify that data at one end of a distributed system makes its way to all of the other parts of the system and, perhaps more importantly, is visible to the various components of the distributed system in a manner that meets the consistency requirements of the system as a whole.

This article discusses general strategies for testing distributed systems as well as specific strategies for testing distributed data storage systems.

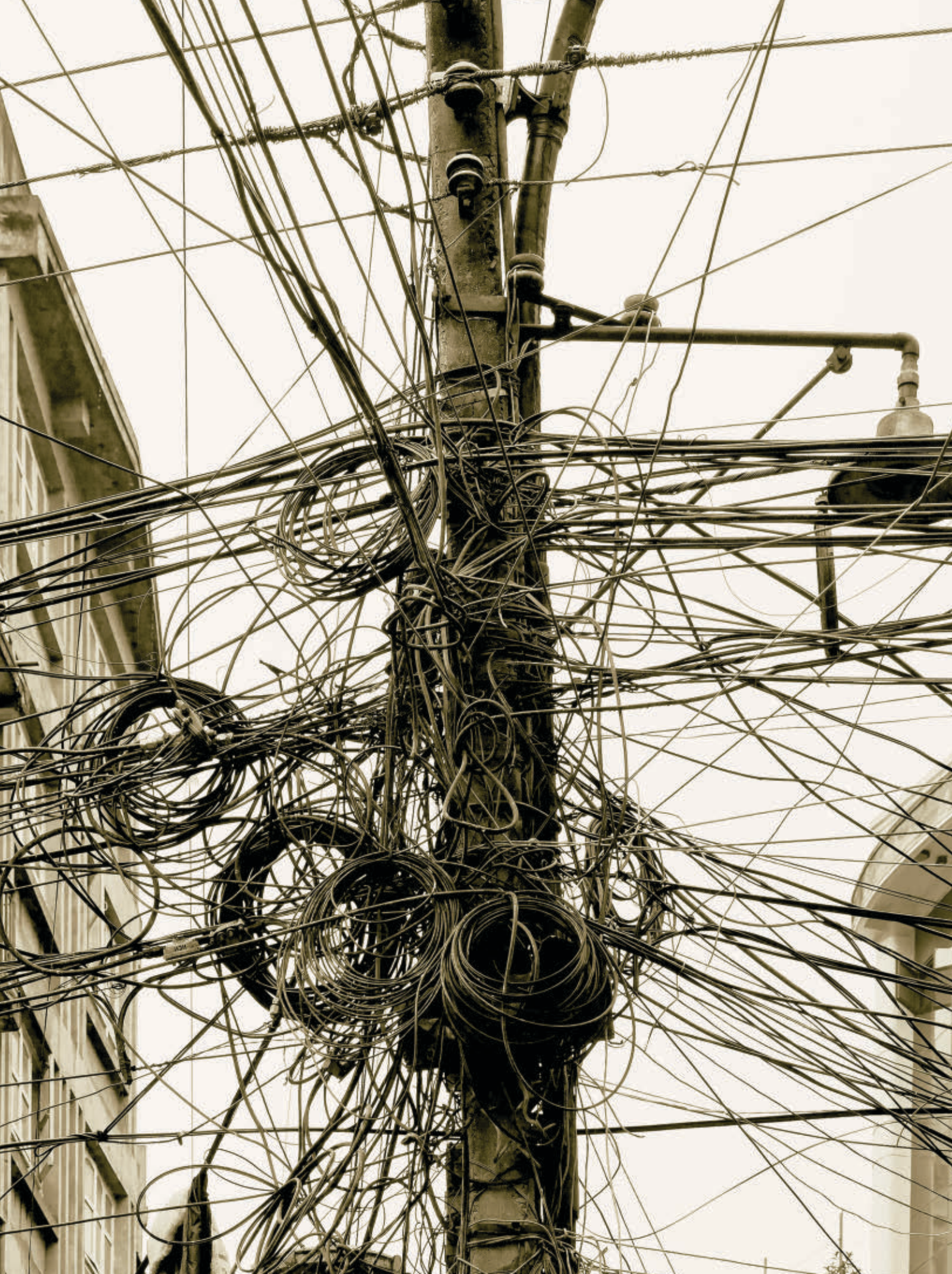
End-to-End Testing

A common pitfall in testing is to check input and output from only a single system. A good chunk of the time this will work to check basic system functionality, but if that data is going to be propagated to multiple parts of a distributed system, it is easy to overlook issues that could cause a lot of problems later.

Consider a system with three parts (illustrated in Figure 1): one that collects data, one that collates all of the data into consistent messages, and one that receives the data and stores it for later retrieval. Test suites can be easily written for each of these components, but a lot can still go wrong from one end to the other. These errors can be tricky—issues such as order of data arrival and data arrival timing can cause all sorts of bugs, and there are so many possible intersections of these things throughout the system that it is difficult to predict and test all of them.

A good way of addressing this problem is to make all the components of the system configurable so they can be run locally. If all of the components of a system can run on the same box, local end-to-end tests are effective. They provide control over when instances of the system elements are brought up and shut down, which is difficult when the elements are spread across different machines. If it is not possible to co-locate processes on the same box, you can run them in virtual machines, which provides much of the same functionality as running them on the same machine. An effective way of testing is to write a series of individual component tests that verify each component is working properly, then write a series of tests that verify data delivery is working properly from one end of the system to the other.

IMAGE BY VADIM PETRAKOV



The example given in Figure 1 would not be difficult to test—after all, there are only three working parts. In the real world, however, highly distributed systems can have hundreds of individual components to run. In addition, most real-world distributed systems are not simply chains of systems that cleanly execute from one end to the other. To demonstrate this point, let's expand the previous example to include hundreds of data collectors throughout the world, each pumping data into the collator, which then pushes all of the data into a distributed data storage system with a configurable number of storage nodes (as shown in Figure 2).

It is nearly impossible to test all the configurations. Typically, the best that can be done is to come up with a good approximation of how this system will work, using a variable number of data collectors and a variable number of storage nodes.

You may also encounter scenarios in which you have access to only a single component of a distributed system. For example, you could be writing a third-party system that collates data from a variety of different sources to assemble data for a user. More than likely, you do not actually have edit access to the other systems and cannot easily control the output from them. In a situation like this, it is useful to write simulators to mimic various types of output from the other systems, both good and bad. This approach has a few drawbacks:

- It is impossible to know all of the different types of good and bad output the other system may provide.
- Writing various simulators can be time consuming.
- The simulators will not react to receiving data the same way the actual system will.

Simulators can be valuable tools to help develop and debug system components, but be aware of their limitations and do not rely on them too heavily.

Distributed Data Systems

Testing a distributed data store is complicated on its own, even without having to test it on the end of another distributed system. These systems require detailed tests of their own to ensure everything works properly. Sticking them at the end of a distributed-system test can add huge amounts of complexity and time to any test suite. Writing a separate set of tests exclusively for the data store is a good idea; otherwise, the amount of time required to get the data from one end to the other can be frustrating—waiting 30 minutes for data to trickle all the way through a system, only for it to fail a single test, requiring the whole thing to be run again.

While it is still important to test the entire system from beginning to end, a complicated subsystem such as a distributed data store demands to be tested on its own in addition to the full system tests. Distributed data systems have many issues that are very difficult to deal with. Among the most difficult to deal with are asynchronous data delivery and node failure.

Note the eventual consistency requirements for your system may be different than the system described here. In my examples, I am making the following assumptions:

- Data is stored on a certain number of nodes defined when the system is built and the node a certain piece of data is on will not change.
- Data will be sent to a single node, then propagated to the other nodes that need to store copies of the data.
- The system will always return the most up-to-date version of the data, regardless of which node it is stored on.
- If one or more nodes are down, the data will correctly flow to the nodes that were offline once they are brought back online.

Asynchronous Data Delivery

In the custom distributed data storage system that I work on, data is delivered to any one node in the system, which then sends that data asynchronously to all other nodes in the system that are responsible for storing the data. The system that sends the data to the first node does not receive any indication the data has been successfully replicated throughout the system, only the first node received the data properly. The system deals with large enough volumes of data that waiting for synchronous data delivery throughout the entire cluster would be unacceptably slow. This system has worked well for many years, but testing it often takes at least as much time as new development efforts, if not more.

When data is added to the system, testing must verify the data reaches not only the intended node, but also all nodes required to store the data to make the system achieve eventual consistency. Therefore, the test suite needs to pull the data from each node where the data supposedly lives, and not just use a generalized query aimed at the cluster. Without the knowledge of when (or if) the data was propagated properly throughout the rest of the system, however, there is a chance the data has not yet reached its intended destination. It would be possible to add a flag to the executable that would wait until all data has propagated throughout the cluster before returning a valid status to the system adding the data to the cluster. This would guarantee the data has propagated before it is pulled, but it would also alter the manner in which the system works on a

Figure 1. Simple distributed system.

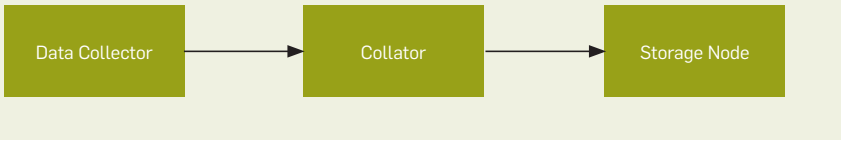
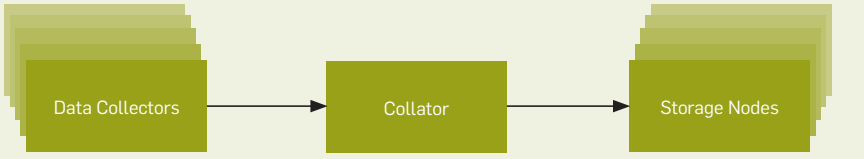


Figure 2. More complex distributed system.



day-to-day basis, and the test would no longer truly be testing normal system functionality.

A solution that maintains normal system functionality is to add artificial delays to the test suite after each successful set of inputs to the primary node. This gives the data time to propagate and increases the chances the data will be on all of the proper nodes by the time an attempt is made to pull it. Obviously, this is not an ideal solution. First, since there are absolutely no timing guarantees, data delivery cannot be guaranteed in any sort of time frame. The timing would also be different based on the speed of the system on which the test suite runs; what works well on one system could be inadequate on another system.

Differences in loads and speeds between the development environment and official build box can also cause problems. I have run a test suite successfully many times on my own development box, only to have the automated build fail the test suite as a result of speed and load differences. This can lead to frustration in trying to determine if there is a legitimate issue with the system or a problem with the test suite.

The obvious solution to this is to increase the number of artificial delays until the system works properly no matter where it runs. These delays, in aggregate, can cause the time the test suite takes to run to increase exponentially. The test suite I use is tied to an automated Jenkins build that runs every time code is committed to our repository. The system is sufficiently complex to necessitate a long, detailed test suite to hit all of the use cases. When enough artificial delays get added to the test suite, the time it takes to run can begin to bloat quickly. A test suite that should take about five minutes to run could end up taking 30 minutes. As more tests are added to the system, it could bloat even more, until running the test suite takes so frustratingly long that you start to ignore it or skimp on adding later tests, leading to bugs slipping through the cracks.

The best way to counteract the impact of this bloating is to run as many tests in parallel as possible. If everything runs in sequence, the time required to run the tests is going to increase rapidly. While the speed of data input likely will not increase if the system is already extremely loaded down, reads can easily be done in parallel.

Testing a distributed data store is complicated on its own, even without having to test it on the end of another distributed system. These systems require detailed tests of their own to ensure everything works properly.

This has an obvious downside—doing your tests asynchronously is much more difficult than simply doing them in sequence. There are far more opportunities for bugs to show up in your test suite itself. You will need to weigh the difficulty of asynchronous testing against the time issues that can crop up if you test everything sequentially. A "hybrid" solution may work well for you—group your tests into small sets, then run the sets sequentially, doing as much of each set in parallel as possible. This allows for minimizing the amount of asynchronous programming you need to do for your test suite while still keeping it reasonably fast.

It is also important to tailor your tests for the requirements of your system. In my example, I am making the assumption there is no specific timing requirement for your system and you can simply extend the amount of time needed to run the tests. However, if you do have specific timing requirements, you cannot simply increase the time required until the tests pass. You will also be dealing with increased loads depending on the number and type of tests being run in parallel. It may become necessary to measure the load and ensure you are reaching your timing requirements based on the amount of data being processed. These are important things to keep in mind when designing your tests, determining the timing in your tests, and determining the amount of load to place on your system when running the test suite.

In the end, there is no ideal way of testing asynchronous data delivery, as each approach has flaws. It really comes down to what works best for a particular system. I find system delays after each successful input to a primary node, combined with high levels of parallel test running, to be the best solution because it most accurately mimics a system actually running in production, without introducing synchronous messages that do not match what the system is doing in production. This is not a one-size-fits-all situation, however. Every distributed data system

is different, and it may make perfect sense for you to add synchronization in testing. Just confirm you are making choices based on what approach best tests your system, not what is easiest for you. When you start writing tests based on ease rather than making sure everything is tested, you are going to start letting bugs through.

Node Failure in a Distributed Data Store

Component failure in a distributed system is difficult to test for. When a component fails, it cannot cause the entire system to fail. The other parts of the system must continue to work properly, even without one or more components. In a distributed data store, ideal behavior is for the node failure to be completely invisible to users. The following situations must be tested to ensure the storage cluster behaves properly:

- Data sent to the system while the node is down still needs to propagate properly throughout the system.
- Data should still be retrievable even with a node failing.
- When the node comes back online, data that must be stored on that node should be propagated to the node and be retrievable from it.
- If the node failure is permanent, the node must be recoverable using data stored throughout the rest of the cluster.

These tests can be via an automated test suite or by setting up a test environment and manually running tests against it to ensure everything is working properly. Once the system is set up and data is flowing into it, you can take a node offline and verify all the data appears to be working. Data can be pulled manually from the data store to ensure it is still retrievable. Once this is verified, the downed node can be brought back online. The data that belongs on this node should begin to flow into the node. After a while, data can be pulled manually from this node to ensure the data that was sent to the cluster when the node was down is stored correctly.

While testing manually is certainly easier, an automated test suite is preferable. As previously discussed, if you can configure your distributed data store to run on one box, you should be able to write an automated test suite to help test it.

The following items need to be guaranteed when testing for node failure in a test suite:

- The test data must have been added to the system and fully propagated throughout the cluster. Several strategies for ensuring this were described earlier.
- When a node is brought down in the test suite, you need to be able to verify it has actually gone down and is inaccessible. A node that did not respond to your signal should not be able to serve data and obscure your results.
- You should be able to pull every single piece of data stored in your distributed data storage system without error. If any pulls fail, then the system has a data-distribution bug that needs to be corrected.

Testing for node failure requires the ability to pull data from individual nodes in the system. It is well worth the time to ensure when attempting to pull data from a node, you can tell the node to do one of the following:

- Provide the "best data" stored in the cluster, along with an indication of where the data came from.
- Provide the data owned by that particular node, regardless of whether or not "better" data is stored elsewhere.

The ability to verify data storage on individual nodes is essential, even if the standard use case is to provide the freshest data. This will help squash lots of bugs before they manifest in the production environment. This may require adding data entry points in the API that are primary (or indeed, exclusively) used for testing, since during typical use, you will likely not care which node the data is served from as long as the data is correct.

Once you have the ability to verify data storage on each individual node in place, it is time to begin testing. The first step is to decide on a node to kill and bring it down. Try to pull data from the node after it fails; the retrieval should fail and exit cleanly. If the retrieval gives back data or hangs, then you need to ensure the data node handles shutdowns properly.

The next step is to pull data from the cluster. You should be able to pull every single data point you added to the cluster, even though a node is down. If any data is missing, you have a data-distribution bug, or the data was not given adequate time to complete all of its asynchronous data distribution before the node was shut down. If the test results are inconsistent from run to run, it is likely the result of asynchronous data-distribution failure. Fix that issue and try again. If the results are consistent, it is much more likely that you have a bug.

If data is stored redundantly across the cluster, then reconstructing a node that has completely failed should be possible. Testing this is similar to testing a node that is down: turn off the node, remove all of the data, rebuild the node, turn the node back on, and verify the data you expect to be on that node actually exists on it. You really want to have this functionality, and you definitely must ensure it works properly. When a production node crashes and loses all of its data, you need to be able to re-create the node quickly and with confidence it is working.

Conclusion

Verifying your entire system works together and can recover cleanly from failure conditions is extremely important. The strategies outlined in this article should help in testing systems more effectively.

It is important to note, however, that no two distributed systems are the same, and what works in some may not work in others. Keep in mind the way your own system works and ensure you are testing it in a manner that makes sense, given the particulars of your system. C

Related articles on queue.acm.org

The Antifragile Organization
Ariel Tseitlin
http://queue.acm.org/detail.cfm?id=2499552

Distributed Development Lessons Learned
Michael Turnlund
http://queue.acm.org/detail.cfm?id=966801

Lessons from the Floor
Daniel Rogers
http://queue.acm.org/detail.cfm?id=1113334

Philip Maddox is a systems engineer at Circonus, a leading provider of monitoring and analytics for IT Operations and DevOps, where he works on a large distributed data storage system. Previously, he wrote code for mail-sorting machines for the U.S. Postal Service.

VRST 2015

The 21st ACM Symposium on Virtual Reality Software and Technology

http://vrlab.buaa.edu.cn/vrst2015/

The 21st ACM Symposium on Virtual Reality Software and Technology (VRST) is an international forum for the exchange of experience and knowledge among researchers and developers concerned with virtual reality software and technology. VRST will provide an opportunity for VR researchers to interact, share new results, show live demonstrations of their work, and discuss emerging directions for the field. The event is sponsored by ACM SIGCHI and SIGGRAPH.

VRST 2015 will be held in Beijing, the capital of China. From the magnificent Palace Museum, also known as the Forbidden City, to the beautiful Summer Palace and the Great Wall, Beijing is the political, economic and cultural center of China for over 800 years from the Yuan Dynasty. The numerous royal buildings with long history endow it with incomparable charm. On the other hand, as the host city of the 2008 Olympic Games, this oriental ancient city presented her best fashion fascination to the world. The conference will be hosted by China State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering in Beihang University (BUAA). VRST 2015 aims at bringing together VR researchers from around the world to present the state-of-the-art advances in this ever-growing dynamic area, and introducing VR research in China.

Important dates.

All deadlines are 15:59 UTC/GMT (Beijing time 23:59):

- **July 20th, 2015:** Abstract submission
- **July 27th, 2015:** Full/short papers submission
- **August 15th, 2015 :** Poster submission
- **September 8th, 2015:** Decisions announced
- **September 15th, 2015:** Camera-ready papers due
- **November 13th–November 15th, 2015:** Conference

Conference Chairs:
Qinping Zhao, Beihang Univerisity
Daniel Thalmann, Nanyang Technological University

Program Chairs:
Enhua Wu, University of Macau & Institute of Software, Chinese Academy of Sciences
Ming C. Lin, University of North Carolina at Chapel Hill
Lili Wang, Beihang University

Local Chair:
Dangxiao Wang, Beihang University

DOI:10.1145/2699412

This defense-in-depth approach uses static analysis and runtime mechanisms to detect and silence hardware backdoors.

BY SIMHA SETHUMADHAVAN, ADAM WAKSMAN, MATTHEW SUOZZO, YIPENG HUANG, AND JULIANNA EUM

Trustworthy Hardware from Untrusted Components

HARDWARE IS THE root of trust in computing systems, because all software runs on it. But is the hardware trustworthy? How can we ensure it has not been corrupted? Can we design it so it is not easily corrupted? Many factors conspire to make hardware more susceptible to malicious alterations and less trustworthy than in the past, including increased use of third-party intellectual property components in system-on-chip designs, global scope of the chip-design process, increased design complexity and integration, and design teams with relatively few designers responsible for each subcomponent. There are unconfirmed reports of compromised hardware[17,21] leading to undesirable economic consequences.[4] A nontechnical solution is to design and manufacture hardware locally in a trusted facility with trusted personnel. However, it is not long term or viable, as it is neither efficient nor guaranteed to be secure. This is why this article instead reviews a series of measures for increasing the trustworthiness of hardware.

To understand how hardware can be compromised, we need to understand how hardware is designed (see Figure 1). The first few steps are similar to software design and construction, beginning with the specification of design requirements. The hardware is then designed to meet operational requirements and coded into a hardware design language (HDL) (such as Verilog) either by designers working with the company designing the chip or with code purchased as intellectual property (such as for a USB controller) from third-party vendors around the world. The next step differs slightly from software. Hardware undergoes much more rigorous validation than most software, as hardware bugs, unlike their software counterparts, are often more expensive to fix following deployment. To minimize the risk of bugs, reputable hardware companies often employ validation teams that are much larger than the design team. They work either in tandem with designers or after the fact in the case of third-party IP components. The design, with all its components, is then processed using computer-aided design (CAD) tools from com-

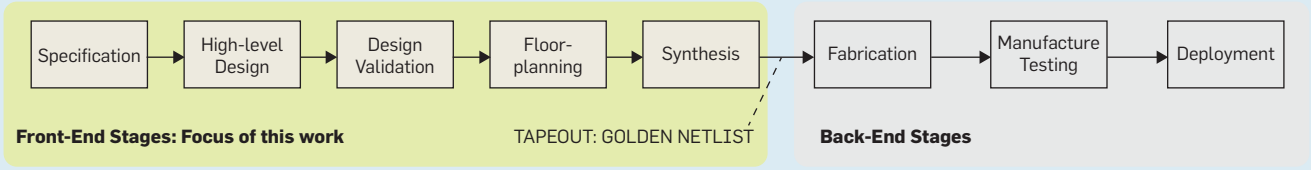
» key insights

- **The approaches described here address a fundamental concern in hardware supply-chain security: Compromised hardware means security cannot be guaranteed.**
- **Mitigations address digital hardware backdoors by including pre-design and runtime techniques.**
- **Developers of mitigation techniques must still address several open problems, including formal verification, unified approaches to design, and foundry security.**

IMAGE BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES


Aperiendum
Expellimus
Fideles
Amen
Illegibilis
Sinistro
Gemino
Fumi Spectatis Revelio
Volumine
Pone Me
Pessundimur
Cuiusdam
Cuiusdam
Cuiusdam

Figure 1. Front end and back end of the modern supply chain for digital hardware design.

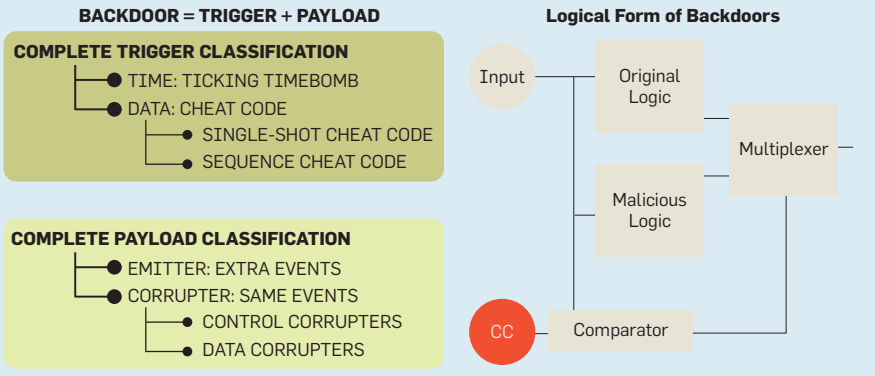


mercial companies that convert the high-level code into gates and wires. When done, the result is a functional design that can be reviewed for security but in practice is simply sent off to a foundry for manufacture. Reviews are encumbered by the complexity of the design and pressure of time-to-market constraints. We refer to everything until compilation with CAD tools as the front end of the process and the physical design and manufacturing at the foundry as the back end of manufacturing.

Thousands of engineers may have access to hardware during its creation and are often spread across organizations and continents. Each hardware production step must be considered a possible point of attack. Designers (either insiders or third-party providers) might be malicious. Validation engineers might seek to undermine the process. CAD tools that could be applied for design synthesis prior to manufacture might be malicious. Moreover, a malicious foundry could compromise security during the back-end manufacturing process. The root of trust is the front-end design phase; without a "golden design" to send off to the foundry, hardware designers, as well as end users, have no basis on which to secure foundries.

We have worked on methods to produce golden designs from untrusted components since 2008, with the goal to secure the front end of the hardware design process, forming a trustworthy root on which to build further security. This article explores the advances we have made toward mitigating the threat and describes remaining problems. The philosophy behind our solution is hardware attacks fundamentally need hardware-based defenses. It is not possible for software alone to create security solutions that cannot be undermined by the underlying hardware. Since these defenses are to be implemented in hardware, they have to obey typical hardware constraints (such as low power, low area footprint, low design and verification complexity, and low performance impact). We show how digital designs can be hardened against an array of known and possible attacks, including but not limited to untrusted insiders and a global conspiracy of attackers, through a defense-in-depth approach. The result of applying our methods is hardware design that can be trusted, despite the resources, people, and components used to produce it being untrusted.

Figure 2. Taxonomy and illustration of digital backdoors.[23]



Adversarial Model

Organizations that aim to earn a profit or even just stay in business are unlikely to sabotage their own designs or sell damaged products intentionally. More likely is one or more "bad apples" in an organization (or external forces) attempting to subvert the design. We thus consider four major sources of insecurity concerning the design process: any third-party vendor from which IP is acquired; any local or insider designers participating in the design process; the validation and verification team; and malicious CAD tools or libraries.

All four are relevant, and we consider the possibility that one or all of them could be malicious. The most obvious threats are from third-party vendors, because modern designs can contain tens or hundreds of distinct IP components, many of which may be sourced from small groups of designers. These third-party components may meet functional specifications but do more than they are supposed to. Another significant threat is from rogue insider designers (such as disgruntled employees and implants from a spy agency). In addition, we also consider the possibility of a larger conspiracy of malicious designers, including among members of the validation and verification team within an organization and third-party IP vendors from outside the organization.

As in many security situations, we can view hardware security as a game between the attacker and defender in which the person who plays the last hand usually wins. As a result, we trust a small number of designers (likely only one) to be trustworthy and able to use or implement our defensive tech-

niques correctly and assume untrusted personnel are unable to modify the design after the security techniques are applied to the design. To keep the defense effort small, we aim to produce simple and low-overhead designs.

Backdoor models. We focus our efforts on digital backdoors, or backdoors targeting digital designs and using well-defined digital signals to turn on and perform malicious actions.[a] The signal that activates a backdoor is called the "trigger," and the resulting action is the "payload." Digitally triggered backdoors, like digital implementations, give a measure of certainty to the attacker and are preferable from an attacker's perspective to analog backdoors. To prevent backdoors from turning on during design validation, they must be triggered with very obscure triggers. On the other hand, a triggerless backdoor that is "always on" would be too obvious to ignore, even if one member of the validation team is malicious; others on the team would notice explicitly bad outputs. Using obscure triggers also ensures the design can be compromised uniquely by the design saboteur. Finally, to be stealthy, backdoors must be small, well-hidden, and appear ordinary compared to rest of the design.

Digital backdoors are simply digital finite state machines and hence can change state in only two ways: over time or with input data. There are thus three types of digital triggers: timing triggers, or "ticking time bombs," that induce bad actions after a certain amount of time; data triggers, or "single-shot cheat codes," supplied as input data; and hybrid data/time triggers, or "sequence cheat codes," delivered as a series of data inputs over multiple clock cycles. Time triggers are convenient because they can be implemented without external inputs; for instance, to trigger a backdoor following approximately 20 minutes of wall-clock operation for a device that operates at 1GHz, the attacker can simply implement a 40-bit down counter. These triggers could be valuable for air-gapped networks. Cheat codes, on the other hand, require access to supply inputs but provide an opportunity for the attacker to exclusively sabotage and own the device using a strong trigger (such as a 128-bit data value).

Digital payloads can be of two types: "emitters" or "corrupters." Emitter payloads perform extra actions or send out extra information, beyond what is specified. Corrupters change existing messages to alter data that plays a specific role. Emitter payloads are easier for an attacker to implement, as corrupter payloads are likely to cause other aspects of the system to fail in unexpected ways without careful engineering.

Finally, as in Figure 2, a digital backdoor takes a logical form of a good circuit and a malicious circuit in the design. The outputs of both feed into a circuit that is semantically equivalent to a multiplexer that selects the output of the malicious circuit when a triggering circuit is activated (see the sidebar "Three Promising but Infeasible Approaches").

Defense In Depth

Given the nature of threats against hardware design, we consider "defense in depth" the appropriate and necessary course of action for practical, real-world security. The philosophy is simple: Assume any security method can be defeated but only at great cost and with great difficulty. Then use multiple, independent security methods to make that cost and difficulty as daunting as possible. As an example, if two independent systems detect an attack with 90% probability each, then there is a 99% chance at least one will successfully detect the attack.

Here, we present three distinct and independent security systems for protecting hardware designs. They act "in series", with later methods coming into play only if a previous system has failed. At a high level, our secure hardware flow works like this: Our first method (which is static) checks that the design being used is backdoor free. If it is somehow subverted, then the second system (which operates at runtime) alters inputs to hardware circuits within a chip to ensure no backdoors can turn on. However, if this, too, fails, the final system (also a runtime system) uses on-chip monitoring to detect that a backdoor has turned on, at which point the backdoor can either be disabled or in the worst case

Three Promising but Infeasible Approaches

Three approaches can, in theory, be used to mitigate the hardware-backdoor problem but are today not possible in practice:

Check outputs. **One defense against hardware backdoors is to procure multiple untrustworthy devices from multiple vendors and check their outputs against each other. As long as there is no conspiracy among the vendors supplying the designs, this solution can work; for instance, an ARM chip can be procured from several vendors, and, while it can detect attacks and perhaps provide resilient operation in the presence of backdoors, it has an overwhelmingly large design and runtime overhead. To detect router backdoors, a data center may need to have twice as many router chips.**

Fully homomorphic encryption (FHE). **FHE allows for computation on encrypted data. Given an FHE version of a hardware circuit, if a designer attempts to insert a backdoor after the circuit is designed and validated but before or during fabrication, the designer is limited to exfiltrating encrypted data or inducing uncontrolled integrity failures. Our prototyping experiments revealed in practice these costs are orders of magnitude more than what would be practical[22] and antithetical to silicon trends, as there is no dark silicon in homomorphic computation.**

Compared to trusted formal specification. **Hardware designers can be assured hardware implementations are backdoor-free if they can be compared to a trusted formal specification. This solution is more promising than the other two. However, the cost of writing formal specifications tends to increase design costs. Formal verification, as used today, can be applied to small systems or parts of a system, but the computational costs scale exponentially with system complexity. Future breakthroughs toward solving these problems could improve the state of the art in trustworthy hardware.**

a We define backdoor as an undisclosed feature that compromises the confidentiality, integrity, and/or availability of the expected design and/or implementation.

the system can be shut down. So even if several of our axioms are violated unexpectedly, the worst attack that can be perpetrated is denial of service.

Static Analysis

The first of the three stages of our defense-in-depth approach to hardware security is static analysis, which occurs during design and code entry, before the hardware is manufactured. The design being analyzed can be developed internally by an organization's own design team or acquired from a third party. As this first line of defense, we have thus developed the first algorithm for performing static analysis to certify designs as backdoor free and built a corresponding tool called Functional Analysis for Nearly unused Circuit Identification, or FANCI.[25]

Key insight. Recall from the earlier section on backdoor models that a backdoor is activated when rare inputs called triggers are processed by the hardware. Since these inputs are rare, the trigger-processing circuit rarely influences the output of the hardware circuit; it switches the output of the circuit from the good subcircuit to the malicious subcircuit only when a trigger is received. If security evaluators can identify subportions of a hardware circuit that rarely influence output, they can narrow down the set of subcircuits that are potentially malicious, stealthy circuits.

Boolean functional analysis helps security evaluators identify subcircuits that rarely influence the outputs. The idea is to quantitatively measure the degree of influence one wire in a circuit has on other wires using a new metric called "control value." The control value of an input wire w_1 on a wire w_2 quantifies how much the truth table representing the computation of w_2 is influenced by the column corresponding to w_1. FANCI detects stealthy subcircuits by finding wires with anomalous, or low, control values compared to other wires in the same design.

FANCI. The algorithm to compute the control value of w_1 on w_2 is presented here as Algorithm 1. The control value is a fraction between zero and one, quantifying what portion of the rows in the truth table for w_2 is directly influenced by w_1. In step 3 of the algorithm, we do not actually construct the exponentially large truth table. Instead, we construct the corresponding Boolean function. Since the size of truth tables grows exponentially, to scale FANCI, the algorithm approximates control values through a constant-size subset of the rows in the truth table.

Algorithm 1. Compute control value.

1: $count \leftarrow 0$
2: $c \leftarrow$ Column(w_1)
3: $T \leftarrow$ TruthTable(w_2)
4: **for all** Rows r in T **do**
5: $\quad x_0 \leftarrow$ Value of w_2 for $c = 0$
6: $\quad x_1 \leftarrow$ Value of w_2 for $c = 1$
7: $\quad$ **if** $x_0 \neq x_1$ **then**
8: $\quad\quad count$++
9: $\quad$ **end if**
10: **end for**
11: $result \leftarrow \frac{count}{size(T)}$

Algorithm 2. How FANCI flags suspicious wires in a design.

```
(Flatten design across modules if necessary)
(Initialize state elements to random values if necessary)
for all modules m do
    for all gates g in m do
        for all output wires w of g do
            T ← TruthTable(FanInTree(w))
            V ← Empty vector of control values
            for all columns c in T do
                Compute control of c
                Add control(c) to vector V
            end for
            Compute heuristics for V
            Denote w as suspicious or not suspicious
        end for
    end for
end for
```

As an example, suppose we have a wire w_2 that is dependent on an input wire w_1. Let w_2 have n other dependencies. From the set of possible values for those n wires (2^n), we choose a constant number of, say, 10,000. Then for these 10,000 cases, we toggle w_1 to zero and then to one. For each of the 10,000 cases, we see if changing w_1 changes the value of w_2. If w_2 changes m times, then the approximate control value of w_1 on w_2 is $m \div 10{,}000$. When we have computed all control values for a given wire (an output of some intermediate circuit), we have a vector of floating-point values we can combine to make a judgment about stealth. We find using simple aggregating metrics (such as the arithmetic mean and median) is effective for identifying stealthy wires. Other metrics may be possible and interesting in the future. The complete algorithm used by FANCI is summarized in Algorithm 2.

Evaluation. To evaluate FANCI effectiveness, we use benchmarks from the TrustHub suite, a popular benchmark suite for work on hardware backdoors.[19] We evaluate multiple stealth metrics built atop the core notion of control values. The most important result is we did not encounter any false negatives. For each benchmark and each of the heuristics, we discovered at least one suspicious wire from each backdoor, or enough for us to manually identify the functionality of the hidden backdoors. We also observed different metrics can highlight different parts of the backdoor. The mean and median tend to highlight backdoor payload wires, while triviality[b] more often highlights the backdoor trigger wires.

Figure 3 includes the results for 18 TrustHub benchmarks we analyzed regarding false positives. In addition to achieving low false-positive rates, we see surprisingly good scalability. As designs get larger, FANCI yields lower false positive rates, because designs tend to get larger due to wider data paths rather than fundamentally more complex logic. When this is the case, FANCI's job gets easier. In addition to TrustHub benchmarks we

b Triviality is an alternate aggregation metric where the score is the fraction of the time the output is fixed (either zero or one).

also ran FANCI on a benign, academic out-of-order microprocessor core.[6] We observed the runtime was approximately linear in the number of wires in the design and all experiments completed within three days with an unoptimized serial implementation.[25]

We also performed a red team/blue team experiment, where several teams from the U.S. and Europe tried to defeat FANCI, but FANCI performed well, catching all stealthy attacks and even a few non-stealthy (frequently- or always-on) attacks.[14] While FANCI is not normally expected to detect frequently-on backdoors, sometimes even frequently-on backdoors use somewhat stealthy or abnormal logic for trigger recognition.

Benefits. FANCI is independent of inputs that might be supplied during validation for testing. Even with high-quality test suites, most tests fail to exercise all internal circuits, as input coverage and code coverage are not equivalent to internal state or value coverage. Such limited coverage provides attackers an obvious way to hide their backdoors. By operating statically, FANCI is able to observe the behavior of every gate in the design. Due to the way FANCI works, the algorithm is guaranteed to flag any stealthy combinational logic that bypasses design validation. FANCI also works against sequential (state-machine-based) backdoors, as state-based backdoors generally require combinational logic to recognize the trigger state, and FANCI is (in our experience) able to catch that combinational logic.

Silencing Triggers

If an attacker evades functional analysis through FANCI, then a backdoor exists within the design and will be fabricated. Our next defensive strategy is to include security circuits within the design itself that check security properties at runtime. We add these circuits after the design is completed and validated, so even a malicious insider on the design or validation teams cannot compromise them.

Key insight. A stealthy hardware backdoor always contains a trigger, the unique signal that turns the module from benign mode to malicious mode and enables the backdoor payload. The idea behind trigger obfuscation is to scramble signals at the input interfaces of hardware modules to prevent backdoor triggers from going off (see Figure 4). Since payloads depend on triggers, if the triggers do not arrive, the malicious payloads cannot be delivered. We discuss in the following sections how the three types of triggers—ticking time bomb, single-shot, and sequence cheat codes—can be prevented from turning on through this approach.

Power resets. Power resets protect untrusted units against ticking time bombs. A ticking time bomb, as mentioned earlier, goes off after a certain amount of time, say, 240 clock cycles of operation. An attacker can implement it with a 40-bit down counter. The time bomb will not go off during design validation because validation is carried out for much shorter time scales due to the slowness of validation testing and time-to-market constraints. Validation suites typically run in the KHz range compared to production hardware that runs in the GHz range. As such, validation tests are run for a small number of cycles of, say, 10 million cycles per test. Given these parameters and constraints, we see why a malicious engineer on the design team with insider information on the duration of the validation tests can easily use this information to make the backdoor trigger fire well after the validation period.

Our solution to mute this trigger is to prevent circuits from knowing a cer-

Figure 3. False-positive rates for the four different metrics and for TrustHub benchmarks; the RS232 benchmark, the smallest, has about 8% false positives, and the others have much lower rates (less than 1%).

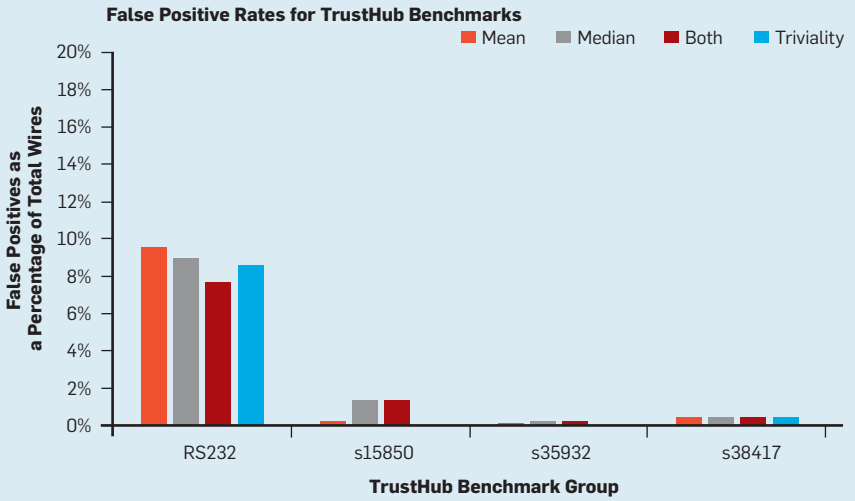
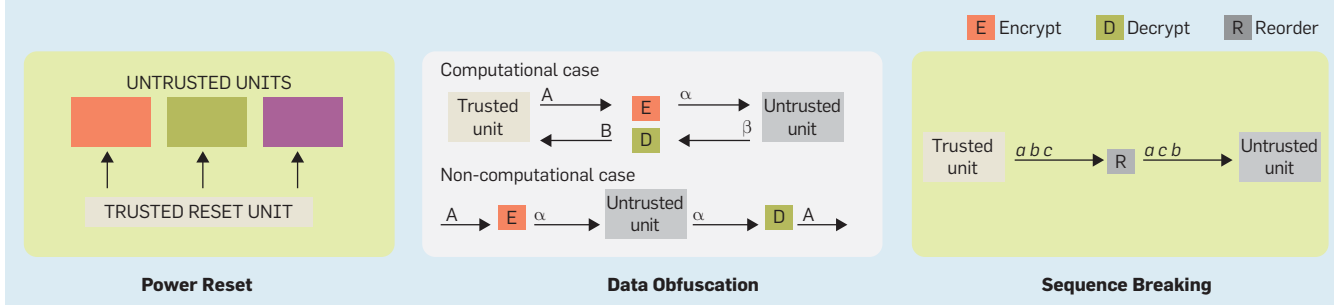


Figure 4. Overview of our three methods for trigger obfuscation.[24]



tain amount of time has passed since start up. We ensure this by frequently powering off and on (or resetting) each unit, causing all microarchitectural data to be lost. The frequency of power resets is determined by the validation epoch. We know and trust the circuit under test to be free of timing backdoors during the validation epoch; otherwise, it would not pass any validation, so resetting power before the end of the validation epoch ensures all state in the unit is wiped clean and the unit no longer has "memory" of previous computations; for instance, if the attacker is using a counter to trigger the backdoor, the counter will be reset to its initial value and never get a chance to hit the trigger.

If the defense technique is to constantly reset power to the unit, how can the computation make forward progress? Many common hardware modules behave "transactionally," meaning the module applies a function only to its inputs and does not need to maintain long-term state. However, some hardware requires continuity of state across reset epochs. To ensure forward progress, we identify and store all architectural state necessary for forward progress outside the unit before the reset. From an implementation perspective the reset idea can be implemented through "power gating" circuits, a technique to limit power consumption where units are reset during periods of inactivity.

With this approach, how can a designer be assured the attacker does not exfiltrate the time counter (or some proxy that encodes the passage of time) through the state necessary for forward progress? This deliberate exfiltration of time is not possible due to our axiom that the unit does not produce incorrect results for the duration of the validation period. If time information is illegally exfiltrated during validation, just one good validation engineer from the entire design team is sufficient to notice and detect this violation. A final move by the attacker could be to store the trigger in nonvolatile state. This is a clever move but somewhat impractical, as the design process for creating nonvolatile hardware differs so much from regular hardware that this move would set off many different alarms at various stages in the design process. Storing the trigger in nonvolatile state may not pose a significant risk to most hardware units, except those that already require nonvolatile memory by design (such as trusted platform modules). Solutions to the problem of storing the trigger in nonvolatile state are covered by Waksman and Sethumadhavan.[24]

Each hardware production step must be considered a possible point of attack.

Single-shot cheat code. The second category of backdoor trigger is the single-shot cheat code that is delivered as a data value through one of the hardware input interfaces. Our solution here is to scramble the inputs supplied to a hardware module. Scrambling the inputs silences backdoors because the attacker cannot anticipate the trigger. However, hardware designers must figure out how to perform computation in the unit without unscrambling the input value.

Our solution draws on domain-specific applications of homomorphic encryption or homomorphic obfuscation, since there is no need for cryptographically strong encryption in the hardware backdoor-detection setting. The following paragraphs are a brief introduction to homomorphic functions, explaining our solution with an example.

An operation f is considered homomorphic with respect to another operation g if $f(g(x), g(y)) = g(f(x, y))$. One example of homomorphic operation is when f is multiplication and g is the squaring function, as in

$$x^2y^2 = (xy)^2$$

If the functionality required of a hardware module is to compute the square of a value, the security engineer can obfuscate the input x to that unit by multiplying it by a pseudo-random value y. The squaring unit then computes the value $(xy)^2$. Then, to decrypt, the security engineer has to only divide by the constant y^2 to get back x^2. Since the engineer permuted the input space in a homomorphic (structure-preserving) way, the engineer did not undermine the usefulness of the squaring module. More generally, if the obfuscation function is homomorphic over the computational function, then the computation can be done on the data while it is encrypted, allowing permutation of the inputs to silence the triggers and at the same time per-

form useful computations. This is one specific application of homomorphic functions, which have been used more broadly in software settings and in theory literature.[7] If the engineer can create low-overhead homomorphic equivalents for typical functional units, protection can be improved against hardware backdoors.

Also worth noting is that for many hardware units, the work performed in the hardware module is an identity operation (moving data or storing without changing it, as in routers and caches), in which case any operation can be used for obfuscation. The engineer can then use a bitwise XOR operation with a changing random number (a one-time pad). This technique adds only one gate to the path delay.

Sequence cheat code. The third and final type of backdoor trigger is the sequence cheat code, which we protect against through a method called "sequence breaking." Here, the trigger is received as several pieces of data over multiple cycles. Our solution is to benignly reorder sequences of inputs. If the attacker cannot predetermine the order transactions will be supplied to the hardware module, the attacker cannot trigger a backdoor with any significant probability. The main constraint here is ensuring changing the order of transactions does not change the functionality of the hardware unit.

As a realistic example of sequence breaking, consider a memory controller with a backdoor triggered by a sequence of 50 loads and stores that must come in a pre-chosen order. By changing the order of the 50 loads, we make it unlikely the correct trigger will ever be supplied. The intuition for why it is easy to make the probability of attack failure effectively 100% is the trigger ends up being one pseudo-random choice out of the set of all possible permutations of memory instructions, which is a factorially large set.

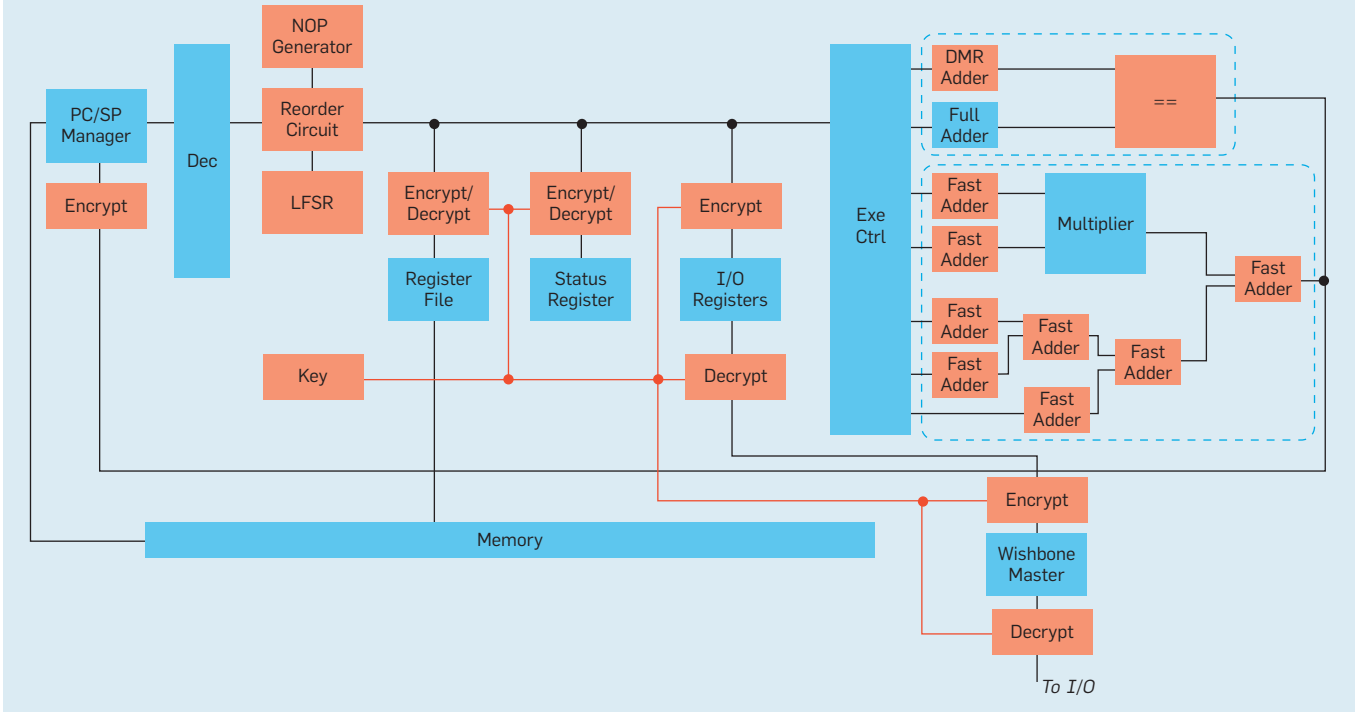
It may not be possible to reorder input transactions to a unit in all cases; for example, a malicious user may be able to contrive a sequence of inputs to a memory controller that is not amenable to reordering without violating correctness requirements (such as consistency or coherence requirements). For such cases, we propose adding random dummy events; for example, inserting random dummy events (such as nonbinding loads to the memory controller) is sufficient to break any predetermined order for a trigger.

Evaluation. To evaluate the overheads of our techniques to silence triggers we prototyped a microcontroller[22] that included all our security features. We found by applying solutions to each component in the design, we were able to protect each module; the only module our silencing triggers approach could not protect was the adder. To protect the adder we used two-way redundancy. The silicon-area overhead of the proposed solution was less than 8%. Figure 5 outlines the architecture of the microcontroller with our silencing modifications.

Security guarantees. The security of cheat-code-based techniques is probabilistic but effectively offers near-perfect coverage; the probability of a failure in each case scales as roughly $1 - 1/2^n$ where n is the width of the trigger signal. In practice, n is large; most units that cannot be exhaustively validated to check for backdoors have 64-bit or larger interfaces. So the probability is negligible in practice. One way the proposed solutions can fail is if the random number generator used for the scrambling remaps a trigger to itself, which is extremely unlikely.

The strength of many of our techniques depends on having a trusted

Figure 5. Microarchitectural overview of the TμC1 microcontroller.[22]



source for generating random numbers. However, our approach does need true cryptographically strong random-number generators and can work with weakened pseudo-random number generators.[5] The ability to work with possibly weak pseudo-random number generators is due mainly to how the silencing-triggers approach uses the random numbers generated in hardware. One way for an attacker to defeat the scrambling schemes used in the silencing-triggers approach is to learn about the constants for pseudo-random number generation by choosing plaintext inputs; for instance, forcing the input to zero can reveal a sequence of outputs produced by the random number generator when the scrambling function is a simple XOR. To generate the trigger despite scrambling, the malicious hardware unit needs to reverse engineer the random-number generator *within* the malicious hardware unit. This requirement imposes severe constraints on the attacker and can severely reduce their stealth. We built this attack, showing the reverse-engineering circuit for pseudo-random number generators can be quite large (22x silicon-area overhead). Such overheads alone can make such attacks very obvious to hardware designers and validators alike.

Figure 6. Overview of the TRUSTNET and DATAWATCH monitoring schemes.[23]

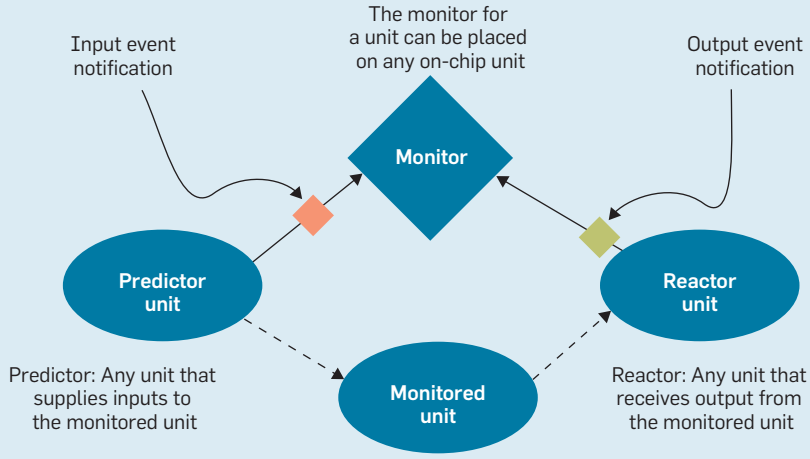
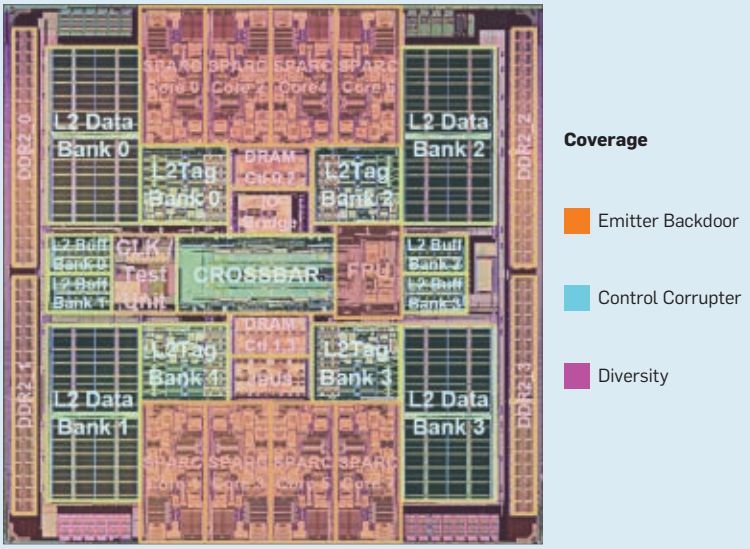


Figure 7. Units covered by TRUSTNET and DATAWATCH on an OpenSPARC microprocessor (approximate illustration).[23]



Payload Detection

If a backdoor is somehow designed to bypass both FANCI and silencing triggers, and manages to turn itself on at a maliciously intended time, then we need to be able to detect the attack payload and either recover or fail gracefully. To make this happen, we use a dynamic on-chip monitoring system that performs backdoor payload detection by continuously checking invariants.

Key insight. Our technique for malicious payload detection is based on two key observations: First is that in hardware systems events between hardware units are largely deterministic, and there exist invariants that are violated only when backdoors achieve their payloads. Examples of invariants between hardware units may be simple checks (such as the number of instructions executed by the functional units should not be larger than the number instructions processed by the instruction decoder). The second is that if we, as security engineers, make the weak assumption that the majority of modules are not malicious, we can build a secure invariant-monitoring infrastructure even if we do not know which modules are malicious. For our initial defensive implementations, we assume exactly one module (or one design team) is malicious, but there are natural extensions to n malicious modules for $n > 1$.[23]

The first system we built is called TRUSTNET. The basic idea (see Figure 6) is to have self-monitoring triangles formed out of any set of three hardware modules. In a typical hardware design, any module is connected to several others; for example, in a microprocessor pipeline, a decoder is connected to a fetch and an execution unit, among others. There is usually some unit that sees data before it (in this case the fetch unit) and some unit that sees data after it (in this case the execution unit). We call them the "predictor" and the "reactor," respectively. By adding a verification step between the predictor and the reactor, we can confirm the untrusted (monitored) unit is doing reasonable work; for example, the fetch unit and the execution unit can communicate to ensure the correct number of instructions is being decoded. This simple invariant (*instructions in* =

instructions out) prevents all decoder-based code-injection attacks.

We also designed an extension to TrustNet called DataWatch. Since TrustNet measures simple invariants that can be checked by counting (such as numbers of instructions or events), DataWatch performs simple data-integrity checks; for example, it can check that memory events have the correct types or the correct opcode types are being supplied to pipeline units. DataWatch is inherently more heavyweight than TrustNet, since TrustNet uses one-bit signals, and DataWatch uses multi-bit signals. However, DataWatch is still efficient for modern hardware designs.

There are several techniques for handling alarms from TrustNet and DataWatch. One response could be to shut down the entire system, turning a confidentiality or integrity attack into an availability attack. In highly secure domains, this response may be desirable to guarantee no exfiltration of sensitive data. If continuous operation is desired, then some form of *N*-way redundancy is an option. If designs from multiple vendors are available, when an alarm sounds, computation can be migrated onto another machine from a different vendor and resumed from a last known good state; for instance, if the alarm sounded on an Intel x86 chip, the computation can be moved to an AMD x86 chip and resumed. The proposed technique can also be used for forensic examination. When an alarm goes off, the data in flight in the hardware units could be flagged as a cheat code and logged for future execution.

Evaluation. We demonstrated our payload-detection approach on portions of the Sun Microsystems OpenSPARC T2 microarchitecture. We used the HDL hardware implementation to systematically determine the number of on-chip units that can be covered by TrustNet and DataWatch. To measure this attack/vulnerability surface we observe a hardware unit is vulnerable to backdoors only insofar as its interfaces are threatened. The processing inside the unit can be checked to be correct simply by checking its inputs and outputs. The interfaces thus present points of vulnerability, and the efficacy of our solution is then determined by whether or not these interfaces are protected from attacks through TrustNet and DataWatch. Figure 7 outlines the units on a chip that can be covered partially or fully through the invariant specification technique.[c]

It is not possible for software alone to create security solutions that cannot be undermined by the underlying hardware.

Security guarantees. Consider a monitor that is set up to watch some unit *X*. Four actors are in play: the predictor of *X* (*P*); the reactor to *X* (*R*); *X* itself; and the monitor of *X* we call *M*. Under a lone-wolf-attacker model, the attacker gets to corrupt one (and only one) of these actors. In order to compromise confidentiality or integrity, the attacker would have to choose to corrupt *X* itself. In this case, *P*, *R*, and *M* are all untampered and the attack can be detected. If the attacker tampers with *P* or *R*, then they would disagree, and the attack would be detected. The attacker can choose to attack *M*, but since monitors are only one or a few logic gates (XOR gate), the monitors can be formally checked not to contain backdoors. However, the proposed solution will not work for a conspiracy among designers of multiple units or when a single malicious designer designs multiple units responsible for monitoring each other. Organizational security is necessary to prevent such attacks.

Related Research Chronology

The earliest discussion of the dangers of hardware backdoors was by Molho et al.[13] in 1970. The earliest solution we could find was described in thesis work by M.K. Joseph[11] at the University of California, Los Angeles, in 1988, suggesting *N*-version redundancy. Defenses for hardware-based attacks began in earnest around 2005, following a report by the Defense Science Board warning of potential counterfeit chips.[21] In 2008, King et al.[12] demonstrated modern processors are vulnerable to relatively simple hardware backdoors. Subsequent work in 2010 taxonomized the types and properties of possible backdoors, focusing primarily on digital hardware.[20,23]

The first solution for integrating

c This is a conservative estimate, based on only those interfaces we could easily cover; further effort or insider knowledge would likely lead to better coverage.

untrusted third-party IP was presented at the 28th IEEE Symposium on Security and Privacy, where Huffmire et al.[10] proposed isolation mechanisms and communication whitelists to prevent software IP cores with different levels of security from accessing each other. This research was performed in the context of reconfigurable hardware, or FPGAs. The first work on design-level backdoors in contemporary hardware (ASICs) was described in two papers presented at the 31st IEEE Symposium on Security and Privacy in 2010. One, by Waksman and Sethumadhavan,[23] described a hardware-only solution for detecting payloads and is described in this article. Another, by Hicks et al.,[8] proposed a hybrid hardware-software solution, replacing hardware circuits suspected to contain backdoors with circuits that report exceptions. If and when a program reaches these exception circuits—indicating a program may be trying to exercise malicious hardware—program control is transferred to a software emulator to emulate the offending instruction. However, the software emulator runs on the same hardware as the original program that caused the exception, thus introducing a chance of deadlocks and potentially opening up a new attack surface due to emulation. Some interesting counterattacks were performed in follow-up work by Sturton et al.[18] in 2012. Researchers have also designed hardware trojans through competitions[15] and developed tools to identify regions to insert trojans.[16]

In parallel with the work on front-end backdoors, a significant amount of work has been done on foundry-side security, addressing the problem of how hardware manufacturing can be secured if a trustworthy design model is provided. The first such solution was devised by IBM researchers Agrawal et al.[2] in 2007. The idea was to characterize the expected power draw of the chip using a "golden" model, following abnormal, anomalous power readings to identify malicious chips. This line of work generated a significant amount of follow-on effort to improve the efficacy of the technique (such as extending it to smaller technologies resulting in a significant amount of background noise) and discovering alternate "fingerprints" (such as circuit path delays and thermal outputs).[1,3,9,26]

Security without trust is impossible, and hardware today is trusted without good reason.

Since security is a full-system property, both the front end and back end of the design process must be secured. As such, most of these techniques complement the work covered here.

Open Problems

Hardware-design security is a relatively new area of study, with many important open problems and challenges we organize into four categories:

Better homomorphic functions. Homomorphic functions present an interesting approach for securing hardware designs. By encrypting data while it is being operated on, foundries and physical design tools are prevented from compromising hardware. It can also be used to enhance resilience of front-end design attacks, as described in the section on silencing triggers. At the same time, homomorphic functions tend to be expensive, especially when one tries to design them to be more generally or broadly applicable. Coming up with homomorphic implementations of additional functional units and processor pipelines would be valuable.

Formal models, proofs, verification. There is no formal model for describing hardware backdoors today and thus no rigorous description of security guarantees provided by various security techniques or their applicability and limitations. Developing such models poses significant challenges for hardware designers because it requires description of not only complex hardware but also the abilities of agents (such as validation engineers) involved in the design process. Finding the right, meaningful abstractions to describe security techniques would be extremely valuable. Further, many techniques proposed for hardware security are likely composable; proofs to this effect would likewise be valuable.

Foundry security. Given the design-level defenses presented here, the notion of a golden design becomes an achievable goal. This design, or completely trusted design, is assumed by hardware security researchers a basis for most work in foundry-level security. Foundry-level security is vital, because a malicious foundry can ignore every-

thing in the design and produce a malicious chip. However, foundry-level and design-level security literatures are today largely disparate. A unified approach, including both foundry and design security, to create a cohesive notion of hardware security would be valuable.

Analog, nondeterministic, ephemeral backdoors. As defenders, we constantly think about how we would create new backdoors to bypass our own techniques. The defenses discussed here assume digital hardware defined in the HDL. However, one can imagine attacks against analog circuits. One can also imagine ephemeral attacks that work based on physical parameters (such as environmental temperature). There could also be nondeterministic triggers. Understanding the feasibility of such attacks and defending against them would require new ways of thinking about hardware backdoors. Such attacks would be very valuable in advancing the field of hardware security.

Conclusion

Any system needs a degree of trust. Security without trust is impossible, and hardware today is trusted without good reason. As the cliché goes, security is an arms race to the bottom. Attackers exploit hardware (if they are not already doing so) to break security. Strong solutions are needed to build trustworthy hardware from untrusted components.

Here, we have presented a full view of the modern hardware supply chain, explaining how it is vulnerable to attacks. Taking into account all the agents and processes involved in producing a hardware design, we have proposed a defense-in-depth approach for making hardware trustworthy while minimizing the trust in the agents and organizations that supply components.

Using all our proposed methods in concert allows for well-defined notions of trust and can guarantee protection against large and critical classes of hardware-oriented attacks. For computer security to advance, all layers of security must be built atop a trustworthy foundation, starting with methods for strengthening hardware designs against attacks.

Acknowledgments

This work is supported by Defense Advanced Research Projects Agency grant FA 87501020253 through the Clean Slate Security program and National Science Foundation CCF/SaTC Programs grant 1054844 and a fellowship from the Alfred P. Sloan Foundation. Opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. government or commercial entities. Simha Sethumadhavan has a significant financial interest in Chip Scan LLC, New York. We also wish to thank the reviewers for their valuable feedback. C

References

1. Abdel-Hamid, A.T., Tahar, S., and Aboulhamid, E.M. IP watermarking techniques: Survey and comparison. In *Proceedings of the IEEE International Workshop on System-on-Chip for Real-Time Applications* (Calgary, Alberta, Canada, June 30–July 2). IEEE Press, 2003.
2. Agrawal, D., Baktir, S., Karakoyunlu, D., Rohatgi, P., and Sunar, B. Trojan detection using IC Fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, CA, May 20–23). IEEE Press, 2007, 296–310.
3. Banga, M., Chandrasekar, M., Fang, L., and Hsiao, M.S. Guided test generation for isolation and detection of embedded trojans in ICS. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI* (Pittsburgh, PA, May 20–22). ACM Press, New York, 2008, 363–366.
4. BBC News. U.S. Government restricts China PCs (May 19, 2006); http://news.bbc.co.uk/2/hi/4997288.stm
5. Becker, G.T., Regazzoni, F., Paar, C., and Burleson, W.P. Stealthy dopant-level gardware trojans. In *Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Security*. Springer-Verlag, Berlin, Heidelberg, Germany, 2013, 197–214.
6. Choudhary, N., Wadhavkar, S., Shah, T., Mayukh, H., Gandhi, J., Dwiel, B., Navada, S., Najaf-Abadi, H., and Rotenberg, E. Fabscalar: Composing synthesizable RTL designs of arbitrary cores within a canonical superscalar template. In *Proceedings of the 38th Annual International Symposium on Computer Architecture* (San Jose, CA, June 4–8). ACM Press, New York, 2011, 11–22.
7. Gentry, C. Computing arbitrary functions of encrypted data. *Commun. ACM 53*, 3 (Mar. 2010), 97–105.
8. Hicks, M., King, S.T., Martin, M.M.K., and Smith, J.M. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *Proceedings of the 31st IEEE Symposium on Security and Privacy* (Oakland, CA, May 16–19). IEEE Computer Society Press, 2010, 159–172.
9. Hu, K., Nowroz, A.N., Reda, S., and Koushanfar, F. High-sensitivity hardware trojan detection using multimodal characterization. In *Proceedings of Design Automation & Test in Europe* (Grenoble, France). IEEE Press, 2013, 1271–1276.
10. Huffmire, T., Brotherton, B., Wang, G., Sherwood, T., Kastner, R., Levin, T., Nguyen, T., and Irvine, C. Moats and drawbridges: An isolation primitive for reconfigurable hardware-based systems. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, CA, May 20–23). IEEE Press, 2007, 281–295.
11. Joseph, M.K. *Architectural Issues in Fault-Tolerant, Secure Computing Systems.* Ph.D. Thesis, University of California, Los Angeles, 1988; http://ftp.cs.ucla.edu/tech-report/198_-reports/880047.pdf
12. King, S.T., Tucek, J., Cozzie, A., Grier, C., Jiang, W., and Zhou, Y. Designing and implementing malicious hardware. In *Proceedings of the First Usenix Workshop on Large-Scale Exploits and Emergent Threats* (San Francisco, CA, Apr. 15). USENIX Association, Berkeley, CA, 2008, 5:1–5:8.
13. Molho, L.M. Hardware aspects of secure computing. In *Proceedings of the Spring Joint Computer Conference* (Atlantic City, NJ, May 5–7). ACM Press, New York, 1970, 135–141.
14. NYU Cyber Security Awareness Week. The 2013 Embedded Systems Challenge; https://csaw.engineering.nyu.edu/
15. Rajendran, J., Jyothi, V., and Karri, R. Blue team-red team approach to hardware trust assessment. In *Proceedings of the IEEE 29th International Conference on Computer Design* (Amherst, MA, Oct. 9–12). IEEE, 2011, 285–288.
16. Salmani, H. and Tehranipoor, M. Analyzing circuit vulnerability to hardware trojan insertion at the behavioral level. In *Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems* (New York, Oct. 2–4). IEEE Press, 2013, 190–195.
17. Simonite, T. NSA's own hardware backdoors may still be a problem from hell. *MIT Technology Review* (Oct. 2013); http://www.technologyreview.com/news/519661/nsas-own-hardware-backdoors-may-still-be-a-problem-from-hell/
18. Sturton, C., Hicks, M., Wagner, D., and King, S.T. Defeating UCI: Building stealthy and malicious hardware. In Proceedings of the 2011 IEEE Symposium on Security and Privacy (Oakland, CA, May 22–25). IEEE Computer Society, 2011, 64–77.
19. Tehranipoor, M., Karri, R., Koushanfar, F., and Potkonjak, M. TrustHub; https://www.trust-hub.org/
20. Tehranipoor, M. and Koushanfar, F. A survey of hardware trojan taxonomy and detection. *IEEE Design Test of Computers 27*, 1 (Jan.–Feb. 2010), 10–25.
21. U.S. Department of Defense. *High Performance Microchip Supply* (Feb. 2005); http://www.acq.osd.mil/dsb/reports/ADA435563.pdf
22. Waksman, A., Eum, J., and Sethumadhavan, S. Practical, lightweight secure inclusion of third-party intellectual property. *IEEE Design and Test Magazine 30*, 2 (2013), 8–16.
23. Waksman, A. and Sethumadhavan, S. Tamper-evident microprocessors. In *Proceedings of the 31st IEEE Symposium on Security and Privacy* (Oakland, CA, May 16–19). IEEE Press, 2010, 173–188.
24. Waksman, A. and Sethumadhavan, S. Silencing hardware backdoors. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy* (Oakland, CA, May 22–25). IEEE Press, 2011, 49–63.
25. Waksman, A., Suozzo, M., and Sethumadhavan, S. FANCI: Identification of stealthy malicious logic using Boolean functional analysis. In *Proceedings of the 20th ACM Conference on Computer and Communications Security* (Berlin, Germany, Nov. 4–8). ACM Press, New York, 2013, 697–708.
26. Wei, S., Li, K., Koushanfar, F., and Potkonjak, M. Provably complete hardware trojan detection using test point insertion. In *Proceedings of the International Conference on Computer-Aided Design* (San Jose, CA, Nov. 5–8). ACM Press, New York, 2012, 569–576.

Simha Sethumadhavan (simha@cs.columbia.edu) is an associate professor in the Department of Computer Science at Columbia University, New York, and a recipient of an Alfred P. Sloan fellowship and National Science Foundation CAREER award.

Adam Waksman (waksman@cs.columbia.edu) is a technology developer at The D. E. Shaw Group, New York, and a recipient of a National Defense Science and Engineering Graduate fellowship; his work on this article was performed when he was a Ph.D. student at Columbia University, New York.

Matthew Suozzo (ms4249@columbia.edu) is an engineer at Chip Scan LLC, New York; his work on this article was performed when he was a student at Columbia University, New York.

Yipeng Huang (yipeng@cs.columbia.edu) is a Ph.D. student at Columbia University, New York.

Julianna Eum (julianna.eum@usma.edu) is an instructor in the Department of Electrical Engineering and Computer Science of the U.S. Military Academy at West Point, New York; her work on this article was performed when she was a student at Columbia University, New York.

DOI:10.1145/2667219

Dynamic analysis techniques help programmers find the root cause of bugs in large-scale parallel applications.

BY IGNACIO LAGUNA, DONG H. AHN, BRONIS R. DE SUPINSKI, TODD GAMBLIN, GREGORY L. LEE, MARTIN SCHULZ, SAURABH BAGCHI, MILIND KULKARNI, BOWEN ZHOU, ZHEZHE CHEN, AND FENG QIN

Debugging High-Performance Computing Applications at Massive Scales

BREAKTHROUGHS IN SCIENCE and engineering are increasingly made with the help of high-performance computing (HPC) applications. From understanding the process of protein folding to estimating short- and long-term climate patterns, large-scale parallel HPC simulations are the tools of choice. The applications can run detailed numerical simulations that model the real world. Given the great public importance of such scientific advances, the numerical correctness and software reliability of these applications is a major concern for scientists.

Debugging parallel programs is significantly more difficult than debugging serial programs; human cognitive abilities are overwhelmed when dealing

» key insights

- Bugs in parallel HPC applications are difficult to debug because errors propagate quickly among compute nodes, programmers must debug thousands of nodes or more, and bugs might manifest only at large scale.
- Although conventional approaches like testing, verification, and formal analysis can detect a variety of bugs, they struggle at massive scales and do not always account for important dynamic properties of program execution.
- Dynamic analysis tools and algorithms that scale with an application's input and number of compute nodes can help programmers pinpoint the root cause of bugs.

IMAGE COURTESY OF UNM RESEARCH DATA SERVICES

with more than a few concurrent events.[12] When debugging a parallel program, programmers must check the state of multiple parallel processes and reason about many different execution paths. The problem is exacerbated at large scale when applications run on top supercomputers with millions of concurrently executing processes. Traditional debugging tools scale poorly with massive parallelism, as they must orchestrate execution of a large number of processes and collect data from them efficiently. The push toward exascale computing has increased the need for scalable debugging techniques.

This article describes the fundamental advances being made in designing scalable debugging techniques, sharing the authors' experience applying their solutions to current petascale supercomputers. It outlines a set of techniques that build on one another to accomplish large-scale debugging: discovery of scaling bugs, or those that manifest themselves when the application is deployed at large scale, behavioral debugging by modeling control-flow behavior of tasks,[a] and software-defect detection at the communication layer. It also describes support tools critical to higher-level debugging tools and how they may evolve into the exascale era. Finally, it discusses the three broad open problems in this domain: programmability, performance bugs, and silent data corruptions.

a We use the words "task" and "process" interchangeably as elements of parallel applications.

Background

Most large-scale HPC applications use the Message Passing Interface (MPI) to communicate among parallel processes in a distributed-memory model. The MPI standard provides more than 300 functions,[19] including data-transfer

operations (such as `send` and `receive` for point-to-point transfer and broadcasts for collective transfer), and process synchronization operations (such as barriers). Many functions have subtle semantics (such as pairing calls in different processes); for example, a bug can be introduced by incorrectly copying a buffer from the application to the MPI library or by sending a message on one process without properly calling a `receive` function on a peer.

Software bugs originate not only in the application but also in the MPI library itself; for example, the MPI library might corrupt data while moving it between buffers in the application and buffers in the MPI library or misuse a lower-level communication API.[21] These bugs can be devastating in a production supercomputing environment, as the majority of parallel applications depend on the MPI library. In addition, many MPI library implementations exploit the standard's own flexibility to make aggressive optimizations to improve performance, possibly leading to unexpected numerical non-deterministic behavior; for example, a computation might produce different results each time it executes. Although they are not bugs per se, such unexpected non-determinism hinders debugging MPI applications, especially at large scale.

Scaling bugs. An insidious class of bugs specific to large-scale parallel programs, or "scaling bugs," manifests only when an application is run at large scale. Scientific simulations are generally modeled and tested at small scale first. Then, when scientists believe the application is correct, they submit campaigns of large-scale production runs (such as with millions of parallel processes). Since resource constraints limit testing, and parallel program behavior is scale- and input-dependent, scaling bugs are often not encountered before these campaigns. Most debugging tools in HPC, including traditional breakpoint-based debugging and relative debugging tools, struggle at such massive scales. Although advances in relative debugging allow scalable comparisons of program runs,[6] the process of finding bugs remains fundamentally manual. Most HPC centers provide access to commercial debuggers like TotalView and DDT, but, while they work at large scale to some degree, they are optimized for small- to medium-size. Further research is required to develop more-effective, more-automated scalable debugging methods like those explored in this article.

A challenge in developing large-scale applications is finding bugs that are latent at small scales of testing but manifest when the application is deployed at a large scale.

Scaling bugs appear for various reasons (overflow errors, resource exhaustion, and incorrect execution flow are common) and can lead to correctness and performance errors. Figure 1 is a sample bug that appeared in the implementation of the `MPI_Allgather` routine in the MPICH[10] MPI implementation. Each parallel process gathers information from its peers through different communication topologies (such as ring and balanced binary tree) depending on the total amount of data to be exchanged. At a large-enough scale, an overflow occurs in the (internal) variable that stores the total size of data exchanged; see the line with the `if` statement. Consequently, it uses a non-optimal communication topology that leads to a slowdown of the target application. Determining that the root cause of this bug is a conditional overflow in a single MPI call has indeed proved challenging and time consuming.

Consequences of software defects. Software defects in HPC applications cause various levels of inconvenience. Hangs and crashes are common bug manifestations, but their causes can be difficult to pinpoint because tight communication dependencies between processes are characteristic of HPC applications; an error in one process spreads quickly to others. Race conditions and deadlocks are challenging because they often become visible at large scales, where messages are more likely to interleave in different, untested orderings. Performance degradation leads to suboptimal use of expensive supercomputing facilities, affecting the power budget at HPC facilities and the rate at which scientific questions can be answered. Numerical errors are among the most perplexing and deleterious, as they directly affect the validity of scientific discoveries. Scientists need effective, easy-to-use debugging tools to isolate these problems quickly.

Conventional approaches: software testing, verification, concurrency bugs. Software verification and testing are two conventional approaches for finding errors (such as by generating

high-coverage tests). Such techniques have gained attention recently due to the significant advances in "constraint satisfiability" that permits development of practical symbolic execution techniques.[4] Debugging techniques for both serial code[13] and parallel code[27] have existed for many years, allowing developers to find programming and runtime errors. The main limitation of these techniques, with respect to the challenges we present here, is they focus on single-machine applications (the shared-memory programming model) or small- to medium-scale (with a few 10s of concurrent execution threads). In this article, we describe techniques for distributed-memory MPI applications that can operate at massive scales (such as millions of processes).

Static versus dynamic analysis. We distinguish between two classes of program analysis for debugging: static and dynamic analysis. Static analysis attempts to find bugs in a program without running it (such as by analyzing the source code), whereas dynamic analysis does the same by running the program (such as by analyzing traces that are obtained at runtime). Static-analysis debugging tools can catch a variety of defects but are inadequate for debugging large-scale parallel computing applications. They lack runtime information (such as exchanged messages among processes), which is a major disadvantage, as this information can be crucial in understanding the propagation of errors.

To illustrate the problem, consider "static program slicing," a widely used static-analysis technique in debugging and software testing.[26] Slicing uses data-flow and control-flow analysis to identify a set of program statements, or the "slice," that could affect the value of a variable at some point of interest in the program. Given an erroneous statement, developers use slicing to identify code that could have (potentially) caused the statement to fail. Consider Figure 2, where an MPI program has an error in statement 10 when saving the results of some computation. It shows a static slice on lines 8–10, because the `result` variable, used in line 10, depends on statements 8 and 9. However, due to the single-program-multiple-data nature of MPI applications, the programmer must account for data propagation among processes and highlight lines 3–6 as possible sources of the bug. The programmer could statically identify which `send` operation could possibly match each `receive` operation and use the match for static slicing, but this would be too inaccurate (and as a consequence inefficient) since matching parameters (such as source, destination, and tag) are often variables. Matching can be done accurately only through dynamic-analysis techniques, our focus here.

Most slicing techniques for debugging message-passing programs thus use dynamic slicing, which considers a particular program execution, including communicated data. However, dynamically slicing a message-passing program usually does not scale well; computing a dynamic slice for each MPI process takes at least $O(p)$, where p is the number of processes. Further, dynamic slicing imposes a high computational cost to generate traces of each task (typically by code instrumentation) and aggregate those traces centrally to construct the slice. To avoid these limitations, the dynamic-analysis techniques presented here leverage lightweight tracing (such as stack tracing), as well as distributed and scalable trace merging.

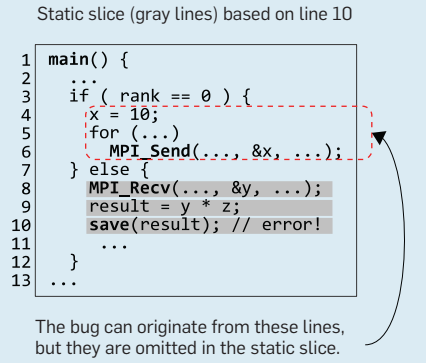
Complementary approach: Formal methods for MPI. Formal analysis of MPI programs[9] can detect a variety of bugs; for example, ISP[24] provides formal coverage guarantees against deadlocks and local safety assertions. A major limitation of these tools is that rule checking incurs high computational overhead due to state explosion. In addition, in implementations, MPI message scheduling is performed in a centralized engine, limiting scalability. The accuracy of the tools can be sacrificed to achieve better scalability for specific checks. However, these approaches have been shown to work up to only a few thousand processes; more research is needed to increase their capability for larger process counts. To detect a wider spectrum of bugs, formal analysis tools can be coupled with runtime MPI checkers (such as Umpire,[25] Marmot,[14] and MUST[11]). However, the debugging process is still predominantly manual, and the tools cover only bugs in the application, not in the MPI library.

Approach overview. Here, we focus on dynamic techniques for detecting bug manifestations, or software errors,

Figure 1. Sample scaling bug in the MPICH2 library caused by an overflow in the `if` statement when run at large process counts or with a large amount of data.

```
1  int MPIR_Allgather (void *sendbuf, int sendcount, MPI_Datatype sendtype,
2     void *recvbuf, int recvcount, MPI_Datatype recvtype,
3     MPID_Comm *comm_ptr)
4  {
5     int comm_size, rank;
6     int mpi_errno = MPI_SUCCESS;
7     int curr_cnt, dst, type_size, left, right, jnext, comm_size_is_pof2;
8     MPI_Comm comm;
9     ...
10    if ((recvcount*comm_size*type_size < MPIR_ALLGATHER_LONG_MSG) &&
11       (comm_size_is_pof2 == 1)) {
12       /* BUG IN ABOVE CONDITION CHECK DUE TO OVERFLOW */
13
14       /* Use recursive doubling algorithm for small messages*/
15    }
16    ...
17 }
```

Figure 2. Debugging example using static analysis.



in large-scale parallel applications that use MPI. Our techniques aid developers by pinpointing the root cause of an error at varying granularities.

Discovering scaling bugs. A challenge in developing large-scale applications is finding bugs that are latent at small scales of testing but manifest when the application is deployed at a large scale. A scaling bug can manifest either on strong or weak scaling. Statistical techniques do not work well because they require comparison with error-free runs at the same scale as when the problem manifests itself, and such error free runs are often unavailable at large scale. We later describe recent developments for dealing with scaling bugs.

Behavior-based debugging. These techniques follow the observation that, although large-scale runs involve a large number of processes, the processes usually fit in only a few behavioral groups. Previous debuggers for MPI applications (such as the Prism debugger[23]) leveraged this notion (of behavioral grouping) to reduce the search space from thousands of processes to a few process groups to help the developer in the bug-hunting process. The techniques we present here extend this notion to build scalable and more automated debugging tools by considering the behavior across time of the processes, rather than an isolated snapshot in time.

Manifestation of a bug could be such that the behavior of a few of the processes is different from the majority of the processes. Abnormal behavior can thus be detected by identifying the abnormal processes. Later, we present `AutomaDeD`, a tool that automatically finds abnormal processes in MPI applications, modeling control-flow and the timing behavior of each MPI process as a Markov model. These models are used to automatically pinpoint suspicious processes and code regions in a scalable manner.

Our experience is that, when errors occur at large scale, only a limited number of behaviors are observed on processes, with only a few following the erroneous path where the fault first manifests. Later, we present the Stack Trace Analysis Tool, or `STAT`, which highlights these behaviors by attaching to all processes in a large-scale job, gathering stack traces, and merging the stack traces into a prefix tree to identify which processes are executing similar code.

Software defects in MPI. Although MPI is widely used in large-scale HPC applications, and its implementations are high quality in general, MPI library implementations continue to suffer from software bugs, especially when ported to new machines. Many bugs are subtle and difficult for the average

Figure 3. An overview of the `Vrisha`-based approach to detect and diagnose scaling bugs, relying on scale as a model parameter and training through error-free runs at small execution scales.

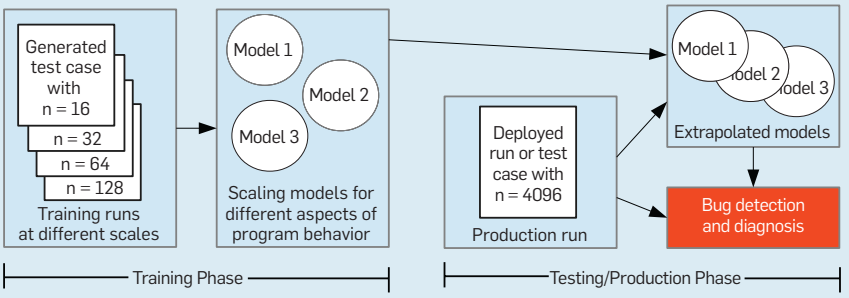


Figure 4. Overview of the `AutomaDeD` framework.

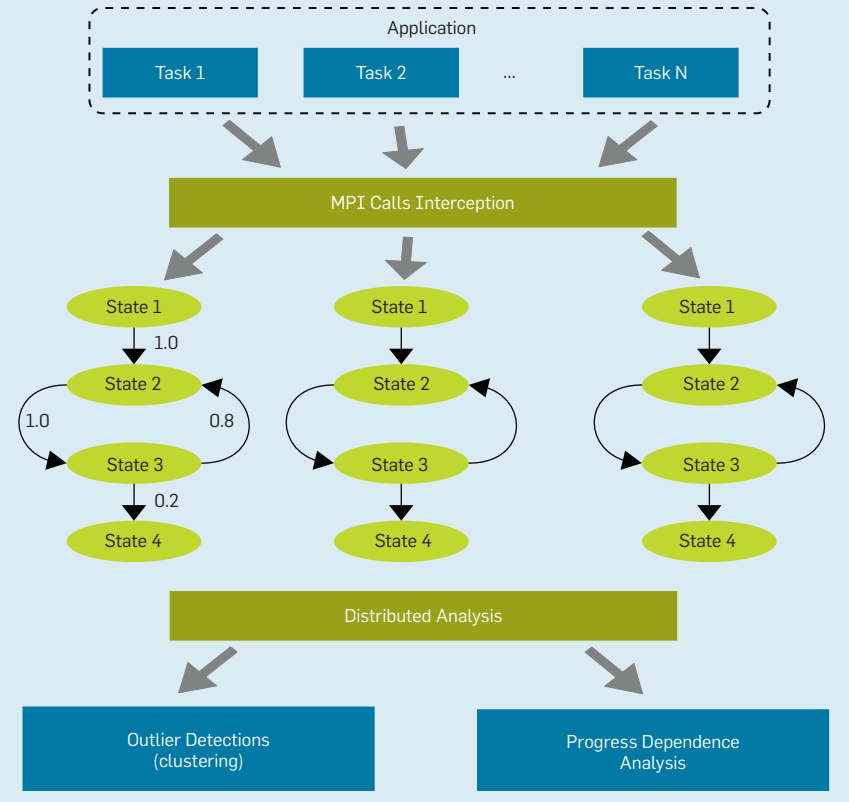
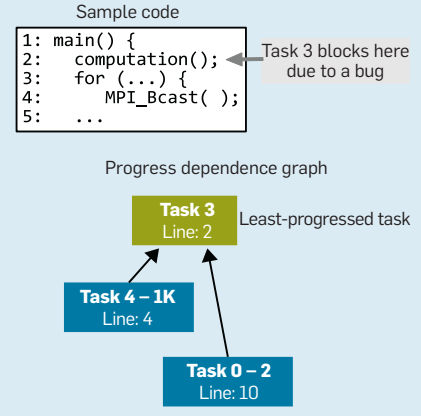


Figure 5. Progress-dependence graph.



programmer to detect.[5] Due to the wide use of MPI libraries, such bugs may affect many parallel jobs. We later present `FlowChecker`, a technique for detecting communication-related bugs in MPI libraries, checking whether messages are correctly delivered from sources to destinations by comparing the flow of messages in the library against program intentions.

Discovering Scaling Bugs

As discussed earlier, scaling bugs are difficult to handle through traditional debugging techniques and are becoming more prominent with the incessant drive to execute HPC applications at massive scales. The goal is to detect them early and narrow down the originating code.

Zhou et al.[28] developed an early attempt to retool traditional statistical debugging techniques to scaling bugs, an approach that builds a statistical model from training runs expected by the programmer to not be affected by the bug. The model captures a program's expected behavior with simple control flow statistics (such as number of times a particular branch is taken or number of times a particular calling context appears). Then, at deployment time, these statistics are measured at different points in the execution; if their values fall outside the model parameters, the deployed run is deemed to manifest a bug, since it is different from the training runs. Examination of the deviating program features localizes anomalies to find potential bugs.

Zhou et al.'s approach[28] failed to find scaling bugs for a simple but fundamental reason: If the statistical model is trained on only small-scale runs, statistical techniques can result in numerous false positives. Program behavior naturally changes as the level of concurrency increases; for example, the number of times a branch in a loop is taken depends on the number of loop iterations that can depend on the scale. Small-scale models will incorrectly label correct behaviors at large scales as anomalous, and therefore erroneous. This effect is particularly insidious with strong scaling, where scaling up divides work among more and more processes, but each process does progressively less work.

The most successful techniques for handling scaling bugs use scale as a parameter of the model and rely on the availability of bug-free runs at small scales.[28,29] `Vrisha`, a framework using many small-scale training runs, is an example of such a technique, building a statistical model (based on Kernel Canonical Correlation Analysis) from these runs to infer the relationship between scale and program behavior. The technique, in essence, builds a scaling model for the program that can extrapolate the aggregated behavioral trend as the input or system size scales up. Bugs can be detected automatically by identifying deviations from the trend. This detection can occur even if the program is run at a previously unmeasured scale; Figure 3 outlines this approach.

To isolate bugs on the largest supercomputers, programmers need scalable distributed algorithms that can isolate anomalous behaviors quickly.

However, `Vrisha` identifies only that the scaling trend has been violated, not which program feature violated the trend or where in the program the bug was manifest. The `WuKong` framework solved this limitation by observing that, if all features are combined and modeled (as with `Vrisha`), they cannot be "reverse-mapped" to identify which feature caused the deviation. `WuKong` therefore uses per-feature regression models, built across multiple training scales that can accurately predict the expected bug-free behavior at large scales. When presented with large-scale execution, `WuKong` uses these models to infer what the value of each feature would have been in a bug-free run. If any value deviates from the prediction, `WuKong` detects a bug. `WuKong` identifies the lines of code that result in unexpected behavior by ranking features based on their prediction error; program features are chosen carefully so they can be linked to particular regions of code. This ranked list provides a roadmap programmers can use to track down the bug.

Scaling models cannot predict program behaviors (or features) that do not correlate with scale. These `WuKong`-related techniques use cross-validation techniques to prune features that are difficult to model accurately from the training runs, an approach that limits the behaviors they can predict.

Behavior-Based Debugging

An application bug can manifest in such a way that the behavior of a few buggy parallel tasks is different from

that of the majority of the tasks. We leverage this observation to focus the developer's attention on the few buggy tasks, rather than requiring an exhaustive investigation. Our approach builds a behavioral model for tasks that allows us to compare the behavior of one task with another. To isolate bugs on the largest supercomputers, programmers need scalable distributed algorithms that can isolate anomalous behaviors quickly. We have implemented all of them in our AutomaDeD[b] framework.[3,16]

b Automata-based Debugging for Dissimilar parallel tasks.

AutomaDeD uses a simple model of task behavior that captures timing information and patterns in each task's control flow. The timing information allows programmers to detect performance problems, and the control flow model allows them to isolate them to particular code regions; Figure 4 is an overview of AutomaDeD. On each concurrent task, a measurement tool builds a model of execution at the granularity of code blocks and execution paths between them. To make our model lightweight, we did not design it to model basic blocks but to model code regions between communication (MPI) calls. AutomaDeD intercepts MPI calls dynamically, building its model of each task from these instrumentation functions. Each task's behavior is stored as a semi-Markov model (SMM). States in the model represent communication code regions, or code within MPI routines, and computation code regions, or code executed between two MPI communication routines. A state consists of a call-stack trace sampled at each MPI call entry. The stack trace gives the dynamic calling context of each communication call. SMM edges represent transfer of control between two states. The modeling strategy we designed into AutomaDeD assigns two attributes to each edge: a transition probability and a distribution of execution times. The distribution models the time spent in the source state for each dynamic invocation where control is transferred next to the destination state.

Given per-task models, AutomaDeD identifies anomalous behaviors through a clustering algorithm that takes as input a set of per-task models and a (dis)similarity metric that compares two models. The algorithm outputs groups of tasks with similar models. Clusters of tasks are considered unusual if AutomaDeD does not expect them based on the clusters in error-free training runs, or error-free iterations of the same run, in an unsupervised setting. For example, consider a master-slave parallel application that, in an error-free run, can be clustered into two behavioral clusters: a master cluster and a slave cluster. If AutomaDeD finds a third cluster in a faulty run, it flags it as unusual, focusing attention on the tasks in the cluster. A similar algorithm identifies the state (or code region) in which the error first manifests in those tasks. We have found other techniques to identify the task outliers are also useful, including in finding tasks abnormally far from its cluster center and tasks on average far from their neighbors using nearest-neighbor algorithms.[16]

Hangs and deadlocks are common bug manifestations in HPC applications and notably difficult to diagnose at massive scale. Large-scale HPC applications are tightly coupled (such as an MPI collective operation involving participation by multiple tasks), so a fault in one task quickly propagates to

Figure 6. View of STAT for a run on more than one million MPI processes.

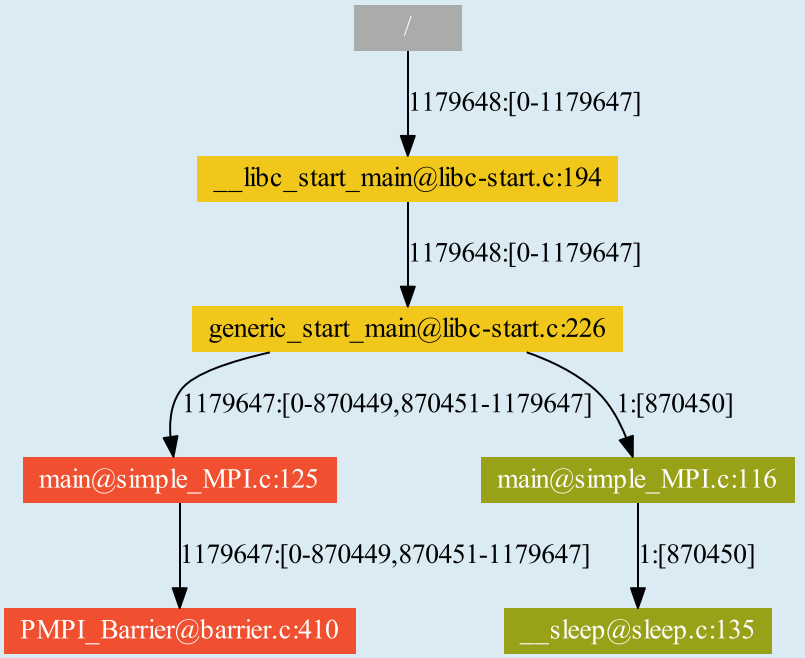
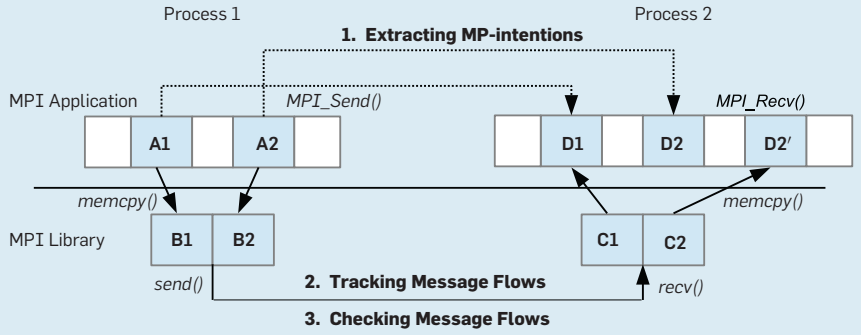


Figure 7. An example of how FlowChecker is used to find bugs in MPI communication libraries.



others. We developed the concept of a progress-dependence graph (PDG) of tasks that captures the partial ordering of tasks with respect to their progress toward the final computation result.[15,20] The PDG is then used to determine the least-progressed task, the likely root cause of the problem. AutomaDeD uses Markov models to create the PDG. Progress dependencies are calculated based on how the MPI tasks visit different application states. For example, suppose two tasks, x and y, are blocked on different states, S_x and S_y, respectively. If y has visited S_x before (and no path exists from y to x in the model), y probably depends on x to make progress, and an edge is created in the PDG from y to x. Figure 5 is a snippet of a PDG with three classes of tasks (0–2, 3, 4–1,000). Task 3 has made the least progress according to the PDG and will then be examined further (as root cause of the fault).

The combination of local, per-task models, and distributed outlier and progress analysis allows AutomaDeD to focus developers' attention on the erroneous period of time, parallel task, and specific code regions affected by faults and performance problems in parallel applications. This focus substantially reduces the bug search space, relieving the developer of having to compare potentially millions of tasks and even more possible execution paths.

All detection and diagnosis analysis steps in AutomaDeD (including outlier detection and progress dependence) are performed in a distributed fashion and thus scale to large numbers of parallel tasks.[16] Outlier task isolation uses scalable clustering,[7] and progress dependence is calculated through a combination of binomial tree-based reductions and per-task local analysis. The complexity of these procedures with respect to the number of parallel tasks is, in the worst case, logarithmic. AutomaDeD has been used to diagnose the origin of real-world bugs that appear only at large scales. Laguna et al.[15] detailed how we used AutomaDeD to identify the offending task when a hang manifested in a molecular dynamics application with more than 8,000 tasks on the BlueGene/L supercomputer.

Stack Trace Analysis Tool

The Stack Trace Analysis Tool (STAT)[2] is a lightweight, highly scalable debugging tool for identifying errors in code running on the world's largest supercomputers. STAT gathers stack traces from a parallel application and merges them into a call-graph prefix tree. The merge operation groups together traces from different processes with the same calling sequence, labeling the edges of these merged traces with the count and set of MPI ranks that exhibited that calling sequence (see Figure 6). Further, nodes in the prefix tree visited by the same set of processes are given the same color, giving the programmer a quick means to identify the various process equivalence classes.

Process-equivalence classes are STAT's main debugging idiom to help programmers focus on a manageable number of processes. Bulk-synchronous applications, particularly in a hang state, typically demonstrate a small number of classes behaving in different ways, with a few processes taking an anomalous call path, a similarly small number of processes waiting for point-to-point communication, and the remaining processes stuck in a collective operation. Programmers can thus effectively debug a large-scale application, even with more than one million MPI processes,[17] by focusing on a small subset of tasks, in particular, a single representative of each equivalence class.

While conceptually simple, STAT is effective at isolating highly elusive errors that often emerge in production computing environments (see the sidebar "Real-World Debugging Case"). In addition, STAT development, deployment, and use give programmers a blueprint for scalable tools. On extreme-scale systems, we learned an effective tool must scale in several dimensions. For one, it must be capable of presenting large amounts of debug information without overwhelming programmers. Equally important, it must become a highly scalable application on its own, collecting, managing, and reducing debug data without suffering a scalability bottleneck.

To become a highly scalable application, STAT was designed on scalable infrastructures (such as Launch-MON[1] for tool start-up and bootstrapping and the MRNet[22] tree-based overlay network for communication). Even so, as we have tested and deployed it on increasingly larger systems, scalability bottlenecks have continually surfaced. We have thus had to keep innovating to improve its key attributes, from internal data representations[18] to file access patterns and testing methods.[18]

Real-World Debugging Case

A scientist experienced hangs in a laser-plasma interaction code (named PF3D) when scaling it to 524,288 MPI processes on Lawrence Livermore National Laboratory's Sequoia BlueGene/Q system. The scientist spent months trying to debug the problem through print statements to no avail and was reluctant to run a traditional parallel debugger at such large scale. Moreover, the scientist was unable to reproduce the hang at smaller scales where these fully featured, heavyweight debuggers would be more plausible.

However, within a few minutes of launching STAT, the scientist could see the application was deadlocked in a computation-steering module. Working with colleagues, the scientist analyzed STAT's output to determine the hang was the result of a race condition between two distinct but overlapping communication regions, the result of PF3D migrating from one version to another more scalable but incompatible one. During migration, PF3D ran through a compatibility layer that introduced the race condition and ultimately caused these timing- and scale-dependent hangs. Following the analysis, the scientist expedited the task of porting to the new version, after which the simulations could proceed without hangs.

STAT reported the precise location of the hang in a few minutes, and a few hours of analysis later determined the cause of the hang. By contrast, the same scientist had previously spent months of effort, and perhaps millions of CPU hours, and was still unable to identify the problem through the common procedure of print debugging.

This is just one example of many problems STAT is designed to help debug at large scales, with others extending beyond 1.5 million MPI processes and problems ranging from user-application errors to system software bugs and even to hardware faults.

Software Defects in MPI

Bugs in MPI libraries can be devastating in a production supercomputing environment, as most parallel supercomputing applications depend on the MPI library. MPI bugs can be especially frustrating for application developers since they could appear to be bugs in client code. Further, these bugs can be tedious for library developers to catch and fix.[21] MPI users' machines can have architectures or compilers that differ from library developers' machines. Worse, the library developers may not have a system as large as the one on which the bug manifests. Certain bugs occur only on large-scale systems.[2] Others manifest only in large MPI applications, and if the applications are sensitive or proprietary, users usually have to generate a small reproducing test program to send to developers, a possibly time-consuming process. In many cases, library developers may simply be unable to reproduce a scaling bug.

`FlowChecker` is a low-overhead method for detecting bugs in MPI libraries[5] (see Figure 7). Its main function is to check whether the underlying MPI libraries correctly deliver messages from the sources to the destinations as specified by the MPI applications. Like a delivery service's package-tracking system, `FlowChecker`'s detection process includes extraction of message-passing intention (source and destination addresses), message flow tracking (package transmission and delivery), and message-delivery verification (user confirmation).

`FlowChecker` first extracts the intentions of message passing ("MP-intentions") from MPI applications. Normally, MP intentions are implied by MPI function calls and corresponding arguments in MPI applications; for example, `FlowChecker` can extract MP intentions based on a matched pair of `MPI_Send` and `MPI_Recv` function calls collected by instrumenting the MPI applications.

Moreover, for each MP intention, `FlowChecker` tracks the corresponding message flows by following relevant data-movement operations, starting from sending buffers at the source process. Data movement-operations move data from one memory location to another within one process or between two processes.[5,8] Examples of data movement include memory copy and network send/receive collected by instrumenting the MPI libraries. This step allows `FlowChecker` to understand how MPI libraries perform message transmission and delivery.

Finally, `FlowChecker` checks message flows (from the second step) against MP intentions (from the first step). If `FlowChecker` finds a mismatch, it reports a bug and provides further diagnostic information (such as faulty MPI functions or incorrect data movements).

In Figure 7, which reflects the main idea of `FlowChecker`, process 1 invokes `MPI_Send` to send a message stored at the buffer {`A1`, `A2`} to process 2, and process 2 invokes `MPI_Recv` to receive the message and store the message at the buffer {`D1`, `D2`}. The MP-intention is {`A1`→`D1`, `A2`→`D2`}, and the message flows are `A1`→`B1`→`C1`→`D1` and `A2`→`B2`→`C2`→`D2′`. Comparing the MP-intention with the corresponding message flow, `FlowChecker` detects the data in `A2` is delivered to an incorrect destination `D2′`, instead of `D2`, as specified by the application. Additionally, `FlowChecker` will provide further diagnosis information—the last broken message flow {`C2`→`D2′`}.

Scalability and Performance

The scalability of the techniques—`Vrisha`, `Wukong`, `AutomaDeD`, `STAT`, and `FlowChecker`—is mainly a function of algorithmic complexity with respect to scale, including number of processes and size of input data. All are dynamic, so they first collect application traces online, or as the application runs, then process them, online or offline. A programmer would determine their performance by measuring application slowdown (caused by gathering traces) and time to process the collected traces, or analysis time. Slowdown is the ratio of the execution time with the tool to the execution time without the tool.

We have had debugging sessions with `STAT` on up to 1,572,864 MPI processes on the IBM Sequoia Blue Gene/Q supercomputer, demonstrating `STAT`'s slowdown and analysis time is small. We have used `AutomaDeD` up to 100,000 MPI processes. Its slowdown, which depends on number of MPI calls intercepted, is on average 1.2 (based on 10 tested applications), and its analysis time is less than 10 seconds for up to 100,000 tasks.

We have used `Vrisha`, `Wukong`, and `FlowChecker` at smaller scales (on the order of 1,000 processes); however, there is no fundamental limitation in their design to running them at the largest scale (such as no centralized element or algorithmically costly procedures with respect to scale). Performance is affected by having to use binary instrumentation to gather traces and storage overhead to save them. The models `Vrisha` and `Wukong` use take less than a second to train, but application slowdown can be between 1.08 and 1.3. `FlowChecker` has a very small slowdown (less than 1.1), but the size of the traces can grow quickly, with application execution causing data rates between 1.77 MB/min and 10.08 MB/min. For tools like `AutomaDeD` this is not a problem since traces are processed online.

Conclusion

We have argued that a robust set of dynamic debugging tools is required for deploying large-scale parallel applications, complementing ongoing work in static analysis, relative debugging, and formal methods for development of correct parallel applications. We surveyed the state-of-the-art techniques for dynamic debugging. The first class of techniques we discussed targets software bugs that arise only when the application runs at large scales, using it to extrapolate from behavior at small-scale runs that are assumed correct. The second class is behavior-based detection via clustering, where behavior includes control flow and timing of a certain granularity of code regions. A sub-class within this technique collects stack traces from each individual process and then, using the insight the stack traces exhibit great commonality, merges multiple traces into one, giving a developer a limited, manageable amount of information. The third class targets bugs in communication libraries by verifying program intent matches observed behavior. These software packages build on robust, usable tools that include stack tracing, dynamic binary instrumentation, scalable data analytics (such as clustering), and runtime profiling.

Looking ahead, discoveries are needed in three main research directions.

First, higher-level software development environments must be created for parallel programming to enable faster development with fewer defects. These environments should support standard principles of software engineering (such as refactoring, code completion, quick-fix support, and intuitive user interfaces). Second, debugging support must be deployed to address performance bugs, in addition to the current focus of correctness bugs; application models must thus include performance measures and support extrapolation of performance characteristics in various dimensions (such as larger process counts, larger data sizes, and different datasets). Third, the programmer's attention must focus on data corruption resulting from software bugs that are often not within the purview of existing detection techniques or do not cause hangs, crashes, or performance slowdowns but "silently" corrupt the data output. Such bugs lead to the specter of incorrect science all programmers surely wish to avoid.

Dynamic debugging techniques have spurred significant innovation and robust tool development. We will build on this promising base to address further challenges, some we have explored here.

Acknowledgments

The research and development related to this article was performed partially under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DEAC52-07NA27344 (LLNL-JRNL-652400) and support from National Science Foundation awards CNS-0916337, CCF-1337158, CCF-0953759, and CNS-0403342.

References

1. Ahn, D.H., Arnold, D.C., de Supinski, B.R., Lee, G.L., Miller, B.P., and Schulz, M. Overcoming scalability challenges for tool daemon launching. In *Proceedings of the International Conference on Parallel Processing* (Portland, OR, Sept. 8–12). IEEE Press, 2008, 578–585.
2. Arnold, D.C., Ahn, D.H., de Supinski, B.R., Lee, G.L., Miller, B.P., and Schulz, M. Stack trace analysis for large-scale debugging. In *Proceedings of the International Parallel and Distributed Processing Symposium* (Long Beach, CA, Mar. 26–30). IEEE Press, 2007, pages 1–10.
3. Bronevetsky, G., Laguna, I., Bagchi, S., de Supinski, B.R., Ahn, D.H., and Schulz, M. `AutomaDeD`: Automata-based debugging for dissimilar parallel tasks. In *Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems and Networks* (Chicago, IL, June 28–July 1). IEEE Press, 2010, 231–240.
4. Cadar, C. and Sen, K. Symbolic execution for software testing: three decades later. *Commun. ACM 56*, 2 (Feb. 2013), 82–90.
5. Chen, Z., Gao, Q., Zhang, W., and Qin, F. `FlowChecker`: Detecting bugs in MPI libraries via message flow checking. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis* (New Orleans, LA, Nov. 13–19). IEEE Computer Society, Washington, D.C., 2010, 1–11.
6. Dinh, M.N., Abramson, D., and Jin, C. Scalable relative debugging. *IEEE Transactions on Parallel and Distributed Systems 25*, 3 (Mar. 2014), 740–749.
7. Gamblin, T., De Supinski, B.R., Schulz, M., Fowler, R., and Reed, D.A. Clustering performance data efficiently at massive scales. In *Proceedings of the 24th ACM International Conference on Supercomputing* (Tsukuba, Ibaraki, Japan, June 1–4). ACM Press, New York, 2010, 243–252.
8. Gao, Q., Qin, F., and Panda, D.K. DMTracker: Finding bugs in large-scale parallel programs by detecting anomaly in data movements. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing* (Reno, NV, Nov. 10–16). ACM Press, New York, 2007, 15:1–15:12.
9. Gopalakrishnan, G., Kirby, R.M., Siegel, S., Thakur, R., Gropp, W., Lusk, E., De Supinski, B.R., Schulz, M., and Bronevetsky, G. Formal analysis of MPI-based parallel programs. *Commun. ACM 54*, 12 (Dec. 2011), 82–91.
10. Gropp, W., Lusk, E., Doss, N., and Skjellum, A. A high-performance, portable implementation of the MPI message-passing interface standard. *Parallel Computing 22*, 6 (1996), 789–828.
11. Hilbrich, T., Schulz, M., de Supinski, B.R., and Müller, M.S. MUST: A scalable approach to runtime error detection in MPI programs. Chapter 5 of *Tools for High Performance Computing 2009*, M.S. Müller et al., Eds. Springer, Berlin, Heidelberg, 2010, 53–66.
12. Kieras, D.E., Meyer, D.E., Ballas, J.A., and Lauber, E.J. Modern computational perspectives on executive mental processes and cognitive control: Where to from here? Chapter 30 of *Control of Cognitive Processes: Attention and Performance*, S. Monsell and J. Driver, Eds. MIT Press, Cambridge, MA, 2000, 681–712.
13. Kinshumann, K., Glerum, K., Greenberg, S., Aul, G., Orgovan, V., Nichols, G., Grant, G., Loihle, G., and Hunt, G. Debugging in the (very) large: 10 years of implementation and experience. *Commun. ACM 54*, 7 (July 2011), 111–116.
14. Krammer, B., Müller, M.S., and Resch, M.M. MPI application development using the analysis tool MARMOT. In *Proceedings of the Fourth International Conference on Computational Science*, M. Bubak et al., Eds. (Kraków, Poland, June 6–9). Springer, Berlin, Heidelberg, 2004, 464–471.
15. Laguna, I., Ahn, D.H., de Supinski, B. R., Bagchi, S., and Gamblin, T. Probabilistic diagnosis of performance faults in large-scale parallel applications. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques* (Minneapolis, MN, Sept. 19–23). ACM Press, New York, 2012, 213–222.
16. Laguna, I., Gamblin, T., de Supinski, B.R., Bagchi, S., Bronevetsky, G., Ahn, D.H., Schulz, M. and Rountree, B. Large-scale debugging of parallel tasks with `AutomaDeD`. In *Proceedings of 2011 International Conference on High Performance Computing, Networking, Storage, and Analysis* (Seattle, WA, Nov. 12–18). ACM Press, New York, 2011, 50:1–50:10.
17. Lee, G.L., Ahn, D.H., Arnold, D.C., de Supinski, B.R., Legendre, M., Miller, B.P., Schulz, M., and Liblit, B. Lessons learned at 208K: Towards debugging millions of Cores. In *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage, and Analysis* (Austin, TX, Nov. 15–21). IEEE Press, Piscataway, NJ, 2008, 1–9.
18. Lee, G.L., Ahn, D.H., Arnold, D.C., de Supinski, B.R., Miller, B.P., and Schulz, M. Benchmarking the stack trace analysis tool for BlueGene/L. In *Proceedings of the Parallel Computing: Architectures, Algorithms, and Applications Conference* (Julich/Aachen, Germany, Sept. 4–7). IOS Press, Amsterdam, the Netherlands, 2007, 621–628.
19. Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard, Version 3.0*, Sept. 2012; http://www.mpi-forum.org/docs/
20. Mitra, S., Laguna, I., Ahn, D.H., Bagchi, S., Schulz, M., and Gamblin, T. Accurate application progress analysis for large-scale parallel debugging. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Edinburgh, U.K., June 9–11). ACM Press, New York, 2014, 1–10.
21. Open MPI Project; https://svn.open-mpi.org/trac/ompi/ticket/689.
22. Roth, P.C., Arnold, D.C., and Miller, B.P. MRNet: A software-based multicast/reduction network for scalable tools. In *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing* (Phoenix, AZ, Nov. 15–21). ACM Press, New York, 2003, 21.
23. Sistare, S., Allen, D., Bowker, R., Jourdenais, K., Simons, J. et al. A scalable debugger for massively parallel message-passing programs. *IEEE Parallel & Distributed Technology: Systems & Applications 2*, 2 (Summer 1994), 50–56.
24. Vakkalanka, S.S., Sharma, S., Gopalakrishnan, G., and Kirby, R.M. ISP: A tool for model checking MPI programs. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Salt Lake City, UT, Feb. 20–23). ACM Press, New York, 2008, 285–286.
25. Vetter, J.S. and de Supinski, B.R. Dynamic software testing of MPI applications with Umpire. In *Proceedings of the ACM/IEEE Supercomputing Conference* (Dallas, TX, Nov. 4–10). IEEE Press, 2000, 51–51.
26. Weiser, M. Program slicing. In *Proceedings of the Fifth International Conference on Software Engineering* (San Diego, CA, Mar. 9–12). IEEE Press, Piscataway, NJ, 1981, 439–449.
27. Yang, J., Cui, H., Wu, J., Tang, Y., and Hu, G. Making parallel programs reliable with stable multithreading. *Commun. ACM 57*, 3 (Mar. 2014), 58–69.
28. Zhou, B., Kulkarni, M., and Bagchi, S. `Vrisha`: Using scaling properties of parallel programs for bug detection and localization. In *Proceedings of the 20th International ACM Symposium on High-Performance and Distributed Computing* (San Jose, CA, June 8–11). ACM Press, New York, 2011, 85–96.
29. Zhou, B., Too, J., Kulkarni, M., and Bagchi, S. `WuKong`: Automatically detecting and localizing bugs that manifest at large system scales. In *Proceedings of the 22nd International Symposium on High-Performance Parallel and Distributed Computing* (New York, June 17–21). ACM Press, New York, 2013, 131–142.

Ignacio Laguna (ilaguna@llnl.gov) is a computer scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory, Livermore, CA.

Dong H. Ahn (ahn1@llnl.gov) is a computer scientist in the Livermore Computing Center at Lawrence Livermore National Laboratory, Livermore, CA.

Bronis R. de Supinski (bronis@llnl.gov) is the chief technology officer of Livermore Computing at Lawrence Livermore National Laboratory, Livermore, CA, a professor of exascale computing at Queen's University of Belfast, Belfast, Ireland, and an adjunct associate professor in the Department of Computer Science and Engineering at Texas A&M University, College Station, TX.

Todd Gamblin (tgamblin@llnl.gov) is a computer scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory, Livermore, CA.

Gregory L. Lee (lee218@llnl.gov) is a computer scientist in the Livermore Computing Center at Lawrence Livermore National Laboratory, Livermore, CA.

Martin Schulz (schulzm@llnl.gov) is a computer scientist in the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory, Livermore, CA, and chair of the MPI Forum, the standardization body for the Message Passing Interface.

Saurabh Bagchi (sbagchi@purdue.edu) is a professor of electrical and computer engineering and computer science at Purdue University, West Lafayette, IN.

Milind Kulkarni (milind@purdue.edu) is an assistant professor in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, IN.

Bowen Zhou (bwzhou@gmail.com) is a senior software engineer at Turn, Redwood City, CA; his research for this article was part of his Ph.D. in computer science at Purdue University, West Lafayette, IN.

Zhezhe Chen (zhezhec@twitter.com) is a software engineer at Twitter Inc., San Francisco, CA; his research for this article was part of his Ph.D. in computer science at The Ohio State University, Columbus, OH.

Feng Qin (qin@cse.ohio-state.edu) is an associate professor in the Department of Computer Science and Engineering at The Ohio State University, Columbus, OH.

DOI:10.1145/2701412

A new dynamic growth model reveals how citation networks evolve over time, pointing the way toward reformulated scientometrics.

BY TANMOY CHAKRABORTY, SUHANSANU KUMAR, PAWAN GOYAL, NILOY GANGULY, AND ANIMESH MUKHERJEE

On the Categorization of Scientific Citation Profiles in Computer Science

A CONSENSUS IN the literature is that the citation profiles of published articles follow a universal pattern—initial growth in number of citations the first two to three years following publication with a steady peak of one to two years and then decline over the rest of the lifetime of the article. This observation has long been the underlying heuristic in determining major bibliometric factors (such as quality of publication, growth of scientific communities, and impact factor of publication venue). Here, we analyze a dataset of 1.5 million computer science papers maintained by Microsoft Academic Search, finding the citation count of the articles over the years follows a remarkably diverse set of patterns—a profile with an initial peak (PeakInit), with distinct multiple peaks (PeakMul) exhibiting a peak later in time (PeakLate) that is monotonically decreasing (MonDec), monotonically increasing (MonIncr), and cannot be categorized into any other category (Oth). We conducted a thorough experiment to investigate several important characteristics of the categories, including how individual categories attract citations, how categorization is influenced by year and publication venue, how each category is affected by self-citations, the stability of the categories over time, and how much each of the categories contribute to the core of the network. Further, we show the traditional preferential-attachment models fail to explain these citation profiles. We thus propose a novel dynamic growth model that accounts for both preferential attachment and the aging factor in order to replicate the real-world behavior of various citation profiles. This article widens the scope for a serious reinvestigation into the existing bibliometric indices for scientific research, not just for computer science.

» key insights

- **Analyzing a massive dataset of scholarly papers revealed six distinctive citation profiles for papers, ranging from a single peak to multiple peaks to peaks that increase monotonically or decrease over time.**
- **Following characterization of the profiles, we found major modifications of the existing bibliographic indices could better reflect real-world citation history.**
- **Unlike existing network-growth models, these profiles can be reproduced but only if they account for "preferential attachment" and "aging."**

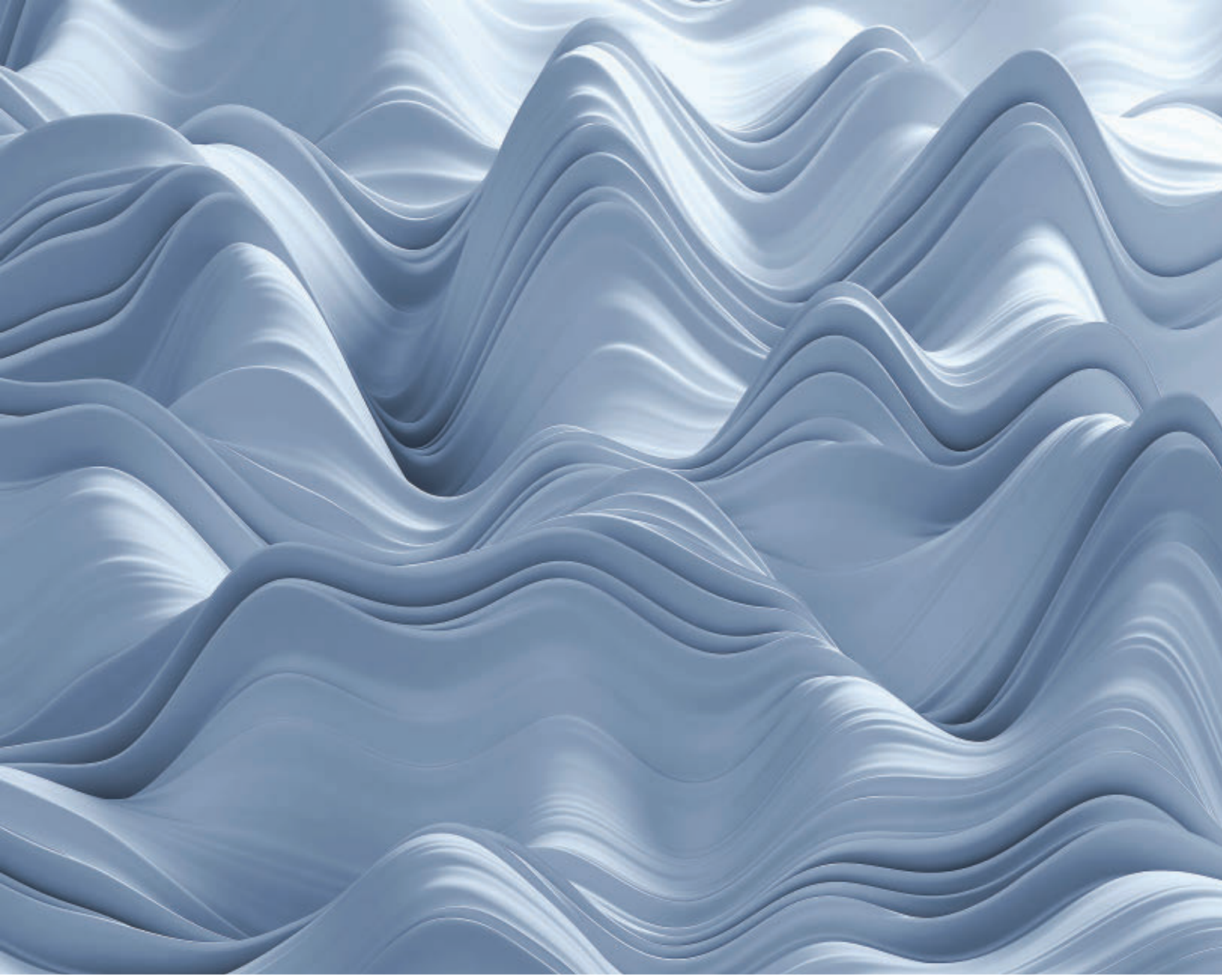


IMAGE BY WACOMKA ART

Quantitative analysis in terms of counting, measuring, comparing quantities, and analyzing measurements is perhaps the main tool for understanding the impact of science on society. Over time, scientific research itself (by recording and communicating research results through scientific publications) has become enormous and complex. This complexity is today so specialized that individual researchers' understanding and experience are no longer sufficient to identify trends or make crucial decisions. An exhaustive analysis of research output in terms of scientific publications is of great interest to scientific communities that aim to be selective, highlighting significant or promising areas of research and better managing scientific investigation.[5,24,25,27] Bibliometrics, or "scientometrics,"[3,22] or application of quantitative analysis and statistics to publications (such as research articles and accompanying citation counts), turns out to be the main tool for such investigation. Following pioneering research by Eugene Garfield,[14] citation analysis in bibliographic research serves as the fundamental quantifier for evaluating the contribution of researchers and research outcomes. Garfield pointed out a citation is no more than a way to pay homage to pioneers, give credit for related work (homage to peers), identify methodology and equipment, provide background reading, and correct one's own work or the work of others.[14]

A citation network represents the knowledge graph of science, in which individual papers are knowledge sources, and their interconnectedness in terms of citation represents the relatedness among various kinds of knowledge; for instance, a citation network is considered an effective proxy for studying disciplinary knowledge flow, is used to discover the knowledge backbone of a particular research area, and helps group similar kinds of knowledge and ideas. Many studies have been conducted on citation networks and their evolution over time. There is already a well-accepted belief among computer science scholars about the dynamics of citations a scientific article receives following publication: initial growth (growing phase) in number of citations within the first two to three years following publication, followed by a steady peak of one to two years (saturation phase), and then a final decline over the rest of the lifetime of the article (decline and obsolete phases) (see Figure 1).[15–17] In most cases, this observation is drawn from analysis of

Figure 1. Hypothetical example showing the traditional view by computer science scholars of the citation profiles of scientific papers following publication.

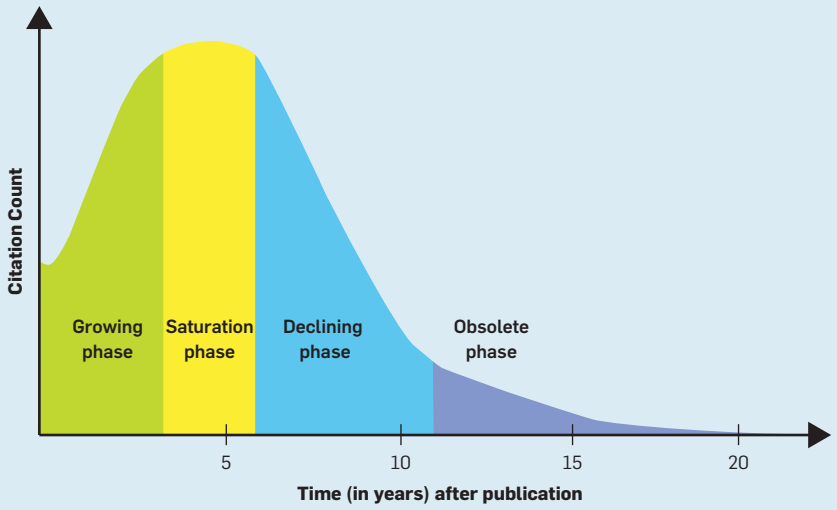
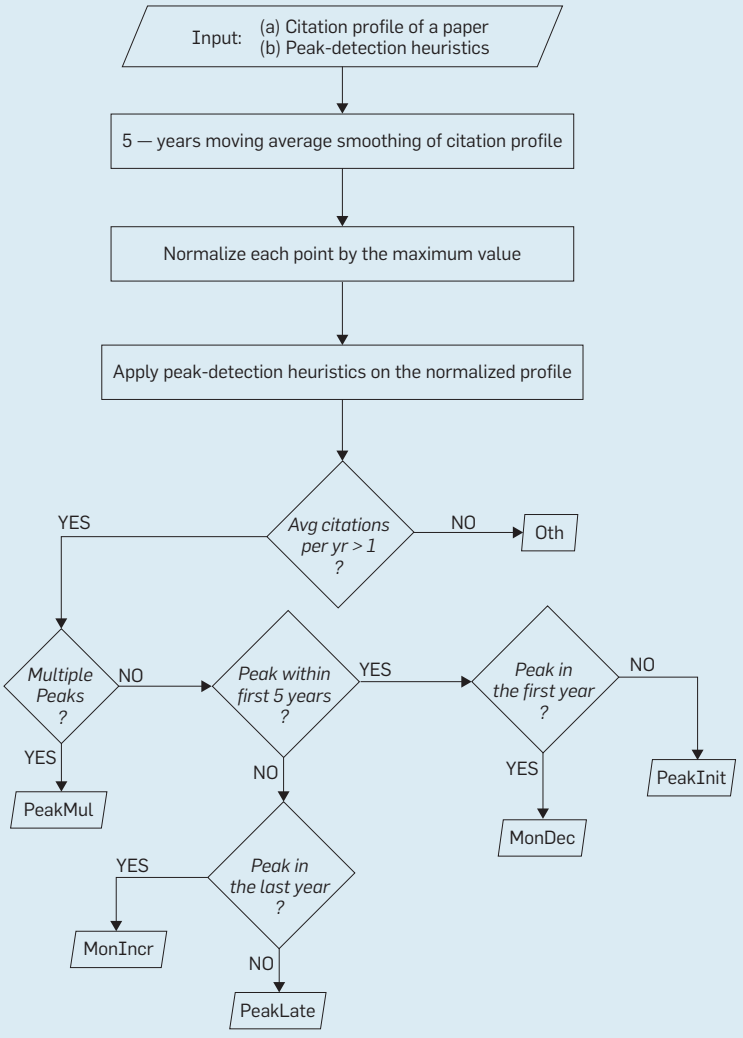


Figure 2. Systematic flowchart of the rules for classifying training samples.



a limited set of publication data,[7,13] possibly obfuscating some true characteristics. Here, we conduct our experiment on a massive bibliographic dataset in the computer science domain involving more than 1.5 million papers published worldwide by multiple journals and proceedings from 1970 to 2010 as collected by Microsoft Academic Search. Unlike earlier observations about paper citation profiles, we were able to define six different types of citation profiles prevalent in the dataset we label PeakInit, PeakMul, PeakLate, MonDec, MonIncr, and Oth. We exhaustively analyzed these profiles to exploit the microdynamics of how people actually read and cite the papers, controlling the growth of the underlying citation network unexplored in the literature. This categorization allows us to propose a holistic view of the growth of the citation network through a dynamic model that accounts for the accepted concept of preferential attachment,[1,2,26] along with the aging factor[20] in order to reproduce different citation profiles observed in the Microsoft Academic Search dataset. To the best of our knowledge, ours is the first attempt to consider these two factors together in synthesizing the dynamic growth process of citation profiles.

Our observations not only help reformulate the existing bibliographic indices (such as "journal impact factor") but enhance general bibliometric research (such as "citation link prediction," "information retrieval," and "self-citation characterization"), reflecting several characteristics:

Citation trajectory. In earlier studies, an article's citation trajectory was assumed by the research community to increase initially, then follow a downward trajectory;

Six trajectories. Analyzing the massive dataset of computer science papers, we identified six distinct citation trajectories; and

Revisit. Since citation profiles can be categorized into at least six different types, all measures of scientific impact (such as impact factor) should be revisited and updated.

Massive Publication Dataset

Most experiments in the literature

on analyzing citation profiles have worked with small datasets. In our experiment, we gathered and analyzed a massive dataset to validate our hypothesis. We crawled one of the largest publicly available datasets,[a] including, as of 2010, more than 4.1 million publications and 2.7 million authors, with updates added each week.[9] We collected all the papers published in the computer science domain and indexed by Microsoft Academic Search[b] from 1970 to 2010. The crawled dataset included more than two million distinct papers that were further distributed over 24 fields of computer science, as categorized by Microsoft Academic Search. Moreover, each paper included such bibliographic information as title, unique index, author(s), author(s) affiliation(s), year of publication, publication venue, related field(s), abstract, and keyword(s). In order to remove the anomalies that crept in due to crawling, we passed the dataset through a series of initial preprocessing stages. The filtered dataset included more than 1.5 million papers, with 8.68% of them belonging to multiple fields, or "interdisciplinary" papers; the dataset is available at http://cnerg.org (see "Resources" tab).

Categorization of Citation Profiles

Since our primary focus was analyzing a paper's citation growth following publication, we needed an in-depth understanding of how citation numbers vary over time. We conducted an exhaustive analysis of citation patterns of different papers in our dataset. Some previous experimental results[9,14] showed the trend followed by citations received by a paper following publication date is not linear in general; rather, there is a fast growth of citations within the first few years, followed by exponential decay. This conclusion is drawn mostly from analysis of a small dataset of the archive. Here, we first took all papers with at least 10 years and a maximum of 20 years of citation history, then followed a series of data-processing steps. First, to smooth the time-series data points in a paper's citation profile, we used five-year-moving-average filtering; we then scaled the data points by normalizing them with the maximum value present in the time series, or maximum number of citations received by a paper in a particular year; finally, we ran a local-peak-detection algorithm[c] to detect peaks in the citation profile. We also applied two heuristics to specify peaks: the height of a peak should be at least 75% of the maximum peak-height, and two consecutive peaks should be separated by more than two years. Otherwise we treated them as a single peak (see Figure 2).

We found most papers did not follow the traditional citation profile mentioned in the earlier studies, as in Figure 1; rather, we identified the six different types of citation profiles based on the count and position of peaks in a profile. We defined six types of citation profiles, along with individual proportions of the entire dataset:

PeakInit. Papers with citation-count peaks in the first five years following

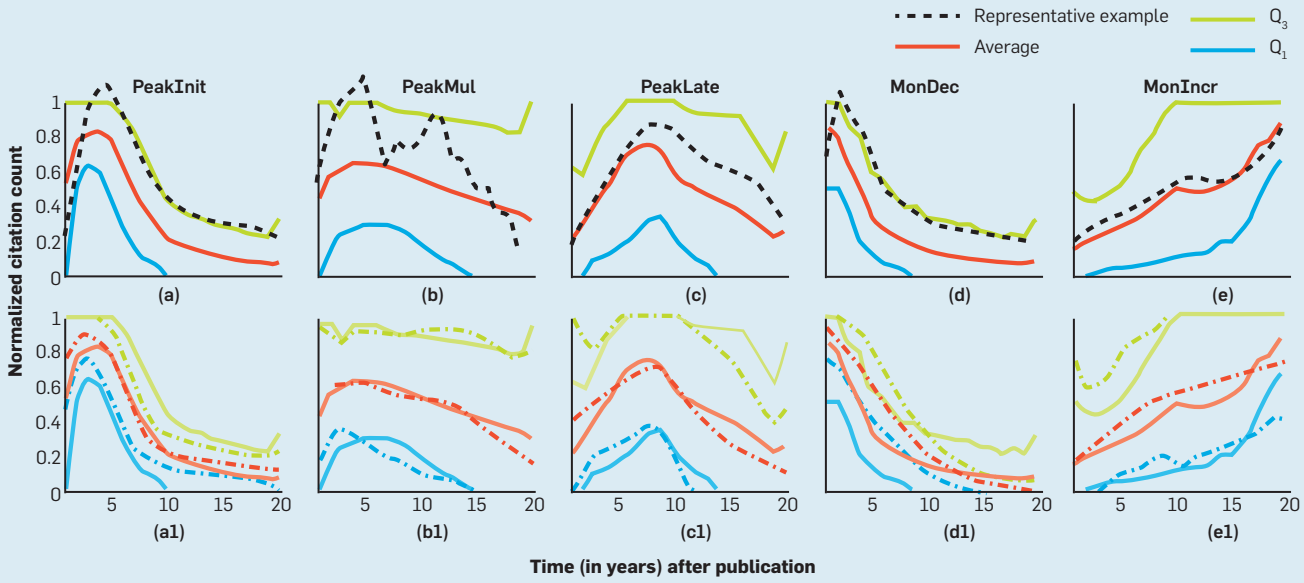
a http://academic.research.microsoft.com

b The crawling process took us approximately six weeks, concluding in August 2013.

c The peak detection algorithm is available as a Matlab Spectral Analysis http://www.mathworks.in/help/signal/ref/findpeaks.html; we used MINPEAKDISTANCE=2 and MINPEAKHEIGHT=0.75 and the default values for the other parameters.

Figure 3. Citation profiles for the first five categories we obtained from analyzing the Microsoft Academic Search citation dataset (top panel) and how it compared with the results obtained from the model (bottom panel).

Each frame corresponds to each category; Oth does not follow a consistent pattern and is not shown. In each frame, a citation belt is formed by the lines Q1 (green line) and Q3 (blue line) representing the first quartile (10% points are below the line) and third quartile (10% points are above the line) of data points, respectively, or effectively 80% points are within the citation belt. The red line within the citation belt represents the average behavior of all profiles corresponding to that category. In the top panel, for each category, one representative citation profile is shown at the middle of the belt (broken black line). The color coding in the bottom panel is similar to the top panel, though the broken lines are the results we obtained from our model.



publication (but not the first year) followed by an exponential decay (proportion = 25.2%) (see Figure 3a);

PeakMul. Papers with multiple peaks at different time points in their citation profiles (proportion = 23.5%) (see Figure 3b);

PeakLate. Papers with few citations at the beginning, then a single peak after at least five years after publication, followed by an exponential decay in citation count (proportion = 3.7%) (see Figure 3c);

MonDec. Papers with citation-count peaks in the immediate next year of publication followed by monotonic decrease in the number of citations (proportion = 1.6%) (see Figure 3d);

MonIncr. Papers with monotonic increase in number of citations from the beginning of the year of publication until the date of observation or after 20 years of publication (proportion = 1.2%) (see Figure 3e); and

Oth. Apart from the first five types, a large number of papers on average receive fewer than one citation per year; for them, the evidence is not significant enough to assign them to one of the first five categories, so they remain as a separate group (proportion = 44.8%).

The rich metadata in the dataset further allowed us to conduct a second-level analysis of the categories for multiple research fields in computer science. We thus measured the percentage of papers in different categories for each of the 24 research fields after filtering out all papers in the Oth category. We noticed that while for all other fields, the largest fraction of papers belong to the PeakMul category, for the Web this fraction is maximum in the PeakInit category (see Figure 4). A possible reason could be since the Web is mostly a conference-based research field, the papers in PeakInit generally dominate the field, as discussed later, in light of three observations:

Web. Most Web-related papers fall into the PeakInit category;

Figure 4. Percentage of papers in six categories for various research fields in computer science; the pattern is generally consistent, except for World Wide Web.

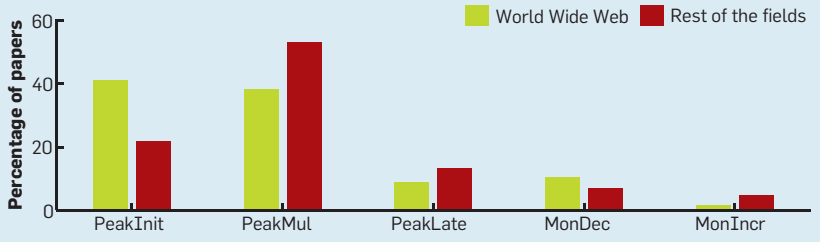
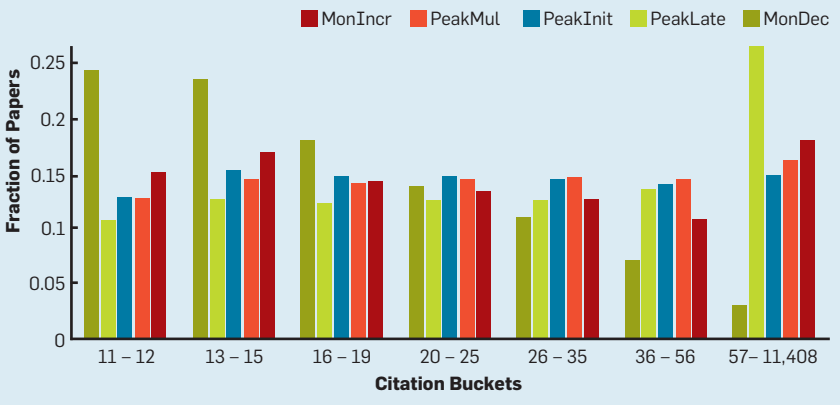


Figure 5. Contribution of papers from each category in different citation buckets. The entire range of citation value in the dataset is divided into seven buckets in which the contribution of papers from a particular category is normalized by the total number of papers in that category.



MonDec. Simulation and computer education have the largest proportion of papers in the MonDec category, and bioinformatics and machine learning have the smallest; and

PeakLate. Security and privacy, as well as bioinformatics, have the largest proportion of papers in the PeakLate category, and simulation and the Web have the smallest.

Categories in Citation Ranges

One aspect of analyzing scientific publications is determining how acceptable they are to the research community. A paper's acceptability is often measured by raw citation count; the more citations an article receives from other publications, the more it is assumed to be admired by researchers and hence the greater its scientific impact.[6] In our context, we ask, which among the six categories includes papers admired most in terms of citations? To answer, we conducted a study in which we divided total citation range into four buckets (ranges 11–12, 13–15, 16–19, 20–11,408) such that each citation bucket included an almost equal number of papers. For a deeper analysis of the highest citation range, we further divided the last bucket (20–11,408) into four more ranges, obtaining seven buckets altogether. We then measured the proportion of papers contributed by a particular category to a citation bucket (see Figure 5). Note in each citation bucket, we normalized the number of papers contributed by a category by total number of papers belonging to that category. The figure is a histogram of conditional probability distribution, the probability a randomly selected paper falls in citation bucket i given that it belongs to category j. Normalization was required to avoid population bias across different categories. Note the higher citation range is occupied mostly by the papers in the PeakLate and MonIncr categories, followed by PeakMul and PeakInit. Also note the MonDec category, which has the smallest proportion in the last citation bucket, shows a monotonic decline in the fraction of papers as citation range increases. This initial evidence suggests a general and nonintuitive interpretation of citation profiles; if a paper does not attract a large number

of citations within the immediate few years following its publication, it does not necessarily mean it will continue to be low impact through its lifetime; rather its citation growth rate might accelerate and could indeed turn out to be well accepted in the literature of computer science. We further explain this behavior in the next section.

Characterizing Different Citation Profiles

The rich metadata information in the publication dataset further allowed us to understand the characteristic features of each of the six categories at finer levels of detail.

Influence of publication year and venue on categorization. One might question whether this categorization could be influenced by the time (year) a paper is published; that is, papers published earlier might be following the well-known behavior, whereas papers published more recently might indicate a different behavior. To verify categorization is not biased by publication date, we measured the average year of publication of the papers in each category. Table 1, second column, suggests the citation pattern is not biased by year of publication, since average years correspond roughly to the same time period. On the other hand, the mode of publication in conferences differs significantly from that of journals, and the citation profiles of papers published in these two venues are expected to differ. To analyze venue effect on categorization, we measured the percentage of papers published in journals vis-à-vis in conferences for each category, as in Table 1, third and fourth columns, respectively. We observed while most of the papers in the PeakInit (64.35%) and MonDec (60.73%) categories were published in conferences, papers in PeakLate (60.11%) and MonIncr (74.74%) were published mostly in journals. If a publication starts receiving more attention or citations later in its lifetime, it is more likely to have been published in a journal and vice versa, reflecting two trends:

Conferences. Due to increasing popularity of conferences in applied domains like computer science, conference papers get quick publicity within a few years of publication and likewise quick decay of that popularity; and

Journals. Though journal papers usually take time to be published and gain popularity, most journal papers are consistent in attracting citations, even many years after publication.

Another interesting point from these results is that although the existing formulation of journal impact factor[14] is defined in light of the citation profile, as in Figure 1, most journal papers in PeakLate or MonIncr do not follow such a profile at all; at least for papers in PeakLate, the metric does not focus on the most relevant timeframe of the citation profile (mostly the first five years after publication). In light of our results, the appropriateness of the formulation of bibliographic metrics (such as journal impact factor) are doubtful; for example, a journal's impact factor[15] at any given time is the average number of citations received per paper published during the two preceding years.

Effect of self-citation on categorization. Another factor often affecting citation rate is "self-citation,"[12] which can inflate the perception of an article's or a scientist's scientific impact, particularly when an article has many authors,

Table 1. Mean publication year Y (its standard deviation (σ(Y)) and the percentage of papers in conferences and journals for each category of a citation profile.

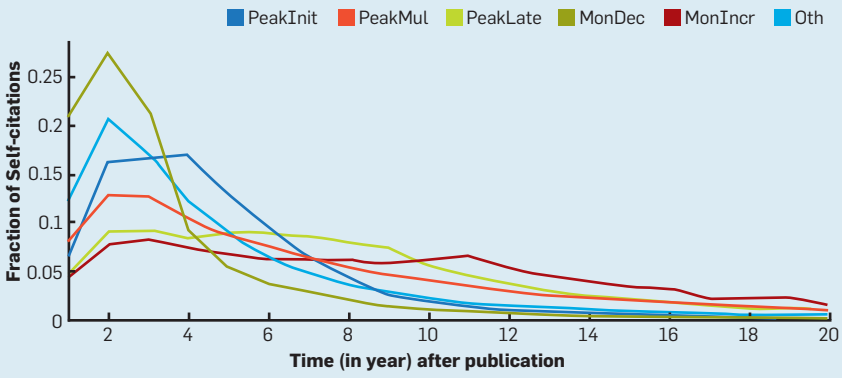
Category	Mean publication year (σ(Y))	% of conference papers	% of journal papers
PeakInit	1994 (5.19)	64.35	35.65
PeakMul	1991 (6.68)	39.03	60.97
PeakLate	1992 (6.54)	39.89	60.11
MonDec	1994 (5.44)	60.73	39.27
MonIncr	1993 (7.36)	25.26	74.74

Table 2. Confusion matrix representing the transition of categories due to the removal of self-citations.

A value *x* in the cell (*i*, *j*) represents *x* fraction of papers in category *i* would have been placed in category *j* if self-citations were absent from the entire dataset. Note no row is specified for the Oth category, as papers in this category never move to other categories through deletion of citations.

Category	PeakInit	PeakMul	PeakLate	MonDec	MonIncr	Oth
PeakInit	0.72	0.10	0.03	0.01	0	0.15
PeakMul	0.02	0.81	0.04	0	0.10	0.11
PeakLate	0.01	0.06	0.86	0	0.01	0.06
MonDec	0.05	0.14	0	0.41	0	0.35
MonIncr	0	0.02	0.01	0.01	0.88	0.09

Figure 6. Fraction of self-citations per paper in different categories over different time periods after publication and fraction of papers in each category migrating to the Oth category due to removal of self-citations, assuming different category thresholds.



increasing the possible number of self-citations;[11,29] there have thus been calls to remove self-citations from citation-rate calculations.[29] We conducted a similar experiment to identify the effect of self-citation on the categorization of citation profiles. We first removed a citation from the dataset if the citing and the cited papers had at least one author in common, then measured the fraction of papers in each category migrating to some other category due to this removal. Table 2 is a confusion matrix, where labels in the rows and the columns represent the categories before and after removing self-citations, respectively.

Note papers in MonDec are strongly affected by the self-citation phenomenon. Around 35% of papers in MonDec would have been in the Oth category if not for self-citations. However, this percentage might be the result of the thresholding we impose, as discussed earlier, when categorizing papers; papers with fewer than or as many as 10 citations in the first 10 years following publication are considered to be in Oth category. Looking to verify the effect of thresholding on inter-category migration following removal of self-citations, we varied the category threshold from 10 to 14 and plotted the fraction of papers in each category migrating to Oth due to our removal of self-citations (see Figure 6). The result agreed with the Table 2 observation that the MonDec category is most affected by self-citations, followed by PeakInit, PeakMul, and PeakLate. This result indicates the effect of self-citations is due to the inherent characteristics of each category, rather than to the predefined threshold setting of the category boundary, following three trends:

Authors. Authors tend to cite their own papers within two to three years of publication to increase visibility;

Conference papers. The MonDec and PeakInit categories, or mostly conference papers, are strongly affected by self-citation; and

Visibility. Self-citation is usually seen soon after publication in an attempt to increase publication visibility.

Figure 6 reflects how self-citations are distributed across different time periods for individual categories; we aggregated all self-citations and plotted the fraction of self-citations following publication. As expected, for the MonDec category we found most self-citations are "farmed" within two to three years of publication. A similar observation holds for both the PeakInit and Oth categories. Note, PeakInit and MonDec are composed mostly of conference papers. We conclude conference papers are the most affected by self-citations. However, the characteristics of the highly cited categories (such as MonIncr and PeakLate) are mostly consistent through the years, showing these categories are less dependent on self-citation.

Figure 7. Alluvial diagram representing evolution of papers in different categories and the flows between categories at time T + 10, T + 15, and T + 20.

The colored blocks correspond to different categories. Block size indicates number of papers in a category, and the shaded waves joining the regions represent flow of papers between the regions, such that the width of the flow corresponds to the fraction of papers. The total width of incoming flows is equal to the width of the corresponding region.

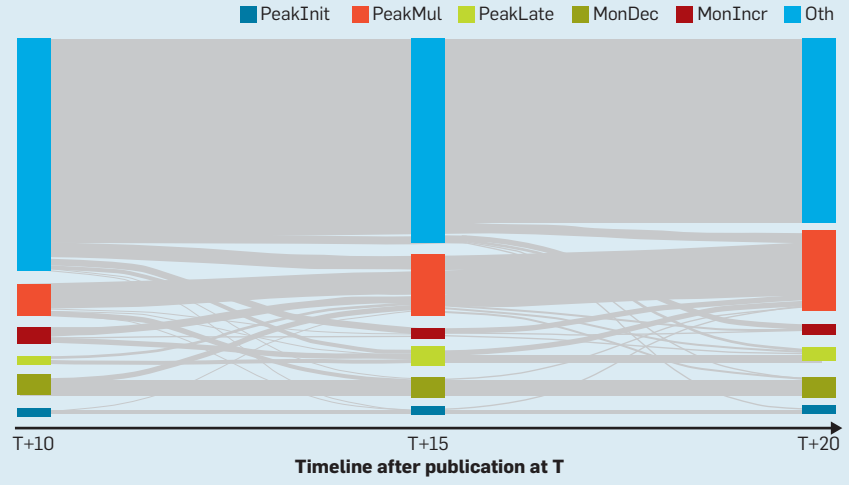
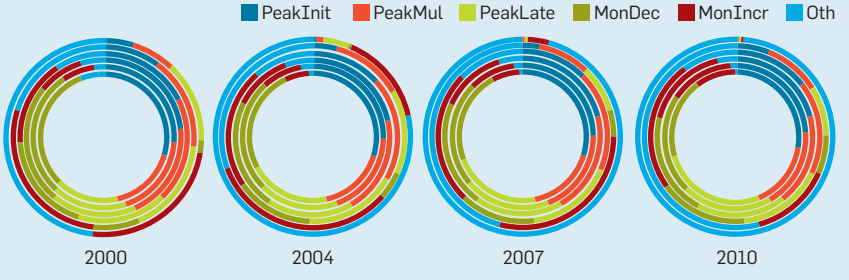


Figure 8. Multi-level pie chart for years 2000, 2004, 2007, and 2010, showing the composition of each category in different k^s-shell regions, where colors represent different categories and the area covered by each colored region in each k^s-shell represents the proportion of papers in the corresponding category in that shell.

The innermost shell is the core region, and the outermost shell is the periphery region. For better visualization of the different regions, we divided the total number of shells identified from the citation network in each year into six broad shells; the core-periphery structure in each year thus has six concentric layers.



Stability of Different Categories

The number of citations for a paper changes over time depending on the paper's effect on the scientific community that might change the shape of the citation profile. Studying the temporal evolution of each citation profile can help researchers understand the stability of the categories individually. Since we know the category of papers with at least 20 years of citation history, we further analyzed how the shape of the profile evolves over those 20 years. Following publication of a paper at time T, we identified its category at time T + 10, T + 15 and T + 20 based on the heu-

ristics discussed earlier. We hypothesize a stable citation category tends to maintain its shape over a paper's entire published timeline. The colored blocks of the alluvial diagram[28] in Figure 7 correspond to the different categories for three different timestamps. We observed that apart from the Oth category, which indeed includes a major proportion of all the papers in our dataset, MonDec seemed the most stable, followed by PeakInit. However, papers we assumed to belong in the Oth category quite often turned out to be MonIncr papers at a later time. This analysis demonstrates a systematic approach to explaining the transition from one category to another with increased numbers of citations.

Since our primary focus was analyzing a paper's citation growth following publication, we needed an in-depth understanding of how citation numbers vary over time.

Core-Periphery Analysis

Although Figure 5 indicates the influence of different categories in terms of raw citation count, it neither explains the significance of the papers in each category forming the core of the network nor gives any information regarding the temporal evolution of the structure. For a better and more detailed understanding of that evolution, we performed k-core analysis[8,21] on the evolving citation network by decomposing the network for each year into its k^s-shells such that an inner shell index of a paper reflects a central position in the core of the network.

We constructed a number of aggregated citation networks in different years—2000, 2004, 2007, and 2010—such that a citation network for year Y included the induced subgraph of all papers published at or before Y. For each such network, we then ran several methods, starting by recursively removing nodes with a single inward link until no such nodes remained in the network. These nodes form the 1-shell of the network, or k^s-shell index $k^s = 1$. Similarly, by recursively removing all nodes with degree 2, we derived the 2-shell. We continued increasing k until all nodes in the network were assigned to one of the shells. The union of all shells with index greater than or equal to k^s is the k^s-core of the network, and the union of all shells with index smaller or equal to k^s is the k^s-crust of the network. The idea is to show how the papers in each category (identified in 2000) migrate from one shell to another after attracting citations over the next 10 years. It also allowed us to observe the persistence of a category in a particular shell.

In Figure 8, most papers in the Oth category are in the periphery and their proportion in the periphery increases over time, indicating they are increasingly less popular over time. The PeakMul category gradually leaves the peripheral region over time and mostly occupies the two innermost shells. PeakInit and MonDec show similar behavior, with the largest proportion of papers in inner cores in the initial year but gradually shifting toward peripheral regions. On the other hand, MonIncr and PeakLate showed the expected behavior, with their proportions increasing in the inner shells over time, indicating rising relevance over time. This helped us identify the temporal evolution of the importance of different categories in terms of how each of them contributes to the central position of a citation network.

Dynamic Growth Model

Extensive research has gone toward developing growth models to explain evolution of citation networks;[19,30] for example, models like those from Barabási-Albert[1,2] and Price[26] attempt to generate scale-free networks through a preferential-attachment mechanism. Most such work seeks to explain the emergence of a network's degree distribution. Here, we propose a novel dynamic growth model to synthesize the citation network, aiming to reproduce the citation categories seen in the Microsoft Academic Search dataset. To the best of our knowledge, this model is the first of its kind to take into account preferential attachment[1] and aging[18,20] to mimic real-world citation profiles.

As input to the model for comparing against the real-world citation profiles, we used the following distributions: number of papers over the years (to determine the number and type of papers entering the system at each time step) and reference distribution (to determine outward citations from an incoming node). At each time step (corresponding to a particular year), we selected a number of nodes (papers) with outdegree (references) for each, as determined preferentially from the refer-

ence distribution. We then assigned the vertex preferentially to a particular category based on the size of the categories (number of papers in each) at that time step. To determine the other end point of each edge associated with the incoming node, we first selected a category preferentially based on the in-citation information of the category, then selected within the category a node (paper) preferentially based on its attractiveness. We determined attractiveness by time elapsed since publication (aging) and number of citations accumulated till that time (preferential attachment). Note the formulation of the attractiveness in our model also varies for different categories.

We found remarkable resemblance between real-world citation profiles and those obtained from the model in Figure 3, bottom panels. Each frame of the figure includes three lines depicting first quartile (10% points below this line), third quartile (10% points above this line), and mean behavior. We also compared the in-degree distributions obtained from the model and from the real dataset for different categories, observing a significant resemblance. Our model thus reflects a holistic view of the evolution of a citation network over time, along with the intra- and inter-category interactions that account for the observable properties of the real-world system.

Conclusion

Access to a massive computer science bibliographic dataset from Microsoft Academic Search made it possible for us to conduct an exhaustive analysis of citation profiles of individual papers and derive six main categories not previously reported in the literature. At the micro-level, we provide a set of new approaches to characterize each individual category, as well as the dynamics of its evolution over time. Leveraging these behavioral signatures, we were able to design a novel dynamic model to synthesize the network evolving over time. The model in turn revealed citation patterns of different categories, showing significant resemblance to what we obtained from the real data.

This article thus offers a first step toward reformulating the existing quantifiers available in scientometrics to leverage different citation patterns and formulate robust measures. Moreover, a systematic machine-learning model of the behavior of different citation patterns has the potential to enhance standard research methodology, including discovering missing links in citation networks,[10] predicting citations of scientific articles,[31] predicting high-impact and seminal papers,[23] and recommending scientific articles.[4]

In future work, we plan to extend our study to the datasets of other domains, possibly physics and biology, to verify the universality of our categorizations. We are also keen to understand the micro-level dynamics controlling the behavior of the PeakMul category, which is significantly different from the other four. One initial observation in this direction is that PeakMul behaves like an intermediary between PeakInit and PeakLate. Also in future work, we would like to understand different characteristic features, particularly for PeakMul. C

References

1. Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of Modern Physics 74*, 1 (Jan. 2002), 47–97.
2. Barabási, A.-L. and Albert, R. Emergence of scaling in random networks. *Science 286*, 5439 (Oct. 1999), 509–512.
3. Bensman, S.J. The evaluation of research by scientometric indicators. *Journal of the Association for Information Science and Technology 62*, 1 (Jan. 2011), 208–210.
4. Bogers, T. and van den Bosch, A. Recommending scientific articles using CiteULike. In *Proceedings of the Second ACM Conference on Recommender Systems* (Lausanne, Switzerland, Oct. 23–25). ACM Press, New York, 2008, 287–290.
5. Boonchom, S., Nuchwana, L., and Amorn, M. The development of standards, factors, and indicators for evaluating the quality of classroom action research. *Procedia - Social and Behavioral Sciences 69* (Dec. 2012), 220–226.
6. Bornmann, L. and Daniel, H.-D. What do citation counts measure? A review of studies on citing behavior. *Journal of Documentation 64*, 1 (Jan.-Feb. 2008), 45–80.
7. Callaham, M., Wears, R.L., and Weber, E. Journal prestige, publication bias, and other characteristics associated with citation of published studies in peer-reviewed journals. *Journal of the American Medical Association 287*, 21 (June 2002), 2847–2850.
8. Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., and Shir, E. A model of Internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences 104*, 27 (July 2007), 11150–11154.
9. Chakraborty, T., Sikdar, S., Tammana, V., Ganguly, N., and Mukherjee, A. Computer science fields as ground-truth communities: Their impact, rise, and fall. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (Niagara Falls, Canada, Aug. 25–28). ACM Press, New York, 2013, 426–433.
10. Clauset, A., Moore, C., and Newman, M.E.J. Hierarchical structure and the prediction of missing links in networks. *Nature 453*, 7191 (May 2008), 98–101.
11. Della Sala, S. and Brooks, J. Multi-authors' self-citation: A further impact factor bias? *Cortex 44*, 9 (Oct. 2008), 1139 –1145.
12. Fowler, J. and Aksnes, D. Does self-citation pay? *Scientometrics 72*, 3 (Sept. 2007), 427–437.
13. Garfield, E. Which medical journals have the greatest impact? *Annals of Internal Medicine 105*, 2 (Aug. 1986), 313–20.
14. Garfield, E. The impact factor and using it correctly. *Der Unfallchirurg 6*, 101 (June 1998), 413–414.
15. Garfield, E. Journal impact factor: A brief review. *Canadian Medical Association Journal 161*, 8 (Oct. 1999), 979–980.
16. Garfield, E. Impact factors, and why they won't go away. *Nature 411*, 6837 (May 2001), 522.
17. Garfield, E. The history and meaning of the journal impact factor. *Journal of the American Medical Association 295*, 1 (Jan. 2006), 90–93.
18. Gingras, Y., Larivier, V., Macaluso, B., and Robitaille, J.-P. The effects of aging on researchers' publication and citation patterns. *PLoS ONE 3*, 12 (Dec. 2008), e4048.
19. Golosovsky, M. and Solomon, S. Stochastic dynamical model of a growing citation network based on a self-exciting point process. *Physical Review Letters 109*, 098701 (Aug. 2012).
20. Hajra, K.B. and Sen, P. Aging in citation networks. *Physica A 346*, 1 (Feb. 2005), 44–48.
21. Harris, J., Hirst, J., and Mossinghoff, M. *Combinatorics and Graph Theory. Springer Undergraduate Texts in Mathematics and Technology.* Springer-Verlag, New York, 2008.
22. Lariviere, V., Ni, C., Gingras, Y., Cronin, B., and Sugimoto, C.R. Bibliometrics: Global gender disparities in science. *Nature 504*, 7479 (Dec. 2013), 211–213.
23. McNamara, D., Wong, P., Christen, P., and Ng, K.S. Predicting high impact academic papers using citation network features. In *Proceedings of the Pacific-Asia Conference in Knowledge Discovery and Data Mining Workshop* (Gold Coast, Australia, Apr. 14–17). Springer, Berlin, Heidelberg, 2013, 14–25.
24. Meho, L.I. The rise and rise of citation analysis. *Physics World 1*, 20 (Jan. 2007), 32–36.
25. Peritz, B.C. On the objectives of citation analysis: Problems of theory and method. *Journal of the Association for Information Science and Technology 43*, 6 (July 1992), 448–451.
26. Price, D.D.S. A general theory of bibliometric and other cumulative advantage processes. *Journal of the Association for Information Science and Technology* (Sept. 1976), 292–306.
27. Radicchi, F. and Fortunato, S., Castellano, C. Universality of citation distributions: Towards an objective measure of scientific impact. *Proceedings of the National Academy of Sciences 105*, 45 (Nov. 2008), 17268–17272.
28. Rosvall, M. and Bergstrom, C.T. Mapping change in large networks. *PLoS ONE 5*, e8694 (Jan. 2010).
29. Schreiber, M. Self-citation corrections for the Hirsch index. *Europhysics Letters 78*, 3 (May 2007), 1–6.
30. Sen, P. Directed accelerated growth: Application in citation network. *Physica A 346*, 1–2 (Feb. 2005), 139–146.
31. Yan, R., Huang, C., Tang, J., Zhang, Y., and Li, X. To better stand on the shoulder of giants. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries* (Washington, D.C., June 10–14). ACM Press, New York, 2012, 51–60.

Tanmoy Chakraborty (its_tanmoy@cse.iitkgp.ernet.in) is a Google India Ph.D. fellow in the Department of Computer Science and Engineering at the Indian Institute of Technology, Kharagpur, India.

Suhansanu Kumar (suhansanu.kumar@cse.iitkgp.ernet.in) is a Ph.D. student in the Department of Computer Science at the University of Illinois, Urbana-Champaign, IL.

Pawan Goyal (pawang@cse.iitkgp.ernet.in) is an assistant professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Kharagpur, India.

Niloy Ganguly (niloy@cse.iitkgp.ernet.in) is a professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Kharagpur, India.

Animesh Mukherjee (animeshm@cse.iitkgp.ernet.in) is an assistant professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Kharagpur, India, and a Simons Associate in the Abdus Salam International Centre for Theoretical Physics, Trieste, Italy.

Call for Nominations

The ACM Doctoral Dissertation Competition

Rules of the Competition

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

Submissions

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

Eligibility

Please see our website for exact eligibility rules. Only English language versions will be accepted. Please send a copy of the thesis in PDF format to emily.eng@hq.acm.org.

Sponsorship

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

Deadline

Submissions must be received by **October 31, 2015** to qualify for consideration.

Publication Rights

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

Publication

Winning dissertations will be published by ACM in the ACM Books Program and appear in the ACM Digital Library. Honorable mention dissertations will appear in the ACM Digital Library

Selection Procedure

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

The selection committee will select the winning dissertation in early 2016.

Award

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

For Submission Procedure

http://awards.acm.org/doctoral_dissertation/

Association for Computing Machinery

DOI:10.1145/2701413

AI has seen great advances of many kinds recently, but there is one critical area where progress has been extremely slow: ordinary commonsense.

BY ERNEST DAVIS AND GARY MARCUS

Commonsense Reasoning and Commonsense Knowledge in Artificial Intelligence

WHO IS TALLER, Prince William or his baby son Prince George? Can you make a salad out of a polyester shirt? If you stick a pin into a carrot, does it make a hole in the carrot or in the pin? These types of questions may seem silly, but many intelligent tasks, such as understanding texts, computer vision, planning, and scientific reasoning require the same kinds of real-world knowledge and reasoning abilities. For instance, if you see a six-foot-tall person holding a two-foot-tall person in his arms, and you are told they are father and son, you do not have to ask which is which. If you need to make a salad for dinner and are out of lettuce, you do not waste time considering improvising by taking a shirt of the closet and cutting

» key insights

- To achieve human-level performance in domains such as natural language processing, vision, and robotics, basic knowledge of the commonsense world—time, space, physical interactions, people, and so on—will be necessary.
- Although a few forms of commonsense reasoning, such as taxonomic reasoning and temporal reasoning are well understood, progress has been slow.
- Extant techniques for implementing commonsense include logical analysis, handcrafting large knowledge bases, Web mining, and crowdsourcing. Each of these is valuable, but none by itself is a full solution.
- Intelligent machines need not replicate human cognition directly, but a better understanding of human commonsense might be a good place to start.

ILLUSTRATION BY PETER CROWTHER ASSOCIATES

it up. If you read the text, "I stuck a pin in a carrot; when I pulled the pin out, it had a hole," you need not consider the possibility "it" refers to the pin.

To take another example, consider what happens when we watch a movie, putting together information about the motivations of fictional characters we have met only moments before. Anyone who has seen the unforgettable horse's head scene in *The Godfather* immediately realizes what is going on. It is not just it is unusual to see a severed horse head, it is clear Tom Hagen is sending Jack Woltz a message—if I can decapitate your horse, I can decapitate you; cooperate, or else. For now, such inferences lie far beyond anything in artificial intelligence.

In this article, we argue that commonsense reasoning is important in many AI tasks, from text understanding to computer vision, planning and reasoning, and discuss four specific problems where substantial progress has been made. We consider why the problem in its general form is so difficult and why progress has been so slow, and survey various techniques that have been attempted.

Commonsense in Intelligent Tasks

The importance of real-world knowledge for **natural language processing**, and in particular for disambiguation of all kinds, was discussed as early as 1960, by Bar-Hillel,[3] in the context of machine translation. Although some ambiguities can be resolved using simple rules that are comparatively easy to acquire, a substantial fraction can only be resolved using a rich understanding of the world. A well-known example from Terry Winograd[48] is the pair of sentences "The city council refused the demonstrators a permit because they feared violence," vs."... because they advocated violence." To determine that "they" in the first sentence refers to the council if the verb is "feared," but refers to the demonstrators if the verb is "advocated" demands knowledge about the characteristic relations of city councils and demonstrators to violence; no purely linguistic clue suffices.[a]

a Such pairs of sentences are known as "Winograd schemas" after this example; a collection of many such examples can be found at http://cs.nyu.edu/faculty/davise/papers/WS.html.[32]

Machine translation likewise often involves problems of ambiguity that can only be resolved by achieving an actual understanding of the text—and bringing real-world knowledge to bear. Google Translate often does a fine job of resolving ambiguities by using nearby words; for instance, in translating the two sentences "The electrician is working" and "The telephone is working" into German, it correctly translates "working" as meaning "laboring," in the first sentence and as meaning "functioning correctly" in the second, because in the corpus of texts Google has seen, the German words for "electrician" and "laboring" are often found close together, as are the German words for "telephone" and "function correctly."[b] However, if you give it the sentences "The electrician who came to fix the telephone is working," and "The telephone on the desk is working," interspersing several words between the critical element (for example, between electrician and working), the translations of the longer sentences say the electrician is functioning properly and the telephone is laboring (Table 1). A statistical proxy for commonsense that worked in the simple case fails in the more complex case.

Almost without exception, current computer programs to carry out language tasks succeed to the extent the tasks can be carried out purely in terms of manipulating individual words or short phrases, without attempting any deeper understanding; commonsense is evaded, in order to focus on short-term results, but it is difficult to see how human-level understanding can be achieved without greater attention to commonsense.

Watson, the "Jeopardy"-playing program, is an exception to the above rule only to a small degree. As described in Kalyanpur,[27] commonsense knowledge and reasoning, particularly taxonomic reasoning, geographic reasoning, and temporal reasoning, played some role in Watson's operations but only a quite limited one, and they made only a small contribution to Watson's success. The key techniques in Watson are mostly of the same flavor as those used in programs like Web search engines: there is a large collection of extremely sophisticated and highly tuned rules for matching words and phrases in the question with snippets of Web documents such as Wikipedia; for reformulating the snippets as an answer in proper form; and for evaluating the quality of proposed possible answers. There is no evidence that Watson is anything like a general-purpose solution to the commonsense problem.

Computer vision. Similar issues arise in computer vision. Consider the photograph of Julia Child's kitchen (Figure 1): Many of the objects that are small or partially seen, such as the metal bowls in the shelf on the left, the cold water knob for the faucet, the round metal knobs on the cabinets, the dishwasher, and the chairs at the table seen from the side, are only recognizable in context; the isolated image would be difficult to identify. The top of the chair on the far side of the table is only identifiable because it matches the partial view of the chair on the near side of the table.

The viewer infers the existence of objects that are not in the image at all. There is a table under the yellow tablecloth. The scissors and other items hanging on the board in the back are presumably supported by pegs or hooks. There is presumably also a hot water knob for the faucet occluded by the dish rack. The viewer also infers how the objects can be used (sometimes called their "affordances"); for example, the cabinets and shelves can be opened by pulling on the handles. (Cabinets, which rotate on joints, have the handle on one side; shelves, which pull out straight, have the handle in the center.)

Movies would prove even more difficult; few AI programs have even tried. *The Godfather* scene mentioned earlier is one example, but almost any movie contains dozens or hundreds of moments that cannot be understood simply by matching still images to memorized templates. Understanding a movie requires a viewer to make numerous inferences about the intentions of characters, the nature of physical objects, and so forth. In the current state of the art, it is not feasible even to attempt to build a program that will be able to do this reasoning; the most that can be done is to track characters and identify basic actions like standing up, sitting down, and opening a door.[4]

Robotic manipulation. The need for commonsense reasoning in autonomous robots working in an uncontrolled environment is self-evident, most conspicuously in the need to have the robot react to unanticipated events appropriately. If a guest asks a waiter-robot for a glass of wine at a party, and the robot sees the glass he is picked up is cracked, or has a dead cockroach at the bottom, the robot should not simply pour the wine into the glass and serve it. If a cat runs in front of a house-cleaning robot, the robot should neither run it over nor sweep it up nor put it away on a shelf. These things seem obvious, but ensuring a robot avoids mistakes of this kind is very challenging.

b Google Translate is a moving target; this particular example was carried out on 6/9/2015, but translations of individual sentences change rapidly—not always for the better on individual sentences. Indeed, the same query given minutes apart can give different results. Changing the target language, or making seemingly inconsequential changes to the sentence, can also change how a given ambiguity is resolved, for no discernible reason. Our broader point here is not to dissect Google Translate per se, but to note it is unrealistic to expect fully reliable disambiguation in the absence of a deep understanding of the text and relevant domain knowledge.

Table 1. Lexical ambiguity and Google Translate. We have highlighted the translation of the word "working." The German word "arbeitet" means "labors;" "funktioniert" means "functions correctly."

English original	Google translation
The electrician is working.	Der Electriker **arbeitet**.
The electrician that came to fix the telephone is working.	Der Elektriker, die auf das Telefon zu beheben kam **funktioniert**.
The telephone is working.	Das Telefon **funktioniert**.
The telephone on the desk is working.	Das Telefon auf dem Schreibtisch **arbeitet**.

Successes in Automated Commonsense Reasoning

Substantial progress in automated commonsense reasoning has been made in four areas: reasoning about

taxonomic categories, reasoning about time, reasoning about actions and change, and the sign calculus. In each of these areas there exists a well-understood theory that can account for some broad range of commonsense inferences.

Taxonomic reasoning. A taxonomy is a collection of categories and individuals, and the relations between them. (Taxonomies are also known as semantic networks.) For instance, Figure 2 shows a taxonomy of a few categories of animals and individuals.

There are three basic relations:

- An individual is an instance of a category. For instance, the individual `Lassie` is an instance of the category `Dog`.
- One category is a subset of another. For instance `Dog` is a subset of `Mammal`.
- Two categories are disjoint. For instance `Dog` is disjoint from `Cat`.

Figure 2 does not indicate the disjointness relations.

Categories can also be tagged with properties. For instance, `Mammal` is tagged as `Furry`.

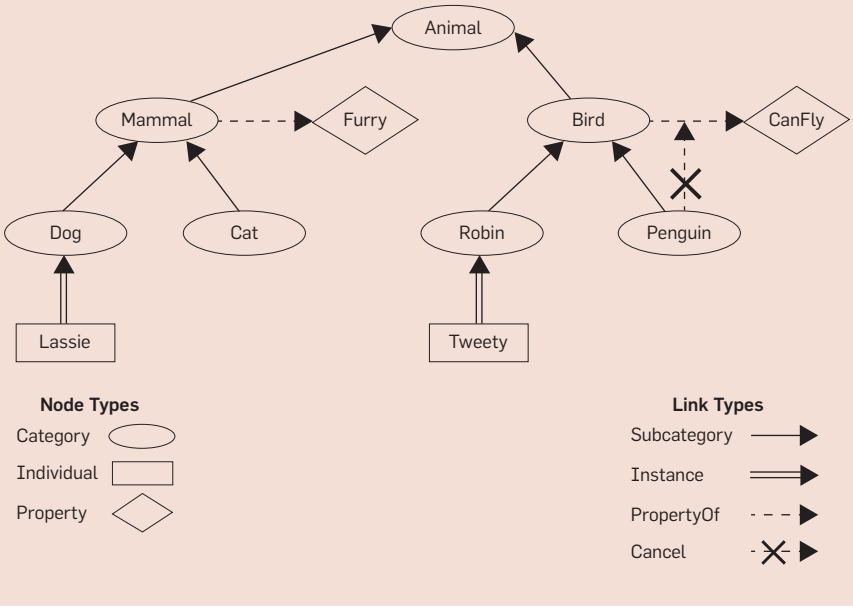
One form of inference in a taxonomy is transitivity. Since `Lassie` is an instance of `Dog` and `Dog` is a subset of `Mammal`, it follows that `Lassie` is an instance of `Mammal`. Another form of inference is inheritance. Since `Lassie` is an instance of `Dog`, which is a subset of `Mammal` and `Mammal` is marked with property `Furry`, it follows that `Dog` and `Lassie` have property `Furry`. A variant of this is default inheritance; a category can be marked with a characteristic but not universal property, and a subcategory or instance will inherit the property unless it is specifically canceled. For instance, `Bird` has the default property `CanFly`, which is inherited by `Robin` but not by `Penguin`.

The standard taxonomy of the animal kingdom is particularly simple in structure. The categories are generally sharply demarcated. The taxonomy is tree-structured, meaning given any two categories, either they are disjoint or one is a subcategory of the other. Other taxonomies are less straightforward. For instance, in a semantic network for categories of people, the individual `GalileoGalilei` is simultaneously a `Physicist`, an `Astronomer`, a `ProfessorOfMathematics`, a `WriterInItalian`, a `NativeOfPisa`, a `PersonChargedWithHeresy`, and so on. These overlap, and it is not clear which of these are best viewed as taxonomic categories and which are better viewed as properties. In taxonomizing more abstract categories, choosing and delimiting categories becomes more problematic; for instance, in constructing a taxonomy for a theory of narrative, the membership, relations, and definitions of categories like `Event`, `Action`, `Process`, `Development`, and `Incident` are uncertain.

Simple taxonomic structures such as those illustrated here are often used in AI programs. For example, WordNet[34] is a widely used resource that includes a taxonomy whose elements are meanings of English words. As we will discuss later, Web mining systems that collect commonsense knowledge from Web documents tend to be largely focused on taxonomic relations, and more suc-

Figure 1. Julia Child's kitchen. Photograph by Matthew Bisanz.

Figure 2. Taxonomy.



Understanding a Classic Scene from *The Godfather*

Hagen foresaw, while he was planning the operation, that
Woltz would realize that
Hagen arranged for the placing of the head in Woltz's bed
in order to make
Woltz realize that
Hagen could easily arrange to have him killed
if he does not accede to Hagen's demands.

cessful in gathering taxonomic relations than in gathering other kinds of knowledge. Many specialized taxonomies have been developed in domains such as medicine[40] and genomics.[21] More broadly, the Semantic Web enterprise is largely aimed at developing architectures for large-scale taxonomies for Web applications.

A number of sophisticated extensions of the basic inheritance architecture described here have also been developed. Perhaps the most powerful and widely used of these is description logic.[2] Description logics provide tractable constructs for describing concepts and the relations between concepts, grounded in a well-defined logical formalism. They have been applied extensively in practice, most notably in the semantic Web ontology language OWL.

Temporal reasoning. Representing knowledge and automating reasoning about times, durations, and time intervals is a largely solved problem.[17] For instance, if one knows that Mozart was born earlier and died younger than Beethoven, one can infer that Mozart died earlier than Beethoven. If one knows the Battle of Trenton occurred during the Revolutionary War, the Battle of Gettysburg occurred during the Civil War, and the Revolutionary War was over before the Civil War started, then one can infer the Battle of Trenton occurred before the Battle of Gettysburg. The inferences involved here in almost all cases reduce to solving systems of linear inequalities, usually small and of a very simple form.

Integrating such reasoning with specific applications, such as natural language interpretation, has been much more problematic. Natural language expressions for time are complex and their interpretation is context dependent. Temporal reasoning was used to some extent in the Watson "Jeopardy!"-playing program to exclude answers that would be a mismatch in terms of date.[27] However, many important temporal relations are not explicitly stated in texts, they are inferred; and the process of inference can be difficult. Basic tasks like assigning timestamps to events in news stories cannot be currently done with any high degree of accuracy.[47]

Action and change. Another area of commonsense reasoning that is well understood is the theory of action, events, and change. In particular, there are very well established representational and reasoning techniques for domains that satisfy the following constraints:[42]

- *Events are atomic.* That is, one event occurs at a time, and the reasoner need only consider the state of the world at the beginning and the end of the event, not the intermediate states while the event is in progress.
- Every change in the world is the result of an event.
- Events are deterministic; that is, the state of the world at the end of the event is fully determined by the state of the world at the beginning plus the specification of the event.
- *Single actor.* There is only a single actor, and the only events are either his actions or exogenous events in the external environment.
- *Perfect knowledge.* The entire relevant state of the world at the start, and all exogenous events are known or can be calculated.

For domains that satisfy these constraints, the problem of representation and important forms of reasoning, such as prediction and planning, are largely understood. Moreover, a great deal is known about extensions to these domains, including:

- Continuous domains, where change is continuous.
- Simultaneous events.
- Probabilistic events, whose outcome depends partly on chance.
- Multiple agent domains, where agents may be cooperative, independent, or antagonistic.
- Imperfect knowledge domains, where actions can be carried out with the purpose of gathering information, and (in the multiagent case) where cooperative agents must communicate information.
- Decision theory: Comparing different courses of action in terms of the expected utility.

The primary successful applications of these kinds of theories has been to high-level planning,[42] and to some extent to robotic planning, for example, Ferrein et al.[16]

The situation calculus uses a branching model of time, because it was primarily developed to characterize planning, in which one must consider alternative possible actions. However, it does not work well for narrative interpretation, since it treats events as atomic and requires the order of events be known. For narrative interpretation, the event calculus[37] is more suitable. The event calculus can express many of the temporal relations that arise in narratives; however, only limited success has been obtained so far in applying it in the interpretation of natural language texts. Moreover, since it uses a linear model of time, it is not suitable for planning.

Many important issues remain unsolved, however, such as the problem of integrating action descriptions at different levels of abstraction. The process of cooking dinner, for instance, may involve such actions as "Getting to know my significant other's parents," "Cooking dinner for four," "Cooking pasta primavera," "Chopping a zucchini," "Cutting once through the zucchini", and "With the right hand, grasping the knife by the handle, blade downward, and lowering it at about one foot per second through the center of the zucchini, while, with the left hand, grasping the zucchini and holding it against the cutting board."

Reasoning about how these different kinds of actions interrelate—for example, if you manage to slice off a finger while cutting the zucchini, your prospective parents-in-law may not be impressed—is substantially unsolved.

Qualitative reasoning. One type of commonsense reasoning that has been analyzed with particular success is known as qualitative reasoning. In its simplest form, qualitative reasoning is about the direction of change in interrelated quantities. If the price of an object goes up then (usually, other things being equal) the number sold will go down. If the temperature of gas in a closed container goes up, then the pressure will go up. If an ecosystem contains foxes and rabbits and the number of foxes decreases, then the death rate of the rabbits will decrease (in the short term).

An early version of this theory was formulated by Johan de Kleer[11] for analyzing an object moving on a roller coaster. Later more sophisticated forms were developed in parallel by de Kleer and Brown[12] for analyzing electronic circuits; by Forbus[18] for analyzing varieties of physical processes; and by Kuipers[30] as a mathematical formalism.

This theory has been applied in many domains, from physics to engineering, biology, ecology, and engineering. It has also served as the basis for a number of practical programs, including text understanding;[29] analogical mapping and geometric reasoning;[33] failure analysis in automotive electrical systems;[41] and generating control strategies for printing.[20]

For problems within the scope of the representation, the reasoning mechanism works well. However, there are many problems in physical reasoning, particularly those involving substantial geometric reasoning, that cannot be represented in this way, and therefore lie outside the scope of this reasoning mechanism. For example, you want to be able to reason a basketball will roll smoothly in any direction, whereas a football can roll smoothly if its long axis is horizontal but cannot roll smoothly end-over-end. This involves reasoning about the interactions of all three spatial dimensions together.

Current computer programs to carry out language tasks succeed to the extent the tasks can be carried out purely in terms of manipulating individual words or short phrases, without attempting any deeper understanding.

Challenges in Automating Commonsense Reasoning

As of 2014, few commercial systems make any significant use of automated commonsense reasoning. Systems like Google Translate use statistical information culled from large datasets as a sort of distant proxy for commonsense knowledge, but beyond that sort of crude proxy, commonsense reasoning is largely absent. In large part, that is because nobody has yet come close to producing a satisfactory commonsense reasoner. There are five major obstacles.

First, many of the domains involved in commonsense reasoning are only partially understood or virtually untouched. We are far from a complete understanding of domains such as physical processes, knowledge and communication, plans and goals, and interpersonal interactions. In domains such as the commonsense understanding of biology, of social institutions, or of other aspects of folk psychology, little work of any kind has been done.

Second, situations that seem straightforward can turn out, on examination, to have considerable logical complexity. For example, consider the horse's head scene in *The Godfather*. The box on the previous page illustrates the viewer's understanding of the scene. We have a statement with embeddings of three mental states ("foresaw," "realize," "realize"), a teleological connection ("in order"), two hypotheticals ("could arrange" and "does not accede") and a highly complex temporal/causal structure.

Some aspects of these kinds of relations have been extensively studied and are well understood. However, there are many aspects of these relations where we do not know, even in principle, how they can be represented in a form usable by computers or how to characterize correct reasoning about them. For example, there are theories of knowledge that do a good job of representing what different players know about the deal in a poker game, and what each player knows about what the other players know, because one can reasonably idealize all the players as being able to completely think through the situation. However, if you want to model a teacher thinking about what his students do not understand, and how they can be made to understand, then

that is a much more difficult problem, and one for which we currently do not have a workable solution. Moreover, even when the problems of representation and inference have been solved in principle, the problem of carrying out reasoning efficiently remains.

Third, commonsense reasoning almost always involves plausible reasoning; that is, coming to conclusions that are reasonable given what is known, but not guaranteed to be correct. Plausible reasoning has been extensively studied for many years,[23] and many theories have been developed, including probabilistic reasoning,[38] belief revision,[39] and default reasoning or non-monotonic logic.[5] However, overall we do not seem to be very close to a comprehensive solution. Plausible reasoning takes many different forms, including using unreliable data; using rules whose conclusions are likely but not certain; default assumptions; assuming one's information is complete; reasoning from missing information; reasoning from similar cases; reasoning from typical cases; and others. How to do all these forms of reasoning acceptably well in all commonsense situations and how to integrate these different kinds of reasoning are very much unsolved problems.

Fourth, in many domains, a small number of examples are highly frequent, while there is a "long tail" of a vast number of highly infrequent examples. In natural language text, for example, some trigrams (for example, "of the year") are very frequent, but many other possible trigrams, such as "moldy blueberry soda" or "gymnasts writing novels" are immediately understandable, yet vanishingly rare.[c] Long tail phenomena also appear in many other corpora, such as labeled sets of images.[43]

The effect of long-tail distributions on AI research can be pernicious. On the one hand, promising preliminary results for a given task can be gotten easily, because a comparatively small number of common categories include most of the instances. On the other hand, it is often very difficult to attain high quality results, because a significant fraction of the problems that arise correspond to very infrequent categories. The result is the pattern of progress often seen in AI: Rapid progress at the start of research up to a mediocre level, followed by slower and slower improvement. (Of course, for any given application, partial success may be acceptable or indeed extremely valuable; and high quality performance may be unnecessary.)

There is no evidence that IBM's Watson is anything like a general-purpose solution to the commonsense problem.

We conjecture that long-tail phenomena are pervasive in commonsense reasoning, both in terms of the frequency with which a fact appears in knowledge sources (for example, texts) and in terms of the frequency with which it is needed for a reasoning task. For instance, as discussed, a robot waiter needs to realize it should not serve a drink in a glass with a dead cockroach; but how often is that mentioned in any text, and how often will the robot need to know that fact?[d]

Fifth, in formulating knowledge it is often difficult to discern the proper level of abstraction. Recall the example of sticking a pin into a carrot and the task of reasoning that this action may well create a hole in the carrot but not create a hole in the pin. Before it encounters this particular example, an automated reasoner presumably would not specifically know a fact specific to pins and carrots; at best it might know a more general rule[e] or theory about creating holes by sticking sharp objects into other objects. The question is, how broadly should such rules should be formulated? Should such roles cover driving nails into wood, driving staples into paper, driving a spade into the ground, push-

c Google reports no instances of either of these quoted phrases as of June 9, 2015. These are not difficult to find in natural text; for example, one recent book review we examined contained at least eight trigrams (not containing proper nouns) with zero Google hits other than the article itself. A systematic study of n-gram distribution can be found in Allison et al.[1]

d Presumably an intelligent robot would not necessarily know that specific fact in advance at all, but rather would infer it when necessary. However, accomplishing that involves describing the knowledge that supports the inference and building the powerful inference engine that carries out the inference.

e Positing that the reasoner is not using rules at all, but instead is using an instance-based theory does not eliminate the problem. Rather, it changes the problem to the question of how to formulate the features to be used for comparison.

ing your finger through a knitted mitten, or putting a pin into water (which creates a hole that is immediately filled in)? Or must there be individual rules for each domain? Nobody has yet presented a general solution to this problem.

A final reason for the slow progress in automating commonsense knowledge is both methodological[10] and sociological. Piecemeal commonsense knowledge (for example, specific facts) is relatively easy to acquire, but often of little use, because of the long-tail phenomenon discussed previously. Consequently, there may not be much value in being able to do a little commonsense reasoning. The payoff in a complete commonsense reasoner would be large, especially in a domain like robotics, but that payoff may only be realized once a large fraction of the project has been completed. By contrast, the natural incentives in software development favor projects where there are payoffs at every stage; projects that require huge initial investments are much less appealing.

Approaches and Techniques

As with most areas of AI, the study of commonsense reasoning is largely divided into knowledge-based approaches and approaches based on machine learning over large data corpora (almost always text corpora) with only limited interaction between the two kinds of approaches. There are also crowdsourcing approaches, which attempt to construct a knowledge base by somehow combining the collective knowledge and participation of many non-expert people. Knowledge-based approaches can in turn be divided into approaches based on mathematical logic or some other mathematical formalism; informal approaches, antipathetic to mathematical formalism, and sometimes based on theories from cognitive psychology; and large-scale approaches, which may be more or less mathematical or informal, but in any case are chiefly targeted at collecting a lot of knowledge (Figure 3). A particularly successful form of mathematically grounded commonsense reasoning is qualitative reasoning, described previously. We consider these in turn.

Research in commonsense reasoning addresses a number of different objectives:

- *Reasoning architecture.* The development of general-purpose data structures for encoding knowledge and algorithms and techniques for carrying out reasoning. (A closely related issue is the representation of the meaning of natural language sentences.[45])
- *Plausible inference*; drawing provisional or uncertain conclusions.
- *Range of reasoning modes.* Incorporating a variety of different modes of inference, such as explanation, generalization, abstraction, analogy, and simulation.
- *Painstaking analysis of fundamental domains.* In doing commonsense reasoning, people are able to do complex reasoning about basic domains such as time, space, naïve physics, and naïve psychology. The knowledge they are drawing on is largely unverbalized and the reasoning processes largely unavailable to introspection. An automated reasoner will have to have comparable abilities.
- *Breadth.* Attaining powerful commonsense reasoning will require a large body of knowledge.
- *Independence of experts.* Paying experts to hand-code a large knowledge base is slow and expensive. Assembling the knowledge base either automatically or by drawing on the knowledge of non-experts is much more efficient.
- *Applications.* To be useful, the commonsense reasoner must serve the needs of applications and must interface with them smoothly.
- *Cognitive modeling.* Theories of commonsense automated reasoning accurately describe commonsense reasoning in people.

The different approaches to automating commonsense reasoning have

Figure 3. Taxonomy of approaches to commonsense reasoning.

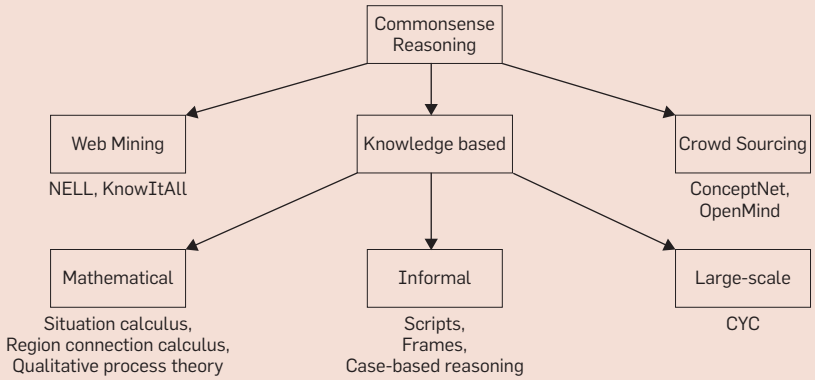


Table 2. Approaches and typical objectives.

	Math-based	Informal	Large-scale	Web mining	Crowd sourcing
Architecture	Substantial	Little	Substantial	Moderate	Little
Plausible reasoning	Substantial	Moderate	Substantial	Little	Little
Range of reasoning modes	Moderate	Substantial	Moderate	Little	Little
Painstaking fundamentals	Substantial	Little	Moderate	Little	Little
Breadth	Little	Moderate	Substantial	Substantial	Substantial
Independence of experts	Little	Little	Little	Substantial	Substantial
Concern with applications	Moderate	Substantial	Substantial	Moderate	Moderate
Cognitive modeling	Little	Substantial	Little	Little	Moderate

often emphasized different objectives, as sketched in Table 2.

Knowledge-based approaches. In knowledge-based approaches, experts carefully analyze the characteristics of the inferences needed to do reasoning in a particular domain or for a particular task and the knowledge those inferences depend on. They handcraft representations that are adequate to express this knowledge and inference engines that are capable of carrying out the reasoning.

Mathematically grounded approaches. Of the four successes of commonsense reasoning enumerated in this article, all but taxonomic reasoning largely derive from theories that are grounded in mathematics or mathematical logic. (Taxonomic representations are too ubiquitous to be associated with any single approach.) However, many directions of work within this approach are marked by a large body of theory and a disappointing paucity of practical applications or even of convincing potential applications. The work on qualitative spatial reasoning[6] illustrates this tendency vividly. There has been active work in this area for more than 20 years, and more than 1,000 research papers have been published, but very little of this connects to any commonsense reasoning problem that might ever arise. Similar gaps between theory and practice arise in other domains as well. The "Muddy Children" problem[f] (also known as the "Cheating Husbands" problem), a well-known brain teaser in the theory of knowledge, has been analyzed in a dozen different variants in a half-dozen different epistemic logics; but we do not know how to represent the complex interpersonal interactions between Hagen and Woltz in the horse's head scene, let alone how to automate reasoning about them.

Unlike the other approaches to commonsense reasoning, much of the work in this approach is purely theoretical; the end result is a published paper rather than an implemented program. Theoretical work of this kind is evaluated, either in terms of meta-theorems (for example, soundness, completeness, computational complexity), or in terms of interesting examples of commonsense inferences the theory supports. These criteria are often technically demanding; however, their relation to the advancement of the state of the art is almost always indirect, and all too often nonexistent.

Overall, the work is also limited in terms of the scope of domains and reasoning techniques that have been considered. Again and again, research in this paradigm has fixated on a small number of examples and forms of knowledge, and generated vast collections of papers dealing with these, leaving all other issues neglected.

Informal knowledge-based approaches. In the informal knowledge-based approach, theories of representation and reasoning are based substantially on intuition and anecdotal data, and to a significant but substantially lesser extent on results from empirical behavioral psychology.

The most important contribution of the informal approach has been the analysis of a broad class of types of inference. For instance, Minsky's frame paper[35] discusses an important form of inference, in which a particular complex individual, such as a particular house, is matched against a known structure, such as the known characteristics of houses in general. Schank's theory of scripts[44] addresses this in the important special case of structured collections of events. Likewise, reasoning by analogy[22,26] and case-based reasoning[28] have been much more extensively studied in informal frameworks than in mathematically grounded frameworks.

It should be observed that in computer programming, informal approaches are very common. Many large and successful programs—text editors, operating systems shells, and so on—are not based on any overarching mathematical or statistical model; they are written ad hoc by the seat of the pants. This is certainly not an inherently implausible approach to AI. The major hazard of work in this approach is that theories can become very nebulous, and that research can devolve into little more than the collection of striking anecdotes and the construction of demonstration programs that work on a handful of examples.

Large-scale approaches. There have been a number of attempts to construct very large knowledge bases of commonsense knowledge by hand. The largest of these is the CYC program. This was initiated in 1984 by Doug Lenat, who has led the project throughout its existence. Its initial proposed methodology was to encode the knowledge in 400 sample articles in a one-volume desk encyclopedia together with all the implicit background knowledge a reader would need to understand the articles (hence, the name).[31] It was initially planned as a 10-year project, but continues to this day. In the last decade, Cycorp has released steadily increasing portions of the knowledge base for public or research use. The most recent public version, OpenCyc 4.0, released in June 2012 contains 239,000 concepts and 2,039,000 facts,

f Alice, Bob, and Carol are playing together. Dad says to them, "At least one of you has mud on your forehead. Alice, is your forehead muddy?" Alice answers, "I don't know." Dad asks, "Bob, is your forehead muddy?" Bob answers, "I don't know." Carol then says, "My forehead is muddy." Explain.

Table 3. Facts recently learned by NELL (6/11/2015).

Fact	Confidence
federica_fontana is a director	91.5
illustrations_of_swollen_lymph_nodes is a lymph node	90.3
lake_triangle is a lake	100.0
louis_pasteur_and_robert_koch is a scientist	99.6
Illinois_governor_george_ryan is a politician	99.8
stephen is a person who moved to the state california	100.0
louis_armstrong is a musician who plays the trumpet	99.6
cbs_early_show is a company in the economic sector of news	93.8
knxv is a TV station in the city phoenix	100.0
broncos is a sports team that plays against new_york_jets	100.0

mostly taxonomic. ResearchCyc, which is available to be licensed for research purposes, contains 500,000 concepts and 5,000,000 facts.

A number of successful applications of CYC to AI tasks have been reported,[8] including Web query expansion and refinement,[7] question answering,[9] and intelligence analysis.[19]

CYC is often mentioned as a success of the knowledge-based approach to AI; for instance Dennett[13] writes, "CYC is certainly the most impressive AI implementation of something like a language of thought." However, it is in fact very difficult for an outsider to determine what has been accomplished here. In its first 15 years, CYC published astonishingly little. Since about 2002, somewhat more has been published, but still very little, considering the size of the project. No systematic evaluation of the contents, capacities, and limitations of CYC has been published.[g]

It is not, for example, at all clear what fraction of CYC actually deals with commonsense inference, and what fraction deals with specialized applications such as medical records or terrorism. It is even less clear what fraction of commonsense knowledge of any kind is in CYC. The objective of representing 400 encyclopedia articles seems to have been silently abandoned at a fairly early date; this may have been a wise decision; but it would be interesting to know how close we are, 30 years and 239,000 concepts later, to achieving it; or, if this is not an reasonable measure, what has been accomplished in terms of commonsense reasoning by any other measure. There are not even very many specific examples of commonsense reasoning carried out by CYC that have been published.

There have been conflicting reports about the usability of CYC from outside scientists who have tried to work with it. Conesa et al.[8] report that CYC is poorly organized and very difficult to use:

"The Microtheory Taxonomy (MT) in ResearchCyc is not very usable for several reasons:

1. There are over 20,000 MTs in CYC with the taxonomical structure of MTs being as deep as 50 levels in some domains.

2. There are many redundant subtype relationships that make it difficult to determine its taxonomical structure.

3. Some of the MTs are almost empty but difficult to discard.

4. Not all the MTs follow a standard representation of knowledge."

They further report a large collection of usability problems including problems in understandability, learnability, portability, reliability, compliance with standards, and interface to other systems. More broadly, CYC has had comparatively little impact on AI research—much less than less sophisticated online resources as Wikipedia or WordNet.

If a cat runs in front of a house-cleaning robot, the robot should neither run it over nor sweep it up. These things seem obvious, but ensuring a robot avoids mistakes of this kind is very challenging.

Web mining. During the last decade, many projects have attempted to use Web mining techniques to extract commonsense knowledge from Web documents. A few notable examples, of many:

The KnowItAll program[14] collected instances of categories by mining lists in texts. For instance, if the system reads a document containing a phrase like "pianists such as Evgeny Kissin, Yuja Wang, and Anastasia Gromoglasova" then the system can infer these people are members of the category Pianist. If the system later encounters a text with the phrase "Yuja Wang, Anastasia Gromoglasova, and Lyubov Gromoglasova," it can infer that Lyubov Gromoglasova may also be a pianist. (This technique was first proposed by Marti Hearst;[25] hence patterns like "W's such as X,Y,Z" are known as "Hearst patterns.") More recently, the Probase system,[50] using similar techniques, has automatically compiled a taxonomy of 2.8 million concepts and 12 million isA relations, with 92% accuracy.

The Never-Ending Language Learner (NELL)[h] program[36] has been steadily accumulating a knowledge base of facts since January 2010. These include relations between individuals, taxonomic relations between categories, and general rules about categories. As of January 2015, it has accumulated 89 million candidate facts, of which it has high confidence in about two million. However, the facts are of

g A number of organizations have done private evaluations but the results were not published.

h http://rtw.ml.cmu.edu/rtw/

very uneven quality (Table 3). The taxonomy created by NELL is much more accurate, but it is extremely lopsided. As of June 9, 2015 there are 9,047 instances of "amphibian" but zero instances of "poem."

Moreover, the knowledge collected in Web mining systems tends to suffer from severe confusions and inconsistencies. For example, in the Open Information Extraction system[i,15] the query "What is made of wood?" receives 258 answers (as of June 9, 2015) of which the top 20 are: "The frame," "the buildings," "Furniture," "The handle," "Most houses," "The body," "the table," "Chair," "This one," "The case," "The structure," "The board," "the pieces," "roof," "toy," "all," "the set," and "Arrow," Though some of these are reasonable ("furniture," "chair"), some are hopelessly vague ("the pieces") and some are meaningless ("this one," "all"). The query "What is located in wood?" gets the answers "The cemetery," "The Best Western Willis," "the cabins," "The park," "The Lewis and Clark Law Review," "this semi-wilderness camp," "R&R Bayview," "'s Voice School" [sic], and so on. Obviously, these answers are mostly useless. A more subtle error is that OIE does not distinguish between "wood" the material (the meaning of the answers to the first query) and "wood" meaning forest (the meaning of the answers to the second query).[j]

All of these programs are impressive—it is remarkable you can get so far just relying on patterns of words, with almost no knowledge of larger-scale syntax, and no knowledge at all of semantics or of the relation of these words to external reality. Still, they seem unlikely to suffice for the kinds of commonsense reasoning discussed earlier.

Crowd sourcing. Some attempts have been made to use crowd-sourcing techniques to compile knowledge bases of commonsense knowledge.[24] Many interesting facts can be collected this way, and the worst of the problems we have noted in Web mining systems are avoided. For example, the query "What is made of wood?" gets mostly reasonable answers. The top 10 are: "paper," "stick," "table," "chair," "pencil," "tree," "furniture," "house," "picture frame," and "tree be plant."[k]

However, what one does not get is the analysis of fundamental domains and the careful distinguishings of different meanings and categories needed to support reliable reasoning. For example, naïve users will not work out systematic theories of time and space; it will be difficult to get them to follow, systematically and reliably, theories the system designers have worked out. As a result, facts get entered into the knowledge base without the critical distinctions needed for reasoning. Instead, one winds up with a bit of mess.

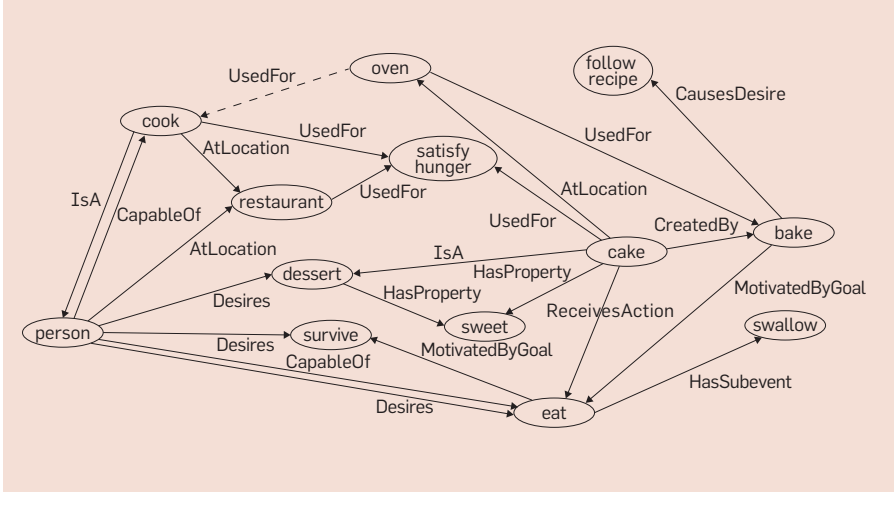
Consider, for example, the fragment of the crowd-sourced ConceptNet 3.5 shown in Figure 4. Even in this small network, we see many of the kinds of inconsistencies most famously pointed out by Woods.[49] Some of these links always hold (for example, "eat HasSubevent swallow"), some hold frequently (for example, "person Desires eat") and some only occasionally ("person AtLocation restaurant"). Some—like "cake AtLocation oven" and "cake ReceivesAction eat"—cannot be true simultaneously. The node "cook" is used to mean a profession in the link "cook isA person" and an activity in "oven UsedFor cook" (and in "person CapableOf cook"). Both cook and cake are "UsedFor satisfy-hunger," but in entirely different ways. (Imagine a robot who, in a well-intentioned effort at satisfying the hunger of its own owner, fricassees a cook.) On a technical side, the restriction to two-place relations also limits the expressivity in important ways. For example, there is a link "restaurant UsedFor satisfy-hunger," another link might easily specify, "restaurant UsedFor make-money." But in this representational system there would be no way to specify the fact the hunger satisfaction and money making have distinct beneficiaries (the customers vs. the owner). All of this could be fixed post-hoc by professional knowledge engineers, but at enormous cost, and it is not clear whether crowds could be efficiently trained to do adequate work that would avoid these troubles.

i http://openie.cs.washington.edu/

j Thanks to Leora Morgenstern for helpful discussions.

k This test was carried out in November 2014.

Figure 4. Concepts and relations in ConceptNet (from Havesi)



Going Forward

We doubt any silver bullet will easily solve all the problems of commonsense reasoning. As Table 2 suggests, each of the existing approaches has distinctive merits, hence research in all these directions should presumably be continued. In addition, we would urge the following:

Benchmarks. There may be no single perfect set of benchmark problems, but as yet there is essentially none at all, nor anything like an agreed-upon evaluation metric; benchmarks and evaluation marks would serve to move the field forward.

Evaluating CYC. The field might well benefit if CYC were systematically described and evaluated. If CYC has solved some significant fraction

of commonsense reasoning, then it is critical to know that, both as a useful tool, and as a starting point for further research. If CYC has run into difficulties, it would be useful to learn from the mistakes that were made. If CYC is entirely useless, then researchers can at least stop worrying about whether they are reinventing the wheel.

Integration. It is important to attempt to combine the strengths of the various approaches to AI. It would be useful, for instance, to integrate a careful analysis of fundamental domains developed in a mathematically grounded theory with the interpretation of facts accumulated by a Web mining program; or to see how facts gathered from Web mining can constrain the development of mathematically grounded theories.

Alternative modes of reasoning. Neat theories of reasoning have tended to focus on essentially deductive reasoning (including deduction using default rules). Large-scale knowledge bases and Web mining have focused on taxonomic and statistical reasoning. However, commonsense reasoning involves many different forms of reasoning including reasoning by analogy; frame-based reasoning in the sense of Minsky;[35] abstraction; conjecture; and reasoning to the best explanation. There is substantial literature in some of these areas in cognitive science and informal AI approaches, but much remains to be done to integrate them with more mainstream approaches.

Cognitive science. Intelligent machines need not replicate human techniques, but a better understanding of human commonsense reasoning might be a good place to start.

Acknowledgments. Thanks to Leora Morgenstern, Ken Forbus, and William Jarrold for helpful feedback, Ron Brachman for useful information, and Thomas Wies for checking our German. C

References

1. Allison, B., Guthrie, D. and Guthrie, L. Another look at the data sparsity problem. In *Proceedings of the 9th International Conference on Text, Speech, and Dialogue*, (2006), 327–334).
2. Baader, F., Horrocks, I. and Sattler, U. Descriptions logics. *Handbook of Knowledge Representation.* F. van Harmelen, V. Lifschitz and B. Porter, Eds. Elsevier, Amsterdam, 2008, 135–179.
3. Bar-Hillel, Y. The present status of automatic translation of languges. *Advances in Computers.* F. Alt, Ed. Academic Press, New York, 1960, 91–163.
4. Bojanowski, P., Lajugie, R., Bach, F., Laptev, I., Ponce, J., Schmid, C. and Sivic, J. Weakly supervised action labeling in videos under ordering constraints. ECCV (2014), 628–643.
5. Brewka, G., Niemelli, I. and Truszczynski, M. Nonmonotonic reasoning. *Handbook of Knowledge Representation.* F. van Harmelen, V. Lifschitz and B. Porter, Eds. Elsevier, Amsterdam, 2008, 239–284.
6. Cohn, A. and Renz, J. Qualitative spatial reasoning. *Handbook of Knowledge Representation.* F. van Harmelen, V. Lifschitz and B. Porter, Eds. Elsevier, Amsterdam, 2007, 551–597.
7. Conesa, J., Storey, V. and Sugumaran, V. Improving Web-query processing through semantic knowledge. *Data and Knowledge Engineering 66*, 1 (2008), 18–34.
8. Conesa, J., Storey, V. and Sugumaran, V. (2010). Usability of upper level ontologies: The case of ResearchCyc. *Data and Knowledge Engineering 69* (2010), 343–356.
9. Curtis, J., Matthews, G., & Baxter, D. On the effective use of Cyc in a question answering system. IJCAI Workshop on Knowledge and Reasoning for Answering Questions, 2005.
10. Davis, E. The naive physics perplex. *AI Magazine 19*, 4 (1998), 51–79; http://www.aaai.org/ojs/index.php/aimagazine/article/view/1424/1323
11. de Kleer, J. Qualitative and quantitative knowledge in classical mechanics. MIT AI Lab, 1975.
12. de Kleer, J. and Brown, J. A qualitative physics based on confluences. *Qualitative Reasoning about Physical Systems.* D. Bobrow, Ed. MIT Press, Cambridge, MA, 1985, 7–84.
13. Dennett, D. *Intuition Pumps and Other Tools for Thinking.* Norton, 2013.
14. Etzioni, O. et al. Web-scale extraction in KnowItAll (preliminary results). In *Proceedings of the 13th International Conference on World Wide Web*, (2004), 100–110. Retrieved from http://dl.acm.org/citation.cfm?id=988687
15. Etizoni, O., Fader, A., Christensen, J., Soderland, S. and Mausam. Open information extraction: The second generation. *IJCAI*, 2011, 3–10).
16. Ferrein, A., Fritz, C., and Lakemeyer, G. Using Golog for deliberation and team coordination in robotic soccer. *Kuntzliche Intelligenz 19*, 1 (2005), 24–30.
17. Fisher, M. Temporal representation and reasoning. *Handbook of Knowledge Representation.* F. Van Harmelen, V. Lifschitz, and B. Porter, Ed. Elsevier, Amsterdam, 2008, 513–550.
18. Forbus, K. Qualitative process theory. *Qualitative Reasoning about Physical Systems.* D. Bobrow, Ed. MIT Press, Cambridge, MA, 1985, 85–168.
19. Forbus, K., Birnbaum, L., Wagner, E., Baker, J. and Witbrock, M. Analogy, intelligent IR, and knowledge integration for intelligence analysis: Situation tracking and the whodunit problem. In *Proceedings of the International Conference on Intelligence Analysis* (2005).
20. Fromherz, M., Bobrow, D. and de Kleer, J. Model-based computing for design and control of reconfigurable systems. *AI Magazine 24*, 4 (2003), 120.
21. Gene Ontology Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research 32* (suppl. 1), 2004, D258–D261.
22. Gentner, D. and Forbus, K. Computational models of analogy. *WIREs Cognitive Science 2* (2011), 266–276.
23. Halpern, J. *Reasoning about Uncertainty.* MIT Press, Cambridge, MA, 2003.
24. Havasi, C., Pustjekovsky, J., Speer, R. and Lieberman, H. Digital intuition: Applying common sense using dimensionality reduction. *IEEE Intelligent Systems 24*, 4 (2009), 24–35; doi:dx.doi.org/10.1109/MIS.2009.72
25. Hearst, M. Automatic acquisition of hyponyms from large text corpora. ACL, 1992, 539–545.
26. Hofstadter, D. and Sander, E. *Surfaces and Essences: Analogy as the Fuel and Fire of Thinking.* Basic Books, New York, 2013.
27. Kalyanpur, A. Structured data and inference in DeepQA. *IBM Journal of Research and Development 53*, 3–4 (2012), 10:1–14.
28. Kolodner, J. *Case-Based Reasoning.* Morgan Kaufmann, San Mateo, CA, 1993.
29. Kuehne, S. Understanding natural language description of physical phenomena. Northwestern University, Evanston, IL, 2004.
30. Kuipers, B. Qualitative simulation. *Artificial Intelligence 29* (1986), 289–338.
31. Lenat, D., Prakash, M. and Shepherd, M. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine 6*, 4 (1985), 65–85.
32. Levesque, H., Davis, E. and Morgenstern, L. The Winograd schema challenge. *Principles of Knowledge Representation and Reasoning*, 2012.
33. Lovett, A., Tomei, E., Forbus, K. and Usher, J. Solving geometric analogy problems through two-stage analogical mapping. *Cognitive Science 33*, 7 (2009), 1192–1231.
34. Miller, G. WordNet: A lexical database for English. *Commun. ACM 38*, 11 (Nov. 1995), 39–41.
35. Minsky, M. A framework for representing knowledge. *The Psychology of Computer Vision.* P. Winston, Ed. McGraw Hill, New York, 1975.
36. Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J. and Carlson, A. *Never Ending Learning.* AAAI, 2015.
37. Mueller, E. *Commonsense Reasoning.* Morgan Kaufmann, San Francisco, CA, 2006.
38. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.
39. Peppas, P. Belief revision. *Handbook of Knowledge Representation.* F. Van Harmelen, V. Lifschitz, and B. Porter, Eds. Elsevier, Amsterdam, 2008, 317–359.
40. Pisanelli, D. *Ontologies in Medicine.* IOS Press, Amsterdam, 2004.
41. Price, C., Pugh, D., Wilson, M. and Snooke, N. (1995). The FLAME system: Automating electrical failure mode and effects analysis (FEMA). In *Proceedings of the IEEE Reliability and Maintainability Symposium*, (1995), 90–95.
42. Reiter, R. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, Cambridge, MA, 2001.
43. Russell, B., Torralba, A., Murphy, K. and Freeman, W. Labelme: A database and Web-based tool for image annotation. *Intern. J. Computer Vision 77*, 1–3 (2008), 157–173.
44. Schank, R. and Abelson, R. *Scripts, Plans, Goals, and Understanding.* Lawrence Erlbaum, Hillsdale, NJ, 1977.
45. Schubert, L. *Semantic Representation.* AAAI, 2015.
46. Shepard, B. et al. A knowledge-base approach to network security: Applying Cyc in the domain of network risk assessment. *Association for the Advancement of Artificial Intelligence*, (2005), 1563–1568.
47. Surdeanu, M. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *Proceedings of the 6th Text Analysis Conference* (2013).
48. Winograd, T. *Understanding Natural Language.* Academic Press, New York, 1972.
49. Woods, W. What's in a link: Foundations for semantic networks. *Representation and Understanding: Studies in Cognitive Science.* D. Bobrow and A. Collins, Eds. Academic Press, New York, 1975.
50. Wu, W., Li, H., Wang, H. and Zhu, K.Q. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of ACM SIGMOD*, 2012.

Ernest Davis (davise@cs.nyu.edu) is a professor in the Department of Computer Science at New York University, New York, NY.

Gary Marcus (gary.marcus@nyu.edu) is a professor of psychology and neural science at New York University, and CEO and co-founder of Geometric Intelligence, Inc., a machine learning startup.

Watch the authors discuss their work in this exclusive *Communications* video. http://cacm.acm.org/videos/commonsense-reasoning-and-commonsense-knowledge-in-artificial-intelligence

DOI:10.1145/2701411

The myths, the hype, and the true worth of bitcoins.

BY AVIV ZOHAR

Bitcoin: Under the Hood

"I JUST WANT to report that I successfully traded 10,000 bitcoins for pizza," wrote user laszlo on the Bitcoin forums in May 2010—reporting on what has been recognized as the first item in history to be purchased with bitcoins.[a] By the end of 2013, about five years after its initial launch, Bitcoin has exceeded everyone's expectations as its value rose beyond the $1,000 mark, making laszlo's spent bitcoins worth millions of dollars. This meteoric rise in value has fueled many stories in the popular press and has turned a group of early enthusiasts into millionaires.

Stories of Bitcoin's mysterious creator, Satoshi Nakamoto, and of illegal markets hidden in the darknet have added to the hype. But what is Bitcoin's innovation? Is the buzz surrounding the new cryptocurrency justified, or will it turn out to be a modern tulip mania? To truly evaluate Bitcoin's novelty, its potential impact, and the challenges it faces, we must look past the hype and delve deeper into the details of the protocol.

Bitcoin, a peer-to-peer digital cryptocurrency launched in 2009, has been slowly growing. Nakamoto described the protocol in a white paper published in late 2008[25] and released the software as an open source project, which has since been maintained by a large number of developers, most of them volunteers. Bitcoin's network and its surrounding ecosystem have grown quite substantially since its initial release. Its dollar value, which most will admit is largely based on speculation on its future worth, has been extremely volatile. The currency had gone through several hype-driven bubbles and subsequent devaluations, attaining higher values each time.

Bitcoin's promise is mainly a result of the combination of features it bundles together: It is a purely digital currency allowing payments to be sent almost instantly over the Internet with extremely low fees. Like cash, it is nearly anonymous, and transactions are effectively irreversible once they are committed. Bitcoin addresses (the

» key insights

- **Bitcoin's operation relies on the Block Chain—a distributed ledger of transactions that is synchronized between all nodes. The main challenge the protocol successfully tackles is to ensure nodes agree on the contents of this ledger.**
- **Going forward, the protocol faces challenges in several domains: ensuring the privacy of users, scaling up to high throughput, maintaining mining decentralization, more easily deploying updates to the core protocol, increasing the robustness of its overlay network, and structuring rewards within the protocol to improve incentives.**
- **Continuous innovations are slowly addressing these challenges. Along with applications outside of the economic domain, Bitcoin may yet fulfill its promise to become a meaningful force in the global money transmission market.**

a https://bitcointalk.org/index.php?topic=137.0

IMAGE BY ASAF ELIASON

equivalent of accounts) are free, and anyone can open as many as they would like. Set apart from other existing forms of digital currency, Bitcoin is based on a decentralized protocol: There is no organization or government in control of its operation. As a consequence, there is no central entity able to apply monetary policy, and its supply has been set in advance—there will never be more than 21 million bitcoins.

Without the initial support of a government or some other large central entity, initial adoption has been slow. Early adopters experienced the negative side of the network effect: having relatively few places to spend bitcoins, or to acquire them has made them less useful. The uncertain regulatory and legal status, the failure of many exchanges,[12,23] as well as the initial lack of user-friendly software wallets have also hindered growth.

All this is slowly changing. Exchanges that trade local currencies for bitcoins have appeared in more places, including ATMs that exchange bitcoins for cash. Digital wallets with improved interfaces can be found in app stores, and point-of-sale systems now allow any business to accept bitcoins more easily than ever before. Progress has also been made on legal and regulatory aspects. In some countries it is now clear how bitcoin transactions are taxed, and regulators have started to draft guidelines for exchanges and banks (most notably, New York's so-called BitLicense[10] has been recently put into effect). From a security standpoint, Bitcoin's core protocol and its network have been surprisingly resilient and have not been successfully compromised to date, adding to the confidence in its foundations.[b]

Will Bitcoin expand to become a substantial part of the payments market, or will it vanish as a passing trend?

b Other systems that use bitcoins have been hacked, and large sums of money have been stolen.

Only time will tell. While not without its flaws, Bitcoin does not need to be perfect to become prevalent—no system is—it need only compete with the alternatives; cash, credit cards, and wire transfers all have their downsides and imperfections. But whether or not it survives, Bitcoin's grand experiment promises to have a deep impact on the way we think of financial systems.

Bitcoin's core innovation, which may yet extend its impact beyond digital currencies, lies at the heart of a well-known problem in computer science, namely, the Byzantine consensus problem. Dealing with the synchronization of information in a distributed system in the presence of malicious adversaries, Byzantine consensus[17] has been extensively researched and several algorithms and impossibility results are known. Bitcoin's main contribution amounts to a solution to a variant of the consensus problem—one that does not require participants to have strong identities, but instead relies on assumptions that limit the computational resources of attackers.[22] But what does agreement over information in a distributed system have to do with money? To explain, we must first discuss the traditional design of digital money, and how Bitcoin's design differs.

Digital money, double spending, and the intermediary. Any viable medium of exchange must have a limited supply. Physical forms of money have always had this property. Precious metals, much to the dismay of alchemists, could not be easily produced, and modern bank notes have had many anti-forgery countermeasures embedded in them. In the age of the Internet, digital money has a clear advantage: it is faster to transmit. Unfortunately, information cannot easily replace physical tokens. It is too easy to replicate, and anyone who uses some string of bits as payment would be able to keep a copy, and perhaps use it to pay someone else. This problem, which is inherent to digital currencies, is known as the double spending problem.[c]

The classic solution to double spending, one at the foundation of most modern online banking systems, is to do away with tokens altogether. Money is, after all, a form of memory representing who has provided services and goods to others. Instead of holding physical tokens that represent credit, it is possible to list the holdings of each individual in a ledger. Transferring money is then accomplished simply by changing the records on the ledger—stored in the memory of some server—adding to one account balance, and subtracting from the other.

This design adds a third party to all transactions—the record keeper. This intermediary is given a great deal of power: It can refuse to carry out certain transfers, to change balances even without the consent of the transacting parties, or to demand high fees in exchange for its indispensable services, something that had never been possible with physical forms of money. Additionally, in contrast to the anonymity of cash transactions, the privacy of individuals transacting with digital currency is compromised. The intermediary itself is explicitly notified of every transaction that takes place. Finally, the existence of record keepers through which all payments are funneled allows for government intervention and regulation. Regulation, which serves to hinder criminal activity and to guard against misuse of the funds by the intermediary itself, has its downsides. Regulatory compliance imposes a direct cost on organizations, and also introduces barriers that restrict entry to the money transmission market, reduce competition, and so serve to increase fees even further.

Bitcoin seeks the best of both worlds: to enjoy the full benefits of the digital domain, but also to greatly weaken any third party through competition and decentralization. Most of Bitcoin's features are natural implications of this choice: the inability to reverse payments and the fixed supply of money, for example, are natural design choices when there is no centralized entity that can verify money has been stolen and payments should be reversed, or whether or not more money should be issued. Many other beneficial properties of Bitcoin are achieved by the application of more modern practices: Unlike credit cards that bear the burden of backward compatibility and have card numbers and expiration dates that are easy to steal,[2] access to bitcoins is guarded by public key cryptography. Other advantages of Bitcoin are due to its open nature. The open source model boosts its transparency, adds confidence in its stability and security, and grants open access to its APIs, which enable agile development within the surrounding ecosystem.

Replacing the intermediary. In order to reduce the influence of any third party but still allow funds to be transferred, Bitcoin's design replaces the centralized intermediary with many weaker entities that maintain the ledger. The nodes in charge of Bitcoin's transaction processing, also known as miners, form a large and connected peer-to-peer network that together authorizes all money transfers. Each miner checks the actions of others to ensure money is not mishandled, and competes to authorize a share of the transactions.

One of the main goals of the protocol's design is to make it easy to join the network. Anyone can download the open source software and use readily available hardware to run a node. Nodes connect to each other via TCP connections over the Internet and share the IP addresses of other known peers to form a robust distribution network. Thus, no one is granted absolute control of the system, and no single entity is able to block transactions, or to extract unreasonably high fees.

Unlike a distributed design that aims to share the load among many machines, Bitcoin nodes do not partition the workload or the ledger among them. In fact, in order to allow each node to act independently, data is massively replicated, and each participant repeats all verification work. This replication naturally requires all nodes to be notified of every transaction, as each transfer of funds must be recorded in all copies of the ledger. Users who wish to send money create a message requesting the transfer. Transfers are made between Bitcoin addresses, which act as the approximate equivalent of an "account" (each address is in fact the hash of a cryptographic public key). The sender's message is digitally signed to prove ownership of the funds, and is transmitted to some of the nodes in the network. Nodes verify the signatures and then forward the message to their peers, ensuring it is sent to the entire network. As

c The exact state of a quantum bit cannot be copied, and so quantum currency systems that disallow double spending are theoretically possible.[1,31]

a consequence, all bitcoin transactions are public.

Notice that nothing stops the owner of funds from creating and signing two conflicting transaction messages that transfer the same funds to different destination addresses. This, in fact, is the double spending problem as it manifests itself in Bitcoin. Nodes that have received these transactions may adopt different ones and consequently disagree about the state of the ledger. This is where Bitcoin's main innovation is rooted, at the synchronization of information in its ledger.

The Block Chain

Bitcoin's main data structure, the block chain, is the key to understanding how information consistency is maintained between nodes and how conflicts are resolved. The block chain, as its name suggests, is composed of blocks—batches of approved transactions that have been grouped together. Each block contains the cryptographic hash of its predecessor that, for all intents and purposes, serves as a unique identifier of the previous block (hash collisions are very rare, and difficult to find—an important property of cryptographic hash functions).

The block chain thus forms an incremental log of all transactions that have ever occurred since the creation of Bitcoin, starting with the "Genesis Block"—the first block in the chain. If one reads the log from start to finish, every transfer of money can be verified and funds can be followed to compute the balance of each Bitcoin address. Nodes that were offline can easily catch up by requesting only the few recent blocks that they are missing.

The block chain grows steadily as new blocks extend it, referencing their predecessors, and adding new transactions. Newly created blocks are flooded within the network to ensure all nodes possess them. In order to maintain the consistency of the chain, valid blocks are only allowed to include transactions consistent with current balances as determined by their predecessors in the chain.

If all nodes possess the exact same copy of the block chain, then all is well; the ownership of every fraction of a bitcoin is known and agreed upon by everyone. This, unfortunately, is not always the case. The network is distributed, and the creation of blocks is uncoordinated. Thus, blocks that are formed approximately at the same time by different nodes may extend the same parent block and create a fork in the chain. Such blocks represent a different version of the transaction log, and are likely to contain conflicting transactions (for example, see Figure 1).

Bitcoin's promise is mainly a result of the combination of features it bundles together: It is a purely digital currency allowing payments to be sent almost instantly over the Internet with extremely low fees.

We have finally reached the core of the Bitcoin protocol: its mechanism for selecting between conflicting histories. The mechanism consists of two main rules that govern block creation and adoption:

1. *Block creation is difficult (by design).* Valid blocks are required to contain a proof-of-work: the solution to a computationally hard problem. A solution to the problem is easily verifiable, but finding it requires many guesses to be made, and takes a long time (the problem itself is based on the repeated application of cryptographic hashing to the block's contents; see the sidebar "Bitcoin's Proof-of-Work" for additional information).

2. *Adopt the longest chain.* Blocks are distributed throughout the network. When nodes learn about conflicting blocks that make up a longer consistent chain, they adopt it and abandon blocks in their shorter version.[d]

The two rules work together to bring the network to consensus regarding the history of transactions. First, as blocks are rarely created, few conflicts occur to begin with: If block creation is infrequent, a new block will most likely be propagated to all nodes before the next one is produced. The next block will thus reference it and will not include conflicting transactions. The difficulty of the computational problem is automatically adjusted so blocks are created only once every 10 minutes in expectation throughout the entire network. This period of time is sufficiently long to make conflicts extremely rare.

The longest-chain rule resolves these conflicts and ensures the net-

d In fact, nodes do not pick the longest chain, but rather the chain that contains the highest aggregate difficulty of proof-of-work computations (this measure is used because the difficulty of block creation is regularly adjusted by the protocol). It is simpler, however, to think instead of the length of the chain as this aggregate measure.

Bitcoin's Proof-of-Work

To make block creation difficult, the protocol requires the cryptographic hash of each block (or, to be more precise, the hash of the block's header) will be a small number under some threshold (called "the target"). The block's header contains a field called the nonce that can contain an arbitrary string of bits. If the block's hash is too large, the nonce can be changed and the hash can be recomputed. An important property of strong cryptographic hash functions is that a change to even a single bit in their input completely and unpredictably changes their output. Many attempts are thus needed to find a nonce that will fit the block, and produce a hash below the target. For example, if the target has 60 zeros in its most significant bits, fewer than one in 2^{60} attempted hashes will result in a successful attempt, requiring miners to perform a great deal of computational operations to create a valid block.

In addition to changes to the nonce, every change to the contents of the block also changes its hash, so once a match is found, the block cannot be modified. The proof-of-work can be easily verified by each node that later receives the block, simply by checking its hash value.

To ensure blocks are created in expectation once every 10 minutes, the threshold for successful block creation is adjusted automatically every 2,016 blocks. If, for example, blocks were created too quickly (as is often the case if additional hashing power was added to the network since the previous adjustment) the difficulty is raised by lowering the target threshold.

work will eventually converge to a single choice: If two conflicting blocks exist, each node in the network adopts one of them, but not the other, as the alternative is not currently part of a longer chain. The network is thus partitioned to nodes that accept one version of events, or the other. Once another block is created by one of the nodes, the tie is broken, and one of the possible versions of transaction history becomes longer. This version will then propagate and be adopted by the entire network. Ties among conflicting chains may in fact last longer, but eventually, due to the random nature of the computation involved in the block creation process, one chain will win the race, and the other will be abandoned.

Notice that as longer chains replace shorter ones, some blocks are discarded along with their contents. Transactions included in these blocks that do not appear in the replacing chain disappear from the ledger. Moreover, if a conflicting transaction exists in the newly adopted chain, the original transaction cannot be included in an extension of the new chain. The mechanism used to choose between different versions of the chain can thus be exploited by a resourceful attacker to reverse payments. This form of attack, as we shall shortly see, is difficult to carry out without access to a large share of computing resources.

Double spending attacks. Consider an attacker that has paid some merchant, has had its transaction embedded in the block chain, but wishes to reverse it (after obtaining some goods in return). The attacker may then use the fact that nodes will adopt an alternative version of the block chain if it is longer than their current one. It can try to create a fork of the current chain that splits off just before the block containing his transaction, and extend this fork in the chain until it is longer than the current chain. Once published, this alternative chain (in which the attacker includes a conflicting transaction that redirects the funds) will take over as the accepted version of events and the attacker's original transaction will be discarded along with the block that contained it.

It is obvious then, that transactions are never fully irreversible in the system; a longer chain may always appear. Such an occurrence, however, becomes increasingly unlikely. Notice that the attacker needs to create enough blocks in his version of the chain to overtake the current main chain. Since block creation requires a difficult proof-of-work computation, the attacker must either have a great deal of computational resources, or be extremely lucky. He must produce blocks at a higher rate than the rest of the network combined in order to overtake the current chain.

Bitcoin's developer Nakamoto has shown in his original analysis that as long as an attacker possesses less than 50% of the computational power in the network, he produces blocks at a lower expected rate than the rest of the nodes, and so the probability of a successful attack on a given transaction decreases exponentially as more blocks are added to the chain on top of it.[25] Each block added is thus considered to add a "confirmation" to all the transactions in preceding blocks as it supports their inclusion in the ledger.

The network is therefore more secure the more computational power there is in the hands of honest nodes. For example, Nakamoto's analysis,[25] (later improved by Rosenfeld[29]), shows that after approximately six confirmations, an attacker with 10% of the computational power can succeed with probability lower than 0.00059. The costs of a mining operation capable of mining even 10% of the blocks is extremely high, establishing a barrier against double spending attacks of transactions that are deeply embedded in the chain.

Merchants—especially those collecting payments in the presence of the buyer—cannot afford to keep customers waiting even for the 10 minutes required for the first confirmation of a transaction. Many have opted instead to accept transactions with 0-confirmations once they have been sufficiently distributed through the network, trusting they will later be included in the block chain. Thus far, relatively few attacks on 0-confirmation transactions have taken place, but such practices still pose a risk.[6,16]

Resilience to the double spending attack relies strongly on the assumption that Bitcoin's P2P network is connected, and that honest nodes are able to communicate. Without communication, blocks and transactions cannot be distributed well. While several mechanisms have been put in place to maintain connectivity, Bitcoin's overlay network has been shown to be susceptible to eclipse attacks in which relatively few attacker nodes manage to attract a sizeable portion of the connections and isolate others from the network.[15] Lower-level infrastructure attacks may also cause problems, especially those committed by adversaries that control many routers, IP addresses, and other network resources (as evidenced by recent BGP hijacking

attacks that were successfully used against mining pools[8]).

The 50% attack. A miner that holds over 50% of the network's computational power can create blocks faster than all other nodes combined, and thus represents a more serious threat to the network. It is always able to create blocks at a faster pace than the rest of the network combined, which allows it to change the block chain, and double spend any transaction it issued (regardless of its depth in the block chain). In fact, it is capable of committing much more devastating attacks: by simply generating a chain of empty blocks and ignoring all other blocks it can stop all transactions from entering the block chain. Interestingly, a 50% attacker is still somewhat limited: it cannot move funds it does not control, as transaction messages still must be cryptographically signed by the sender.

Rewards and Incentives

Bitcoin includes an important incentive mechanism that encourages mining activity and thus indirectly increases the system's resilience. Miners are awarded with bitcoins in return for their effort: Each transaction offers a small fee claimed by the node that includes it in a block. Transactions compete for limited space in blocks and so market forces should eventually set the fees. Nodes, in turn, are incentivized to join the network, to collect transactions, and include as many of them as possible in blocks. As a side effect, they contribute their computational power toward improving the network's resilience to double spending.

In addition to fees, creators of blocks are awarded newly created bitcoins. Rewards are issued to the block's creator in a special transaction included in each block called the coinbase transaction. Bitcoin's money creation is gradual, and occurs according to a fixed schedule. Instead of launching the system with all coins in the hands of a single individual, Nakamoto decided to spread their distribution over time. Starting with the Genesis Block, each new block issued 50 bitcoins. The number of bitcoins generated in this manner is halved every 210,000 blocks (approximately every four years), and so the total sum of bitcoins that will ever exist is nearly 21 million. This fixed money supply is one of the key economic features of Bitcoin. It leaves no room for monetary intervention and essentially implies the currency is deflationary (bitcoins may be lost and never replaced if, for example, the private keys needed to transfer them are lost).

Mining bitcoins has become increasingly more attractive as their value has gone up—turning bitcoin mining into a fast-growing industry. Miners have transitioned from using PCs to more efficient hardware such as GPUs, and eventually ASICs—custom designed chips that efficiently perform the operations needed for Bitcoin's proof-of-work. While a single CPU provides several mega hashes per second, modern mining rigs are approximately a million times faster, performing several tera hashes per second. As a result, the network's hash rate has grown to over 300 peta hashes per second, making it more secure against double spending attacks.

Strategic behavior. While rewards have generally attracted more nodes and have strengthened the network, it is important to consider other behaviors nodes may adopt in order to increase their profits from mining. In particular, if nodes find it profitable to deviate from the protocol, the system's performance may deteriorate.

Indeed, some activities related to the basic maintenance of the network have not been properly incentivized

Figure 1. A fork in the block chain.

Two conflicting blocks extend a chain. The two, which are created and held by different nodes, contain somewhat different transaction sets including conflicting transactions (tx9a and tx9b in this example).

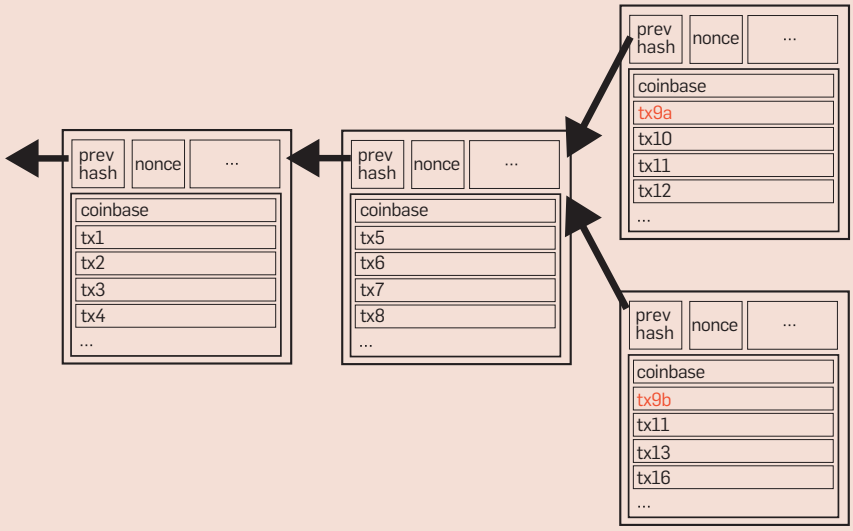
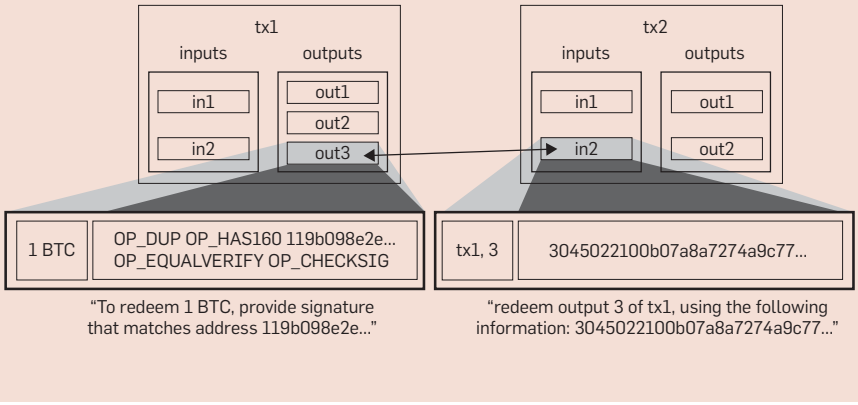


Figure 2. The structure of transactions.



with payments. Storage costs, for example, are not accounted for and are not reflected in fees. Anyone moving a small amount of money via the block chain creates records that might never be expunged and will forever take up space on all full nodes.

Research has additionally shown that nodes are not properly incentivized to share information. For example, while the protocol requires nodes to flood transaction messages to each other, those who do not distribute messages may gain higher rewards simply by removing the chance that some other node will claim the transaction fees associated with the withheld transaction.[4] Payments to nodes who forward messages may add incentives for distribution and correct this problem.

Another example of bad incentives in the protocol relates to mining behavior: The protocol requires nodes to create blocks that reference the longest chain and to publish them immediately. A paper by Eyal and Sirer[13] shows that nodes can strategically delay the release of blocks they create in order to become more profitable. This selfish behavior is profitable mostly for large miners and is easily detectable, making it somewhat less likely to be used in practice. Still the fact remains: the protocol—in its current form—is susceptible to some level of manipulation by selfish participants.

Mining pools. Given the large number of participants in the network, a small miner can only expect to create a relatively small fraction of the blocks—roughly equivalent to its share of the computational power. For some, this may mean finding a block only after months of continuous attempts. Such rare large rewards may provide a sufficiently high expected payment, but miners that would like to obtain a steady revenue stream also worry about the risk: Payments have high variance and some months may go by without any reward.

The solution to this problem appeared in the form of mining pools: groups of miners that join together and split the profits from mining to receive lower, but more frequent payments. A miner that chooses to participate in a pool contributes his computational power and works to generate the proof-of-work for a block the pool's server is creating. If successful, the pool distributes the block reward among the participants, aiming to provide rewards in proportion to the effort of each miner and guaranteeing them nearly the same expected payment as mining on their own (the difference is due to fees the pool collects for its services and to small inefficiencies in the slightly more complex block creation process). While early implementations of pool reward mechanisms suffered from incentive problems that allowed miners to gain more than their fair share of the rewards through manipulations, newer mechanisms, now in place at most pools, do better.[28] Game theoretic analysis of the competition between pools to attract miners shows there may well be no stable partitions, and that miners will continually switch.[18]

Pools have been incredibly successful—a relatively small number of them are currently creating the majority of the blocks in the network prompting concerns that too few entities control block creation. CEX.IO, a company that owns a great deal of mining hardware and runs GHash.IO—a popular mining pool—has approached 50% of the network's hash rate on several occasions, a size that could potentially allow them to disrupt the Bitcoin network with a 50% attack. While no attack was launched by the owners of the pool, many have been concerned the system is at risk if, for example, a hacker manages to compromise the pool's servers. The resulting public outcry caused many miners to switch to other pools, and prompted CEX.IO to state they will restrict the size of their pool in the future. It is important to note however, that there is no way to ensure any single entity controls under 50%, as it may be mining using different addresses and through multiple servers. The lack of strong identities in the protocol implies Bitcoin is inherently not secure when a pool controls more than 50% of the computational resources.

A key aspect of the Bitcoin protocol is the way it represents and changes the ownership of money. Every Bitcoin transaction is in fact a reassignment of money from inputs to outputs.

The Structure of Transactions

Another key aspect of the Bitcoin protocol is the way it represents and changes the ownership of money. Every Bitcoin transaction is in fact a reassignment of money from inputs to outputs (many to many). Outputs are

composed of a sum of money associated with a short script in Bitcoin's scripting language. Scripts can be thought of as a sort of challenge: anyone wishing to move the money associated with an output must provide a string that will make the script return "true."[e] Transaction inputs point to a previous transaction output and contain the string that correctly answers the challenge. Thus, a transaction, which is a combination of inputs and outputs, proves the owner is allowed to transfer money from previous outputs that it owns and redirect it into new transaction outputs. Once an output is used, all of the money associated with it is considered spent. Other transactions that attempt to access the same output will be considered conflicting transactions and will be rejected. Since money from inputs does not necessarily sum up to the amount one may wish to send, transactions often include an output that returns any leftover funds back to the sender.

While it is possible to write simple scripts like "check to see if the input equals 3" that can be easily satisfied, the most commonly used output script is one that requires a cryptographic signature in order to free funds. The script compares the signature provided to the public key associated with a certain bitcoin address and allows the output to be used only if they match (The address is the hash of the public key with some additional bits used for error detection, in case it is mistyped). Ownership of bitcoins is therefore just a matter of knowing the right input to the script. From a practical perspective, a merchant that wishes to receive funds needs to send his address to the buyer. The buyer then creates a transaction with an output that is redeemable by anyone who possesses the corresponding private key, which is kept secret by the merchant for future use.

Other more complicated scripts are also possible; For example, scripts requiring signatures generated by two different keys (effectively implementing a joint account that needs consent from two sources for every transfer).

e Complexities related to infinite loops and to the halting problem have been avoided by making the scripting language less expressive. It is not Turing complete.

Privacy, anonymity, and auditability. The structure of transactions and the fact they are publicly available on the block chain allows anyone to follow money and see where it is being moved. This is both a blessing and a curse. On one hand, organizations that wish to do so can reveal which addresses they control and allow anyone to see how they are using their money. On the other hand, the privacy of individuals is compromised. Since addresses are easily and freely generated, it is possible to generate a unique address for every transaction. This helps restore privacy to some extent, but some information is always leaked even when Bitcoin addresses are not reused.[3,20,27]

The mixture of partial privacy and transparency within Bitcoin has led to interesting innovations. The collapse of MtGox, the large bitcoin exchange, which had lost a sizeable amount of bitcoins was followed up by forensic analysis of transaction data that dispelled some possible explanations for its loss of funds.[12] Exchanges have since been pressured to implement mechanisms that allow account owners to securely and privately verify that their balances are indeed held by the exchange. Similar mechanisms have been applied in other domains like crowd funding, online gambling, and charity fund-raising. On the opposite side of the privacy spectrum, some organizations utilize the relative privacy offered by Bitcoin to hide their activities. As an extreme example, criminal organizations like the Silk Road, an online market for illicit goods that had been busted by authorities in the U.S., benefit from the relative anonymity of Bitcoin addresses.

The public aspects of money also enable the use of taint analysis: coins considered to have been involved in illegal activity can be tracked no matter how many times they change hands, and can be treated differently: exchanges, for example, may refuse to accept them. Marking money in this way may have devastating consequences on its fungibility—another important property of money.[24] Issues of privacy, anonymity, taint, and regulation are at the center of debate within the Bitcoin community, and are naturally of great concern to policymakers.

Mixing services and protocols such as CoinJoin allow users to mask the origins and destination of payments by mixing together many transactions and splitting the outputs in ways that do not allow them to be easily associated with the corresponding inputs. Zerocash, a protocol modification designed to provide enhanced anonymity, uses advanced cryptographic tools to allow nodes to process transactions without knowing the details of the transfer.[7] These modifications and others reshape the mixture of privacy and transparency that Bitcoin and similar protocols may provide.

What Does the Future Hold?

Scalability. The Bitcoin protocol is highly wasteful. A high amount of effort is expanded in arbitrary proof-of-work computations. Thus far, no provably secure replacement that uses fewer resources or utilizes the computation for useful purposes has emerged, although many have tried to suggest alternative designs. In addition to the proof-of-work, Bitcoin's design requires wasteful replication. All relevant information is saved at all mining nodes, messages are essentially broadcast through the network, and verification is always repeated. For these reasons, it appears the system would not scale well. Bitcoin's block size has been artificially (and somewhat arbitrarily) limited to 1MB per block. The protocol currently processes under two transactions per second on average, a rate that has been steadily, albeit slowly, increasing. Fortunately, the average transaction size is relatively small, averaging approximately 0.5KB per transaction, which currently allows all transactions generated between block creation events to clear. Concern about the growth of transaction rates has caused some core developers to push for an increase in the block size limit and has sparked lively debate. Those who oppose argue the costs of running nodes will increase beyond the reach of "regular users."

But can Bitcoin scale to process much more significant volumes? As a hypothetical scenario, one may consider rates of 2,000 transactions per second (which are closer to the order of

magnitude of Visa's worldwide transaction volume). With 0.5KB per transaction, the flow of data needed to keep up with all transactions is only approximately 1MB per second (additional protocol messages will in fact require a bit more). Storing all these transactions in the block chain implies storage will grow at a rate of around 2.5TB per month.[f] While this is a high rate of growth, outside the reach of home users, it is certainly manageable for a small mining business even with today's technology.

It is important to note that even mining nodes do not have to hold the entire history of transactions. Some of the contents of the block chain can be safely erased. Furthermore, everyday users of the currency that do not engage in mining do not need to store the full block chain. They can manage with a much smaller portion of the data. Simplified Protocol Verification clients (SPV), also known as light nodes, allow users to connect to the network and download only the information they require. Such nodes are light enough to run on mobile devices, and drastically alleviate storage costs for small users. Miners and others who run full nodes are the only ones that need to hold a full copy of the block chain.

The size of blocks, however, has other important implications. Large blocks take longer to transmit and to propagate through the network. As a result, more conflicting blocks will be created. This fact has been empirically observed,[11] and has severe implications to Bitcoin's resilience to double spending attacks. With many conflicting blocks, the block chain does not grow efficiently, and many of its created blocks are discarded implying that weaker attackers can overtake it. Security deteriorates well before bandwidth limits are reached. An alternative to the longest-chain selection rule, nicknamed GHOST, has been shown to alleviate this security problem.[30] Additional ideas, such as replacing the block chain with a directed acyclic graph structure[19] to include transactions from off-chain blocks, block compression, and off-chain transactions channels[9] offer further improvements to transaction throughput.

Another problem encountered under high transaction rates is the reward distribution between miners becomes skewed in favor of larger, better connected miners (that is, miners connected to the Bitcoin network). This may endanger Bitcoin's decentralized nature, as small miners that cannot invest heavily in connectivity quickly become unprofitable.

Decentralization at risk. While the Bitcoin protocol is decentralized, the current system is in fact controlled in many aspects by small groups of miners, and wallet providers. Protocol development too, is in the hands of relatively few developers.[14]

The race for advanced ASICs used in bitcoin mining is still ongoing, and hardware is often made obsolete within months. As the electricity costs of running a PC far exceed the rewards it generates, mining using CPUs has quickly become a losing proposition. Mining is thus gradually shifting to the hands of larger organizations that continuously invest in the latest hardware. Some have suggested using alternative proof-of-work procedures that will make specialized hardware less effective and will thus weaken this effect (one such example appears in Miller et al.[21]).

Other strong economic forces are also pulling Bitcoin in the same direction of increased centralization. Large miners enjoy effects of increasing returns to scale; They can produce their own hardware, or purchase it en masse, or they may better optimize the location of their mining centers in order to gain access to cheaper electricity. Similar advantages result from the specific nature of the protocol. Storage and bandwidth costs, for example, are the same regardless of the miner's size. This greatly benefits large miners that pay less per generated block for these overheads. Smaller, less profitable competitors, are then slowly eliminated from the market.

Development and protocol changes. Protocol updates in Bitcoin are difficult. Unlike more conventional software, a bug in Bitcoin's core may cause inconsistencies between different versions of the code and may cause the block chain to split. Such an event occurred in March of 2013. A bug in the code caused two versions of the protocol to behave differently (one version refused to accept a block created by the other) which resulted in a long-lasting fork in the block chain. Large mining pools were quickly asked to downgrade to the older version, which eventually resolved the split.

Similarly, intentional updates that do not maintain backward compatibility may cause the chain to fork if they are not accepted by all. Such updates cannot be rolled out gradually—they require a majority of the network to accept them before they are activated. Bitcoin's core developers have therefore been extremely conservative with updates. This justifiably careful behavior also implies the protocol itself is slowly "calcifying," as substantial updates become progressively more difficult to roll out.

Alternative currencies. Bitcoin's open source code has been used to launch many alternative currencies (altcoins). Many have been created by applying relatively minor modifications to its code. One example is Litecoin, which aims to be "the silver to Bitcoin's gold." Litecoin's proof-of-work hashing algorithm has been changed in hope of preventing ASICs from dominating the mining race and its blocks are created at a somewhat accelerated rate of once every 2.5 minutes (ASICs were eventually developed for Litecoin mining as well). Many alternative currencies have found some following (for example, Dogecoin is based on a famous Internet meme), but have usually struggled to attract many miners, and to maintain a secure network.

Not all altcoins are minor modifications. Some include more substantial changes, and have taken ideas from Bitcoin to new realms. Namecoin, for example, uses its block chain as a key-value store rather than to manage a currency (one of its uses is as a distributed alternative to DNS).

In this context, it is also worthwhile to mention Ripple and Stellar,[26] (https://www.stellar.org) two companies developing protocols not derived directly from Bitcoin, but that create a distributed system for money transfer. Both are primarily based on a network of IOUs that are transferred locally, and have dif-

f The block chain's size today is approximately 40GB, and it currently includes all of the transactions since Bitcoin was launched in 2009.

ferent consensus mechanisms.[26]

Altcoins are considered by many to be the proving grounds for new risky ideas that may someday be incorporated into Bitcoin if they have proven to be viable. Others complain that there are many "pump-and-dump" schemes, that is, coins that are created with a lot of hype to lure in naïve speculators who invest money and get little in return. Ideas like sidechains[5] that may allow bitcoins to be exchanged for other compatible alternative coins have been suggested as potential mechanisms that allow for easier integration with Bitcoin, and may be the path for safer innovation in cryptocurrencies.

Beyond money. While initially designed only to encode monetary transfers, it quickly became clear that Bitcoin's block chain can be used to encode other pieces of information.

Examples range from innocuous ASCII art images to WikiLeaks cables that have been embedded in transactions. This has raised several concerns both regarding legal aspects of embedding copyrighted or otherwise prohibited information into the block chain (which is then copied to every full Bitcoin node).

Discontent with the scripting capabilities that Bitcoin offers, some higher-level protocols have opted to extend the functionality of its scripting language to include additional actions. Counterparty, Omni, and Colored-Coins are several higher-level protocols that do just that. Reasoning that Bitcoin's network is large and highly secure, these protocols use Bitcoin transactions to encode more sophisticated scripts that allow for multiple currencies to exist within its block chain. Other higher-level functions like distributed exchanges, bets, and financial derivatives are also enabled.

The Ethereum project, which uses a separate block chain, has taken transaction scripts one step further and developed a Turing-complete scripting language. Ethereum allows anyone to create contracts, which are essentially programs that are executed jointly by the nodes in the Ethereum network. Ethereum's block chain is used to maintain the state of each contract, and transaction messages generate events that update these states.

The realization that decentralization has value in and of itself, as well as the rise of platforms like Ethereum, has led some to believe computer programs can become fully autonomous economic entities. "Decentralized Autonomous Corporations" (DACs), can collect fees (paid with cryptocurrencies) for services rendered, and use them to pay for servers, and other resources they consume. Existing within decentralized platforms, they can truly have a life of their own, independent of their creator, without depending on any single machine to run their code.

Conclusion

Bitcoin's design fundamentally reshapes and reimagines money—one of humanity's most basic and foundational social constructs. Essentially allowing us to transmit value over the Internet just as easily as we transmit information, its disruptive nature promises to change markets, enable new business models, and impact the ability of governments to control money and to regulate businesses. While still facing many challenges, a steady stream of innovations and solutions is continuously being developed to address its shortcomings. The evolutionary path of the protocol and of the system itself is greatly influenced by the protocol's technical strengths and weaknesses, but also by strong social, political, and economic undercurrents. Miners, developers, regulators, and adopters all affect the direction of its growth. With ongoing development, and possible applications beyond the financial domain, Bitcoin, and other protocols that extend it, may yet come to deeply impact our lives. C

References

1. Aaronson, S. Quantum copy-protection and quantum money. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity*. IEEE, 2009, 229–242.
2. Anderson, R. and Murdoch, S.J. EMV: Why payment systems fail. *Comm. ACM, 57*, 6 (June 2014), 24–28.
3. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T. and Capkun, S. Evaluating user privacy in Bitcoin. *Financial Cryptography and Data Security*. Springer, 2013, 34–51.
4. Babaioff, M., Dobzinski, S., Oren, S. and Zohar, A. On bitcoin and red balloons. In *Proceedings of the ACM Conf. on Electronic Commerce*. ACM, 2012, 56–73.
5. Back, A. et al. Enabling blockchain innovations with pegged sidechains; http://blockstream.com/sidechains.pdf.
6. Bamert, T., Decker, C., Elsen, L., Wattenhofer, R. and Welten, S. Have a snack, pay with bitcoins. In *Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing*, Sept. 2013.
7. Ben-Sasson, E. et al. Zerocash: Decentralized anonymous payments from Bitcoin. In *Proceedings of the IEEE Security and Privacy Symposium*. IEEE, 2014.
8. BGP hijacking for cryptocurrency profit; http://www.secureworks.com/cyber-threatintelligence/threats/bgp-hijacking-for-cryptocurrencyprofit/.
9. Bitcoin lightning network; https://lightning.network/.
10. BitLicense; http://www.dfs.ny.gov/legal/regulations/adoptions/dfsp200t.pdf.
11. Decker, C. and Wattenhofer, R. Information propagation in the Bitcoin Network. In *Proceedings of the 13th IEEE International Conference on Peer-to-Peer Computing* (Sept. 2013).
12. Decker, C. and Wattenhofer, R. Bitcoin transaction malleability and MtGox. In *Proceedings of the 19th European Symposium on Research in Computer Security* (Wroclaw, Poland, Sept. 2014).
13. Eyal, I. and Sirer, E.G. Majority is not enough: Bitcoin mining is vulnerable. *Financial Cryptography and Data Security, LNCS* (2014). 436–454.
14. Gervais, A., Karame, G., Capkun, S. and Capkun, V. Is Bitcoin a decentralized currency? *IACR Cryptology ePrint Archive 829* (2013).
15. Heilman, E., Kendler, A., Zohar, A. and Goldberg, S. Eclipse attacks on Bitcoin's peer-to-peer network. In *Proceedings of 24th USENIX Security Symposium*. Aug. 2015 (to appear); http://eprint.iacr.org/2015/263.pdf.
16. Karame, G., Androulaki, E. and Capkun, S. Two bitcoins at the price of one? Double-spending attacks on fast payments in Bitcoin. *IACR Cryptology ePrint Archive 248* (2012).
17. Lamport, L., Shostak, R. and Pease, M. The byzantine generals problem. *ACM Trans. Programming Lang. Systems 4*, 3 (1982), 382–401.
18. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and Rosenschein, J.S. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. 2015, 919–927.
19. Lewenberg, Y., Sompolinsky, Y. and Zohar, A. Inclusive block chain protocols. *Financial Cryptography and Data Security*. Springer, 2015 (to appear).
20. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M. and Savage, S. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the 2013 Conference on Internet Measurement*. ACM, 127–140.
21. Miller, A., Juels, A., Shi, E., Parno, B. and Katz, J. Permacoin: Repurposing Bitcoin work for data preservation; http://cs.umd.edu/ amiller/permacoin.pdf, 2014.
22. Miller, A. and JLaViola, Jr., J.J. Anonymous byzantine consensus from moderately hard puzzles: A model for Bitcoin, 2014.
23. Moore, T. and Christin, N. Beware the middleman: Empirical analysis of Bitcoin-exchange risk. *Financial Cryptography*. Springer, 2013, 25–33.
24. Möser, M., Böhme, R. and Breuker, D. Towards risk scoring of Bitcoin transactions. *Financial Cryptography and Data Security, Lecture Notes in Computer Science*. Springer, 2014, 16–32.
25. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. 2008; https://bitcoin.org/bitcoin.pdf.
26. Ripple; https://ripple.com/.
27. Ron, D. and Shamir, A. Quantitative analysis of the full Bitcoin transaction graph. *Financial Cryptography and Data Security*. Springer, 2013, 6–24.
28. Rosenfeld, M. Analysis of Bitcoin pooled mining reward systems. arXiv preprint (2011); arXiv:1112.4980.
29. Rosenfeld, M. Analysis of hashrate-based double spending; https://bitcoil.co.il.Doublespend.pdf, 2012,.
30. Sompolinsky, Y. and Zohar, A. Secure high-rate transaction processing in Bitcoin. *Financial Cryptography and Data Security*. Springer, 2015 (to appear).
31. Wiesner, S. Conjugate coding. *ACM SIGACT News 15*, 1 (1983) 78–88.

Aviv Zohar (avivz@cs.huji.ac.il) is a senior lecturer in the Rachel and Selim Benin School of Computer Science and Engineering, at The Hebrew University of Jerusalem, Israel, and a visiting researcher at Microsoft Research, Israel.

research highlights

DOI:10.1145/2801947

To view the accompanying paper, visit doi.acm.org/10.1145/2801945 rh

Technical Perspective
A Woodworker's Easy Fix

By Marc Alexa

I LIKE WOODWORKING. I have built quite a number of pieces of furniture, like cabinets, shelves, beds, and tables, as well as a ladder, a shack, and other things. Not being professionally trained, I restrict myself to simple construction projects, for which I trust my intuition on stability. For more complex constructions I stick to the standards I find in books. This obviously limits my freedom.

As a computer scientist, I know it would be possible to check the validity of a construction using physical simulation. If the panels chosen are the right type (thickness or kind of wood), the main structural problems are where the pieces come together—in the joints. Each joint bears linear and rotational forces; the result of the weight of the wood itself and the stuff being put onto or into the object (for example, the person on a ladder). There are natural limits to these forces in the joints, which I could find in the literature. Given all this information (combinatorial structure of the panels, dimensions, loads, limits on joint forces) checking validity is a straightforward project in scientific computing.

Now, what would I have gained from this programming project? If the computer tells me my construction is stable I can go ahead and build it. But what if it does not confirm? I would change my design and try again. This is obviously a tedious exercise. Nonetheless, this "workflow" is the current design standard.

Wouldn't it be much better if the computer told me how to fix my construction? Or, even better, offered several ways to fix it, so that I could just chose one? Certainly yes, but this is a difficult problem. Just as in NP problems: deciding if a given answer is correct is easy, but finding one is hard. Here, the technical difficulty is the large dimension and the mixed discrete-continuous nature of the design space: each panel has two dimensions, each joint has numerous parameters, and some constructions can only be fixed by changing the combinatorial structure (typically adding a panel), hence the mixed discrete-continuous space.

Nonetheless, the authors of the following paper have found a way to provide the user with instant feedback on how to fix unstable or toppling wooden panel constructions. They formulate the desired goal as finding *small* design changes, that is, the length of the vector of design parameter changes is minimal.

The solution to this problem requires several key ingredients: First, it is easier to check for validity in the space of joint forces. So, for the current design, they linearize the connection between the design space and the 'force space' (and make some additional simplifying assumptions). This allows them to quickly check if a small change in the design space leads to a feasible construction. Still, the dimension of the design space is too large to find short vectors that make the construction feasible. The second main idea is to limit the change to only a few independent parameters, while the rest stay fixed. This limitation drastically reduces the dimension of the search space, not only making it feasible to compute solutions (for all possible combinations of changeable parameters), but also easier to understand the suggested modification for the human user in the loop. Lastly, discrete changes are restricted to adding a panel, and the location of the panel is based on established rules.

Together with some additional features in the user interface, the resulting system really solves the problem of developing, rather than just checking, wood panel constructions. As a woodworker, I cannot wait to get my hands on this system. As a computer scientist, I am more interested in the general solution for the interactive search of feasible solutions in a high-dimensional design space—this solution will serve as a blueprint for a variety of related design problems. C

Marc Alexa is a professor in the Department of Computer Science at Technische Universität Berlin, Germany.

DOI:10.1145/2801945

Guided Exploration of Physically Valid Shapes for Furniture Design

By Nobuyuki Umentani, Takeo Igarashi, and Niloy J. Mitra

Abstract

It is common to use computers to design shapes of physical objects such as furniture, but geometric modeling and the physical validity of shapes are conventionally considered independently. This makes creating aesthetically pleasing yet physically valid models challenging. In this paper, we propose an interactive design framework for the efficient and intuitive exploration of geometrically and physically valid shapes. During any geometric editing operation, the system continuously allows visualization of the valid range of the parameters being edited. When one or more constraints are violated following an operation, the system generates multiple suggestions involving both discrete and continuous changes to restore validity. Each suggestion is accompanied by an editing mode, which simultaneously adjusts multiple parameters in a coordinated manner to maintain validity. Thus, while the user focuses on the aesthetics of the design, our computational design framework helps to achieve physical realizability by providing active guidance to the user. We demonstrate our framework on plank-based furniture designs with nail-joints and frictional constraints. We use our system to design a range of examples, conduct a user study, and also fabricate a physical prototype to test its validity and usefulness.

1. INTRODUCTION

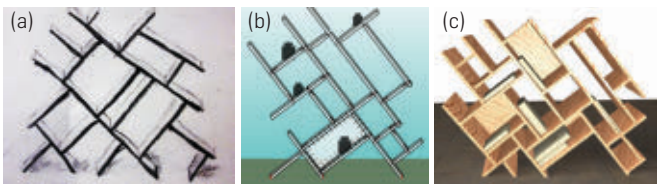
Recent advances in three-dimensional (3D) modeling systems (such as the appearance of Blender and SketchUp) have enabled novice users to design complex shapes, thus making content creation widely accessible. However, along with aesthetic appeal of the designed shapes, the physical properties are often very important, particularly if the resulting model is to be fabricated and used in the real world. For example, in the context of do-it-yourself (DIY) furniture design, various physical constraints must be satisfied. For example, a chair is only useful if it remains stable and does not break with the target load distribution. However, current modeling systems typically do not consider such physical plausibility in the design phase. This makes creating interesting shapes that also satisfy physical constraints difficult for novice users, who may not have specialist knowledge or relevant experience.

Geometric design and physical functionality are conventionally considered independently. The designer typically creates a 3D geometric shape that is then validated using a physical simulation, for example, a finite element method (FEM) solver. If the shape violates one or more physical constraints, it will be returned to the designer, who refines the shape. This process is repeated until a satisfactory design is found. Such a workflow may be undesirable for a number of reasons: the process is time consuming, essentially amounting to trial-and-error; it provides no guidance to the designer on how to rectify violations of the design constraints; and it encourages users to opt for standard shapes rather than to explore novel shapes.

For example, IKEA provides a range of design-at-home tools that are specialized for offices, kitchens, and bedrooms, and allow the user to define the dimensions of the room, interactively select 3D models from product catalogs, and place them in the room to create a layout. However, such systems only allow users to select from a list of preexisting objects. With the growing demand for customization, it is desirable for a system to allow users to change the shape of the furniture, while still being guaranteed that the objects remain functional, that is, will not, for example, collapse under target loads. Thus, our goal is to support such design exploration by providing real-time physical analysis.[17,20,22] We investigate this in the context of nail-jointed furniture, with the aim of producing unusual and artistic shapes with non-standard inclinations, yet ensuring physical validity (see Figure 1).

A few advanced computer-aided design (CAD) systems (e.g., CATIA) support continuous feedback to check for validity. Similarly, Umetani et al.[20] proposed an interactive system to provide real-time feedback on the physical constraints for garment simulations. Such methods, however, only inform the user whether or not the model is valid; they do not suggest *how* to restore validity. Whiting et al.[21] reported a system to optimize procedurally generated buildings over a range of

Figure 1. Modeling a design concept (a) often produces invalid 3D realizations (b) due to model instability (i.e., toppling) or non-durability (i.e., excessive joint forces) under target loads. Our interactive computational design framework supports guided shape exploration to help the user reach a valid configuration, which then can be readily manufactured (c).



The original version of this paper is entitled "Guided Exploration of Physically Valid Shapes for Furniture Design" and was published in *ACM Transactions on Graphics* (SIGGRAPH), August 2012.

variables to produce a final model that is structurally stable. However, such an approach is not satisfactory for exploratory modeling as it neither provides creative support nor facilitates informed exploration of the design space.

We introduce a computational design framework for the efficient and intuitive exploration of valid shapes. Specifically, we actively guide the user to explore parts of the shape space that satisfy the constraints, thus relieving the user of the burden of ensuring realizability, via the following two modes: we analyze the current shape configuration and indicate the valid range of the parameter being edited; then we propose both continuous and discrete suggestions with coordinated editing modes to restore validity when the design becomes invalid. Note that, in contrast to a direct optimization-based solution, we leave the designer in control of form-finding by providing a visualization of the valid range with multiple deformation suggestions, as required, to guide the designer toward feasible geometric forms (see Figure 2).

In this work, we enable constrained modeling in the context of nail-jointed furniture design, while observing geometric and physical constraints. We consider three aspects: *connectivity*, that is, joint connections among planks are geometrically maintained; *durability*, that is, that the object does not fail at any of the joints under the target load distributions; and *stability*, that is, that the object does not topple or lose contact with the ground. The user interactively designs a shape model using standard modeling operations. In the background, the system continuously runs a rigid body simulation with frictional contact to provide real-time feedback on the structural validity of the design. The system uses a sensitivity analysis to describe how changes to the design affect the validity of the design.

We use this information to provide a range for the valid parameters being edited, as well as continuous suggestions to restore validity using a novel force space analysis approach. Each suggestion is accompanied by a coordinated editing mode that synchronously adjusts multiple components, which otherwise may be difficult for users to achieve manually, especially while satisfying multiple nonlinear constraints. Thus, the user can efficiently navigate the physically valid design space by following the ranges shown in the visualization and exploring the proposed suggestions (see Figures 1 and 14, the supplementary video and the executable demo).

The main contributions of this work are that we provide: (i) an interactive modeling framework for guided exploration of physically valid shapes, (ii) a design environment for nail-jointed, plank-based furniture, and (ii) a force space sensitivity analysis to generate design suggestions with continuous and discrete modifications to restore geometric and physical validity.

2. EXPLORING PHYSICALLY VALID 3D DESIGNS

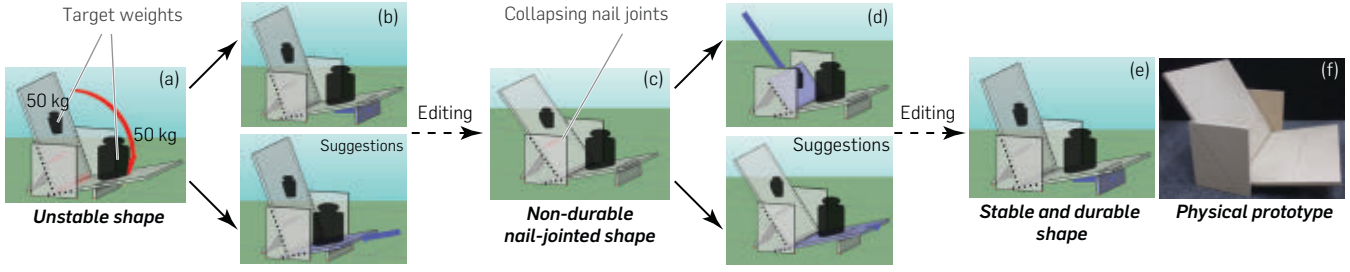
2.1. Suggestive modeling

Advances in geometric modeling have resulted in well-established CAD modeling tools. Exploratory design, however, remains challenging. This is mainly because mapping a partially formed design concept to a final 3D shape is ambiguous. Hence, researchers have proposed various frameworks for suggesting possible shapes to inspire and guide the user. Based on user-specified geometric relations across 3D components, Igarashi and Hughes[7] generate a gallery of possible subsequent modeling operations to facilitate quick and intuitive modeling. In another influential work, Funkhouser et al.[5] propose a data-driven modeling system in which the user provides the conceptual design using sketches, while the system suggests plausible geometric realizations by searching through a database of 3D models. Inspired by this philosophy, Chaudhuri and Koltun[4] propose a data-driven system to compute and present components that can be added to the current design shape. The Insitu system[12] provides spatial context by fusing data from multiple sources and combines them with image-billboarding to provide light-weight 3D environments for conceptual designs. In this work, we introduce a novel suggestive system for exploring shapes that are physically valid.

2.2. Physical simulation

Given a 3D shape, state-of-the-art techniques can efficiently and accurately evaluate its physical validity. However, the inverse problem is less understood. In the context of animation, even small perturbations in initial parameters can lead to large changes in final configurations. Hence, Popović et al.[14] propose a system to optimize parameters to satisfy user annotated keyframe specifications. In another interesting formulation, Twigg and James[18] introduce backward steps to generate animations of rigid bodies with desired goal configurations. In this work, instead of motion parameters,

Figure 2. Given physically invalid designs due to model instability (i.e., toppling) (a) or non-durability (i.e., excessive joint force) (c), we propose design suggestions (b, d) to restore physical validity. The suggestions provide guided shape space exploration to the user, who can quickly realize valid nail-jointed furniture designs under target weight-bearing and practical material specifications (e, f).



we explore how shape changes affect the physical validity of a shape. We then use the information to interactively propose smart shape deformation modes.

2.3. Interactive shape exploration

Immediate and meaningful feedback is essential in any design setting, especially in artistic exploration (see also Kerr and Pellacini[8]). Although such design spaces are often high dimensional, only low-dimensional subspaces are typically useful to intuitive exploration. In a data-driven setting, researchers have extracted low-dimensional embeddings (e.g., using mixture of Gaussian models) of desirable design spaces for appearance[15] and for geometric modeling.[17] Recently, Ovsjanikov et al.[11] study variation patterns directly in appropriate descriptor spaces to extract low-dimensional deformation models on a representative template for exploration and navigation of collections of 3D models. In another approach, the deformation framework iWires[6] demonstrates that direct preservation of inter- and intra-part relations using junction curves is effective for manipulating man-made models. More recently, Yang et al.[22] propose a geometric framework to identify constrained modeling spaces where appropriate geometric properties in the form of algebraic constraints (e.g., planarity of quad faces) are preserved in the course of deformations and edits. Such exploration approaches, however, mostly focus on geometry, and physical validity considerations have so far been ignored.

2.4. Design optimization

Different optimization strategies have been proposed for various design problems: voxel selection and placements to create multiple target shadows,[10] relief optimization for prescribed shadow footprints,[1] furniture layout while increasing functional considerations such as accessibility,[9, 23] or optimizing combinations of materials to reach target deformation behavior.[3] In the context of buildings, Smith et al.[16] model truss structures by structural optimization, while Whiting et al.[21] optimize free variables in the context of procedural modeling with regards to structural feasibility by ensuring non-negative force between brick elements. These approaches propose final optimized shapes, which are not beneficial in initial exploratory stages. Instead, we introduce shape space investigation to understand the effect of geometric changes on physical validity and use the findings to expose the valid and useful parts of the shape space as suggestion modes (see also Yang et al.[22]).

3. SYSTEM OVERVIEW

3.1. Overview

Figure 3-left shows our modeling interface, which consists of a modeling panel and a suggestion panel. The modeling panel basically works as a standard modeling system (e.g., Google SketchUp), although it is specialized for models consisting of multiple planks connected by nail joints. Our system continuously checks for validity in the background and shows whether or not the configuration satisfies geometric and physical requirements. Specifically, the system examines connectivity, durability, and stability. Note that we do not check for self-intersections at runtime. The result of the analysis appears as an annotation in the main panel during mouse dragging. Further, we provide suggestions in the suggestion panel after mouse release if the current shape is invalid. Suggestions, when selected, appear in the modeling panel.

3.2. Modeling user interface

Figure 3-right shows the basic modeling operations supported by our system. The user draws two 2D lines on the screen to specify a new rectangular plank (a–c) of predefined thickness (12 mm in our setting). The first line is drawn by mouse dragging and is placed on a selected plank. The end point of the first line becomes the starting point of the second line and its end point is indicated by a mouse click. The second line is either projected to an existing plank or aligned to the canonical *xyz*-coordinate system. We automatically generate a joint between the newly created plank and the existing planks on which the first and second lines are placed. The user can translate, rotate, and scale a plank using 3D widgets (d–f). When an edge of a plank is placed near another plank, these planks are automatically connected (g). Finally, the user places a weight by clicking on a plank in the weight mode (h). Note that the exact placement location of the weight on the selected plank is not important.

3.3. Validity visualization and suggestions

In Figure 4, we show the different scenarios when the current configuration becomes invalid. (a) When a joint becomes disconnected, the system highlights the joint in red. (b) When the model breaks at a joint, the system also highlights the joint in red. (c) When the model falls down, the system shows a red arrow. These warnings automatically appear and are continuously updated as the user interacts with the

Figure 3. (Left) The modeling interface consists of the modeling and the suggestion panels. (Right) The modeling interface with typical stages shown: creation (a-c), translation (d), rotation (e), scaling (f), and connection (g), along with placing a weight (h).

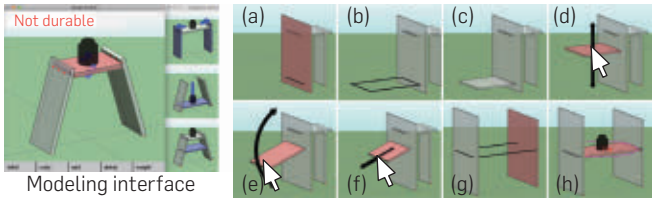
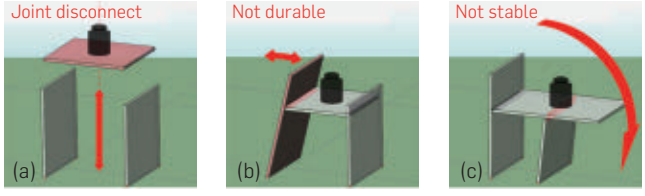


Figure 4. Different warnings flagged by our system for invalid configurations: joints get disconnected (a), a model becomes non-durable due to excessive force on the nails (b), or it becomes unstable, that is, topples (c).



design, so that the user can move back to a valid state by direct manipulation based on the feedback.

In addition to checking whether or not the current configuration is valid, the system computes the valid *range* of the parameter (degrees of freedom, DOF) being manipulated and shows it to the user during direct manipulation (mouse drag). When the current configuration is valid, the system shows the valid range in black. When the current configuration is invalid, the system shows the valid range in red (see Figure 5). Explicitly showing the valid range reduces the need for trial and errors to stay within or return to a valid state during direct manipulation editing.

When necessary, after each mouse release, the system provides suggestions (capped to a maximum of 8 in our setting) on how to resolve an invalid state. When a joint becomes disconnected, the system shows how to reconnect it (Figure 6a). When the model is non-durable or unstable, the system shows how to make it durable and stable (Figure 6b and c). Each suggestion consists of a representative configuration and an optional coordinated edit mode. When the user clicks on a suggestion, the representative configuration appears in the modeling panel together with arrow marks indicating the coordinated edits (Figure 7a). The user drags one of these arrows to make coordinated editing, thus allowing the user to control multiple DOFs of a model simultaneously while satisfying the required constraints. These multiple DOFs are coupled together, that is, the user cannot fix the non-durability or instability moving each DOF individually. For example, in Figure 7, if the user slides the top board of the table toward the left, the angle of the left leg becomes perpendicular to the ground to compensate for the increase of the bending force on the left joint (Figure 7b and c).

Figure 5. Range indicators. Range is shown in black when the current configuration is valid and in red when invalid.

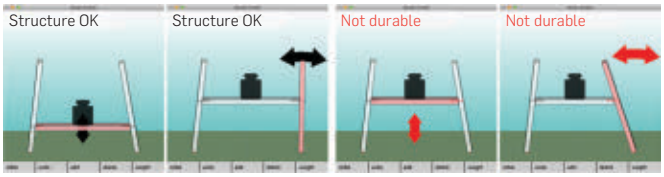


Figure 6. Example of suggestions. A joint is connected (a), the model is made durable (b), or the model is made stable (c).

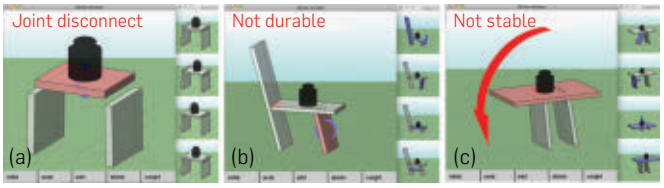
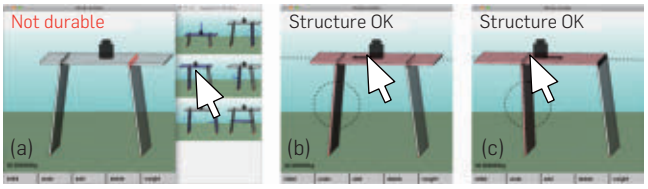


Figure 7. Example of coordinated editing using suggestions. The table is non-durable and the system gives multiple suggestions (a). The user clicks on a suggestion and it appears in the modeling window (b). The user can change the position of the top board and left leg simultaneously by dragging any of the arrow handles (c).



4. ALGORITHM OVERVIEW

As the user edits the model (i.e., adds, removes, translates, rotates, or scales objects), we first attempt to restore the geometric constraints (i.e., joint connectivity and ground contact) by adjusting the length of the other objects. If this fails to satisfy the geometric constraints, we suggest discrete changes to fix the design. Once the model is made to satisfy the geometric constraints, we check the physical validity of the current shape and present the results to the user. We test for durability and stability, which amounts to checking for inequality constraints at joints and contact forces, respectively. In addition to indicating whether the design is valid or not, we analyze how the validity changes with respect to further geometric modifications, that is, what changes make an invalid model valid, and vice versa. The results of this analysis are used to compute valid ranges and make suggestions. Section 5 describes how we measure and analyze the physical validity, and Section 6 describes how we compute the valid ranges and make suggestions based on this analysis.

5. PHYSICAL VALIDITY

In our interactive framework, we continuously analyze the current design to provide feedback to the user as to the physical validity of the current shape during editing. Specifically, the system checks for two kinds of physical validity: (i) whether or not the nail joint is durable and (ii) whether or not the structure is stable. In this section, we describe how to compute joint and contact forces and how to measure joint durability.

In any nail-jointed wooden structure, the joints form the weakest links, that is, such structures typically fail at the joints.[13] For this reason, in our framework we model component planks of wooden furniture as assemblies of unbreakable rigid bodies, focusing on the joint and the contact forces.

5.1. Forces on a joint and contact point

We assume static equilibrium. Under this assumption, all of the forces in a rigid plank should be balanced and, therefore, the sum of rotational moments and the sum of linear moments will both be zero. Furthermore, the nail-joints should connect two planks rigidly, hence the relative position of any two planks is constrained such that their translation and rotation will be the same. We consider a further constraint in that the predefined contact points do not penetrate the floor. We determine the static equilibrium of the rigid plank by solving for static energy minimization under these constraints. Then we arrive at the translation $\mathbf{h}^t$ and the bending force $\mathbf{h}^r$ at each joint, and the contact force $\mathbf{f}_{cont}$ at each contact point.

5.2. Joint durability

With the translation $\mathbf{h}^t$ and the bending force $\mathbf{h}^r$ at each joint, we check for durability of the nail-joints in response to the given forces. The mechanical properties of nails are well known, and have long been standardized with precise specifications in terms of the load-bearing capacity (see Bergman[2]). At each joint, there are two loads on the nails: (i) a pulling force, which acts along the axis of the nail, and (ii) a shearing force, which acts perpendicular to the axis of the nail. We represent these forces $\mathbf{h}$ (i.e., $\mathbf{h}^t$ and $\mathbf{h}^r$) in a local coordinate system, where h_n is the component normal to the joint face of plank P_j, h_x is the component along the normal of plank P_i, and the remaining component is h_y (see Figure 8). Each component of $\mathbf{h}^r$ denotes the torque to twist the plank P_j with an axis of rotation in each direction. We assume the thickness of the plank is less than the width of the joint between P_i and P_j. Hence, the bending force h_y determines whether the joint will fail. In Figure 8, we show how the force that pulls the nail from the joint arises from the bending force h_y^r. The joint forms a lever where the length of the lever arm is $0.5l_z$; the pulling force is given by

$$f_{pull} = \frac{1}{N_{nail}} (2|h_y^r|/l_z - h_n^t), \quad (1)$$

where l_z represents the thickness of the plank (12 mm in our tests) and N_{nail} denotes the number of nails in joint N_{ij}. The shear force is then given by

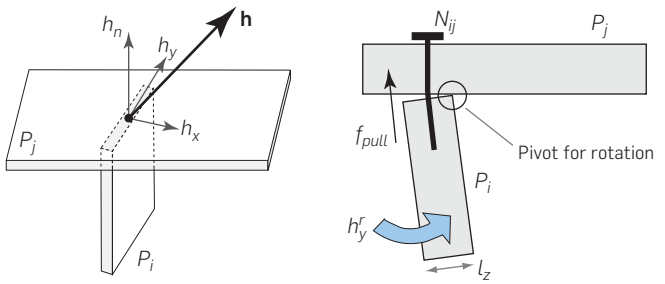
$$f_{shear} = \frac{1}{N_{nail}} \sqrt{h_x^{t2} + h_y^{t2}}. \quad (2)$$

We determine a joint to be durable if both forces are within allowable threshold margins.[2] Please see Umetani et al.[19] for further details.

6. EXPLORATION OF VALID SPACES

If the current design is valid, we indicate the range of user manipulations that maintain design validity. If the current design becomes invalid, we make multiple suggestions to restore validity. Note that, even though the (unconstrained) configuration space has a large number of dimensions, our framework only exposes meaningful (i.e., valid) suggestions, thus greatly simplifying the task for the user. We make both continuous and discrete suggestions: while continuous suggestions leave the inter-plank joint topology unchanged, discrete suggestions involve adding support materials.

Figure 8. (Left) Decomposition of a constraint force into components in the local coordinate system. (Right) The rotational force leads to a force that pulls the nail, which can affect the durability of the joint.



6.1. Geometric constraints

Aside from the physical validity of our system's shapes, that is, durability and stability, shapes are geometrically restricted by two further constraints: (i) geometrical joint constraints and (ii) contact constraints (see Section 5). We first restrict the design space so that the shape satisfies these geometrical constraints, and then investigate the physical validity. Each plank has 8 DOFs in the design: three for translation, three for rotation, and two for edge lengths around the plank faces (the plank thickness is fixed). For each DOF, we ensure that the contact constraints and joint constraints are satisfied by adjusting the length of the planks (Figure 9-left). Furthermore, some DOFs may be invalid, for example, if both ends of a plank are nailed, the plank length cannot be adjusted (see Figure 9-right). We identify and remove such invalid DOFs from the design space. Note that if there are C plank components and $\#\text{DOF}_{invalid}$ invalid design DOFs, the constrained design space Γ has dimensionality of $N_\gamma := 8C - \#\text{DOF}_{invalid}$. Each basis corresponds to one translation, rotation, or change in length of a given plank and/or adjacent plank. We scale the translation and length change bases inversely with the edge length of the maximum bounding box to make the DOFs for translation and changes in length dimensionless, as is the case for rotational DOFs. Then we enable an exploration of a physically valid subspace of a constrained design space Γ.

6.2. Valid shape space

Recall that a shape is physically valid if two conditions are satisfied: (i) the shape is *durable*, which amounts to each joint with both pulling and shear forces below allowed thresholds, which is formalized as

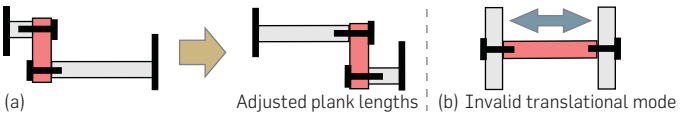
$$|f_{pull}| \leq f_{pull\,max} \text{ and } |f_{shear}| \leq f_{shear\,max}\ \forall N_{ij}, \quad (3)$$

and (ii) the shape is *stable* (i.e., it does not topple), which amounts to each contact point with a non-negative contact force f_{cont} in the direction normal to the ground, that is,

$$f_{cont}^l \geq 0\ \forall \text{ contact points } l. \quad (4)$$

Let the corresponding subspaces of the configuration space Γ be Γ^{durable} and Γ^{stable}. Thus, the valid shape space is $\Gamma^{\text{valid}} := \Gamma^{\text{durable}} \cap \Gamma^{\text{stable}}$. When the current design becomes invalid, the goal is to provide multiple suggestions to return to the valid shape space (see Figure 10).

Figure 9. Constrained design modes: (a) the lengths of neighboring planks of the edited planks are adjusted so that joints stay connected; (b) a translation mode is invalid if both ends of the planks are jointed.



The valid space typically has a complex boundary since it is characterized by nonlinear inequality constraints. Furthermore, because the configuration space has a larger number of dimensions, computing exact boundaries is difficult and time-consuming. In addition, it is almost impossible to arbitrarily pick a valid shape directly from the high-dimensional space Γ^{valid}. Instead, we first pick several meaningful search directions to pursue, that is, directions whereby the invalid shape becomes valid under relatively small manipulations. For each such direction in the design space, we use a line search to identify configuration intervals where all of the validity conditions are satisfied.

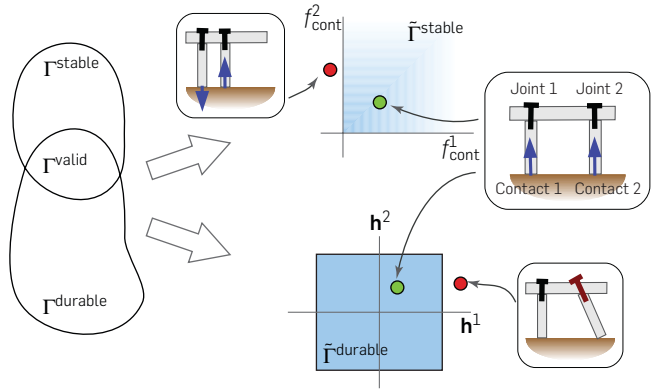
Because the boundaries of $\Gamma^{durable}$ and Γ^{stable} are characterized by force inequalities, we consider the valid shape space boundary in the *force space*, that is, a coordinate space with forces as the axes. This simplifies the problem as the boundary is then geometrically prescribed by the corresponding inequality. For example, with two contact points, the stable region is 2D and Equation (4) simply indicates that the first quadrant is in the stable region (see Figure 10).

To efficiently characterize the force space describing joint durability, we make two approximations: (i) the translation force $\mathbf{h}^t$ in Equation (1) remains constant with small design changes and only the bending force $\mathbf{h}^r$ varies, and (ii) the shearing force in Equation (2) does not change with small design changes. These approximations are good when the bending force $\mathbf{h}^r$ is dominant, and the design is more sensitive to $\mathbf{h}^r$ than $\mathbf{h}^t$ in response to changes in the design. Thus, Equation (3) becomes

$$|h_y^r| \leq 0.5 l_z (f_{pull\,max} N_{nail} + h_n^t) = \Lambda_{max}. \quad (5)$$

Geometrically, the stable region Γ^{stable} can be approximated as a high-dimensional, axis-aligned cuboid with edge lengths of Λ_{max} and centered at the origin in the joint bending force space.

Figure 10. A shape is valid if it is both stable and durable. For invalid shapes, we propose deformation suggestions to return to the valid part of the shape space. We work in force spaces defined by the contact and bending forces for stability and durability, thus simplifying the problem. Specifically, stability amounts to contact forces being restricted to the first quadrant, while durability amounts to bending forces being restricted to a durability rectangle. Note that, although in this example the force spaces are two-dimensional (2D), in general we work in higher-dimensional spaces.



Note that the number of dimensions of both the contact force space and of the bending force space are smaller than that of the configuration space ($|\Gamma| \approx 8C$). In particular, the contact force space has the same number of dimensions as the number of contact points, and the joint bending space has the same number of dimensions as the number of joints N_{ij}. Next we describe how to efficiently search for directions in this simplified representation. We denote the boundaries of the stable and durable force space as $\tilde{\Gamma}^{stable}$ and $\tilde{\Gamma}^{durable}$, respectively.

6.3. Visualization of the valid range

During direct editing, we display the valid range of the parameter being manipulated. To do this, we evaluate the validity by changing the parameter. If the current configuration is already valid, the search proceeds in both directions until it becomes invalid to identify the bounds. If the current configuration is invalid, we first select the direction of the search using the results of the sensitivity analysis, and then run a bisection search along that direction to identify the valid range; this is described in the following subsection.

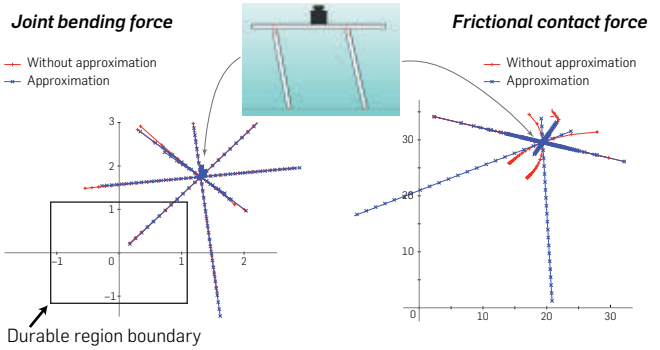
6.4. Continuous shape suggestions

When the current configuration becomes invalid, we compute several suggested configurations: (i) if only stability is violated, the system finds designs to restore stability by analyzing the boundary of $\tilde{\Gamma}^{durable}$; (ii) if only durability is violated, the system finds designs to restore durability by analyzing the boundary of $\tilde{\Gamma}^{stable}$; and (iii) if both stability and durability are violated, the system first proposes designs to restore stability, and then to restore durability.

To generate valid suggestions, we first assume forces vary linearly as a function of the changes to the design, and find candidate suggestions. Then we run a physics simulation using the nonlinear constraints to compute whether the designs are valid. Figure 11 shows a sample comparison with and without the linear approximation. If we cannot find valid shapes in this direction, we simply omit it from the suggestions.

Simultaneous exploration of the full design space ($|\Gamma| \approx 8C$ dimensions) is impractical for real-time performance. In addition, users may find suggestions that involve

Figure 11. Comparison of a stable shapes and a durable space with and without the linear approximation.



variation in many parts to be confusing and not helpful. Instead, we restrict suggestions to at most M DOF for any suggestion, where $M = 3$ in the examples described here. We try all possible combinations by selecting m design DOFs (where $m \leq M$), denoted by $\{\gamma_1, \ldots, \gamma_m\}$. We parameterize the changes to the design to make the candidates into a vector $\mathbf{s} \in \mathbb{R}^m$, where $\mathbf{s} := \sum_{i=1}^{m} s_i \gamma_i$.

Durability-restoring candidates. A desirable change to the design $\mathbf{s}$ should quickly make the design durable. Thus, we find a design change $\mathbf{s}$ such that the joint bending forces are in the valid region $\tilde{\Gamma}^{\text{durable}}$, while ensuring that the change is small:

$$\mathbf{s}^\star := \arg\min_{\mathbf{s}} |\mathbf{s}|, \quad \text{where} \quad \mathbf{K}_0 \mathbf{s} + \mathbf{h}^r_{y0} \in \tilde{\Gamma}^{\text{durable}}, \tag{6}$$

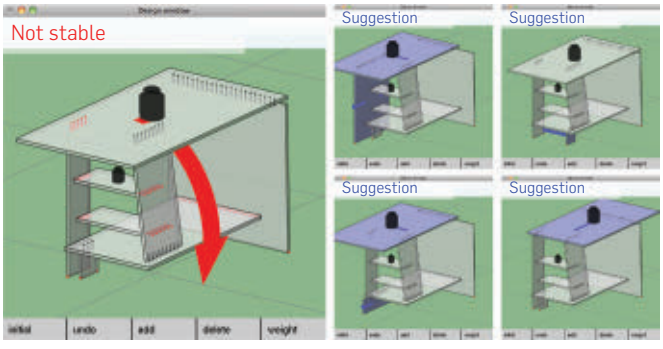
where matrix $\mathbf{K}_0 \in \mathbb{R}^{N_{ij} \times m}$ encodes the sensitivities of the joint forces with respect to design changes, that is, $\mathbf{K}_0 := \nabla_\gamma \mathbf{h}^r_y$ evaluated with the current joint bending force $\mathbf{h}^r_{y0}$. For the real-time performance necessary for an interactive software application, we restrict $\mathbf{s}$ so that the changes are in the direction which makes the norm of the bending forces as small as possible. We use a bruteforce method, whereby all possible $M!$ combinations in the configuration space are explored, taking advantage of the simple axis-aligned cuboid approximation of the durable region. Solving Equation (6) can be seen as detecting collisions of rays with the durability cuboid where a ray is in the search direction and has a origin at $\mathbf{h}^r_{y0}$. We accelerate the search by culling any direction if either the norm of the sensitivity $|\mathbf{K}_0|$ is small (≤ 1 in our tests) or the ray is in the direction away from the cuboid.

Stability-restoring candidates. We compute stable shape suggestions in a similar manner to the durability case. Specifically,

$$\mathbf{s}^\star := \arg\min_{\mathbf{s}} |\mathbf{s}|, \quad \text{where} \quad \mathbf{L}_0 \mathbf{s} + \mathbf{f}_{cont\,0} \in \tilde{\Gamma}^{\text{stable}} \tag{7}$$

and where $\mathbf{L}_0 \in \mathbb{R}^{N_{contact} \times m}$ is a sensitivity matrix of the contact forces with respect to design changes, that is, $\mathbf{L}_0 := \nabla_\gamma \mathbf{f}_{cont}$ evaluated at the location of the current contact force $\mathbf{f}_{cont0}$. The candidates are generated in the same manner as with durability restoration. Figure 12 shows some typical examples.

Figure 12. Stability-restoring suggestions.



6.5. Discrete shape suggestions

If the structure is not durable, we try to make it durable by adding a support plank as a reinforcement around a joint that is under excessive force. Typically, nail joints connect two planks that are nearly at right angles, making it difficult to attach supports between the planks that are connected by joints that are not durable. Instead, we try to connect two planks that are parallel and place a supporting plank so that it is orthogonal to the other two planks. We do this using a greedy strategy. First, we choose a combination of two planks P_i, P_j such that (i) they are nearly parallel (we use $|\mathbf{n}_i \cdot \mathbf{n}_i| < 0.5$ where, $\mathbf{n}_i$ is the face normal of the plank P_i, and $\mathbf{n}_j$ is the face normal of the plank P_j); (ii) there is a third plank P_k between the two planks, which is connected to P_i and P_j using nail joints; and (iii) one or both joints N_{ik} and N_{jk} are not durable. We suggest adding support material between P_i and P_j at a location chosen from several (rule-based) candidate positions so that the support does not intersect with any of the existing planks (see Umetani et al.[19] for further details). We check for durability of the joint by running a physical simulation to ensure that the supporting plank is effective. The system attempts numerous combinations of planks until it finds effective supporting planks based on standard rules used in woodwork.[2] There is scope here for the use of more complex strategies, which may form part of future work.

7. RESULTS

In our system, we consider furniture designs using 12 mm-thick medium density fiberboard (MDF) with 32 mm-long nails, spaced at interval of 20 mm. Such a placement can take a maximum shear force of $f_{shear\,max} = 190$ N and maximum pulling force of $f_{pull\,max} = 3.5$ kN/m.[13] We set the coefficient of static friction to 0.5. In our current implementation, we can handle 10 to 15-plank designs with sufficient speed for interactive design. In each exploration session, the user progressively adds planks and proposes an initial configuration along with the target load-bearing capacity. For example, in Figure 2, we place a 50 kg weight on the horizontal plank and a 15 kg weight on the supporting back plank. The final design was found following several iterations of suggestions and exploration of the design space. We built a physical prototype (where the construction took around 4 h) and found it to behave satisfactorily under the target load (see the supplementary video on the project page).

In Figure 13, we use our system to design non-conventional bookshelves. The computational support is critical, as we have little intuition in such unusual situations and cannot benefit from prior experience. Guided exploration helps the user to explore the design limits while not having to consider physical validity.

Figure 14 shows additional design sessions with our system. Note that we show only a few representative suggestions; we refer the reader to the supplementary video and demo for further details. The user is provided with corrective suggestions *only* when the design becomes invalid. Furthermore, each suggestion comes with a range whereby the designed shape remains valid. Thus, even when the suggestions involve multiple planks, the user only has to

Figure 13. Designing non-standard furniture is difficult for novice users. Our guided exploration framework allows users to design furniture with strange configurations easily.

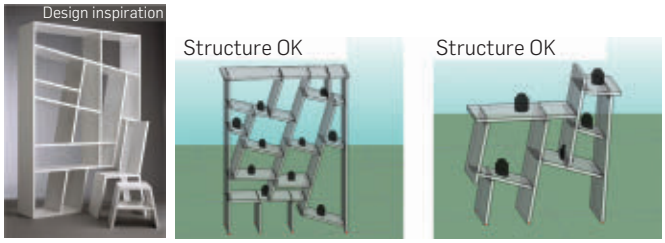
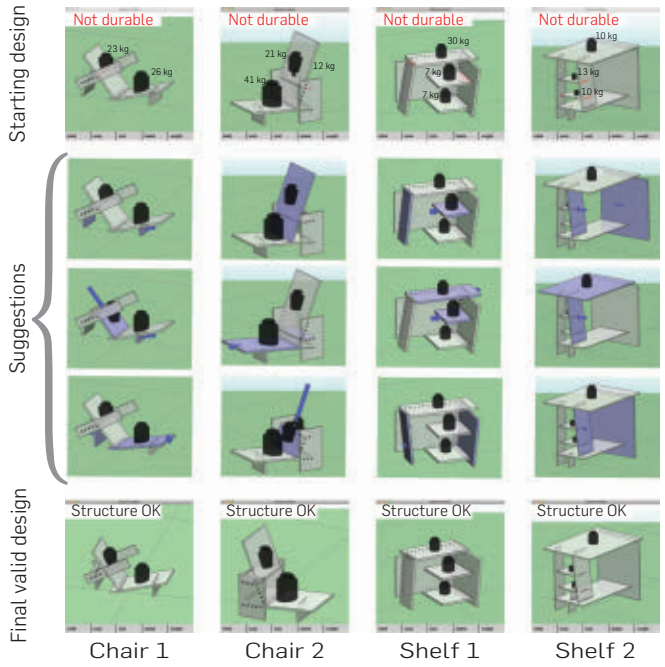


Figure 14. Typical nail-jointed furniture design sessions in our guided exploration framework. Only a few suggestions are shown in each example. Please refer to the accompanying video and demo for further details. Note that suggestions often involve synchronous manipulation of multiple planks, which is difficult to achieve without computational support.



adjust a *single* parameter along the suggested deformation direction. For example, with chair 1, we show three different suggestions, each involving a pair of planks to be simultaneously manipulated to restore validity. In the case of chair 2, the situation is similar, except we have three specified weights.

In the case of shelf 1, note that the geometry of the initial and final configurations is similar. However, it is non-trivial to determine the validity-restoring path using trial-and-error, particularly since there are different interactions involving simultaneous rotation and anisotropic scaling of multiple components. In the case of shelf 2, the top and the large side planks are adjusted multiple times over the course of the guided exploration to result in a shape that can withstand the three vertical loads. Note that the complexity of the configuration space rapidly grows with the number of planks, making it increasingly difficult to design valid shapes manually without computational support and guidance. Table 1 lists typical continuous and discrete times for generating suggestions. To generate suggestions, we must explore $O(N_\gamma^3)$ directions in real time, even for 15–20 planks using a linear approximation, with the line search step accounting for the majority of the computational time. We recall that each additional plank results in an increase in $N\gamma$ of roughly 8 (see Section 6.1).

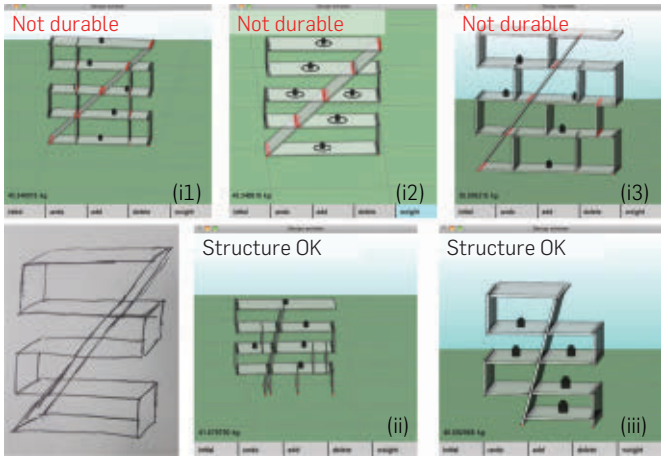
Table 1. Performance statistics on a computer with an Intel Core™ i7 2.8GHz processor with 4GB of RAM.

	Figure 14-right	Figure 13-right	Figure 13-middle	Figure 1-right
#planks	9	10	20	28
#joints	13	13	33	49
#continuous suggestion	8	8	8	6
Candidate generation (ms)	13.2	22.3	160	758
Line search (ms)	92.3	83.1	670	1512
#discrete suggestion	1	1	2	1
Discrete suggestion (ms)	3.8	5.6	48	52
Total time (ms)	110	123	880	2420

User study. We carried out a user study to obtain feedback from the participants. Recreating a typical design scenario, we asked the user to design a piece of furniture freely, following a concept of their own. Nine test users (all novice designers, computer science graduates, and with one female user) designed furniture with three types of system: (i) one without feedback from the physical simulation (i.e., the participants were not informed as to which joints were durable or whether or not the furniture was stable); (ii) one with feedback from the simulation (validity check and valid range visualization) but without suggestions; and (iii) one with feedback and suggestions (i.e., the system described here). While using system (i), the user was able to see the results of the simulation up to five times whenever they liked, assuming the use of conventional shape-modeling software along with simulation software. Each participant began by creating a concept design on paper. Then the participant created 3D furniture models with the three systems to realize their concept design. To counterbalance learning effects, we separated the nine participants into two groups: five participants used the system in the order (i, ii, iii), while the remainder used the system in the reverse order (i.e., iii, ii, i). On average, the participants took roughly 30 min per successful design.

In Figure 15, we show a session where the participant, who was proficient at using Google SketchUp, used the systems in order (i, ii, iii). The participant simply failed to design a valid shape using system (i). With system (ii), he managed to design a valid piece of furniture, but complained that the shape of the furniture was boring and far from his initial design concept. Using system (iii), he successfully designed a valid piece of furniture closely following his initial concept. Other participants fared similarly. All nine participants successfully created valid pieces of furniture that were close to their initial concepts using

Figure 15. Starting from a design concept (bottom-left), three failed attempts with no feedback or suggestions (i1–3), using only feedback without suggestions (ii), and using our system (iii).



our system. Note that even the participants that used the system in the reverse order (iii, ii, i) mostly failed to recreate their initial design using systems (i) or (ii), although they did achieve successful designs using system (iii). The validity-restoring suggestions often involve synchronous editing of multiple parts, which is challenging without suitable computational support. One participant commented that displaying the range of edits was very useful for fine-tuning a design.

8. CONCLUSION AND FUTURE WORK

We have described an interactive computational design framework for guided exploration of physically valid shapes for nail-jointed furniture. Our system provides active real-time guidance to help users avoid invalid designs, due to either stability violations or excessive joint forces. We use a novel force space analysis for both bending forces and contact constraints to generate multiple suggestions, along with respective valid deformation ranges, involving continuous and discrete geometric changes.

We have identified a number of directions for future work. These are summarized here. *Better stress modeling*: we modeled each plank as an unbreakable rigid body; however, to handle flexible planks, it is desirable to accurately model the internal stresses. *Designing dynamic objects*: designs with multiple possible configurations, such as a rocking chair, require an extension of our method; one approach is to treat the problem as a coupled guided exploration with multiple target configurations. *Scalability*: exploring valid designs involving a large number of components is difficult in our system; a possible solution is to reduce the number of DOFs by exploiting the structure and redundancy of the input models and exploring a hierarchical solution space.

Acknowledgements

We thank Lars Hesselgren, Helmut Pottmann, Anthony Steed, and Michael Wand for their comments and useful suggestions. The work was supported in part by a KAUST visiting student grant, the Marie Curie Career Integration Grant 303541, the ERC Starting grant SmartGeometry (StG-2013-335373), JST, and JSPS.

References

1. Alexa, M., Matusik, W. Reliefs as images. *ACM TOG (SIGG.) 29*, 4 (2010), 60:1–60:7.
2. Bergman, R. *Wood Handbook – Wood as an Engineering Material*. Forest Products Laboratory, Madison, WI, 2010.
3. Bickel, B., Bächer, M., Otaduy, M.A., Lee, H.R., Pfister, H., Gross, M., Matusik, W. Design and fabrication of materials with desired deformation behavior. *ACM TOG (SIGG.) 29*, 4 (2010), 63:1–63:10.
4. Chaudhuri, S., Koltun, V. Data-driven suggestions for creativity support in 3D modeling. *ACM TOG (SIGG. Asia) 29*, 6 (2010), 183:1–183:10.
5. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D. Modeling by example. *ACM TOG (SIGG.) 23*, 3 (2004), 652–663.
6. Gal, R., Sorkine, O., Mitra, N.J., Cohen-Or, D. iWIRES: An analyze-and-edit approach to shape manipulation. *ACM TOG (SIGG.) 28*, 3 (2009), 33:1–33:10.
7. Igarashi, T., Hughes, J.F. A suggestive interface for 3D drawing. In *UIST* (2001), 173–181.
8. Kerr, W.B., Pellacini, F. Toward evaluating material design interface paradigms for novice users. *ACM TOG (SIGG.) 29*, 4 (2010), 35:1–35:10.
9. Merrell, P., Schkufza, E., Li, Z., Agrawala, M., Koltun, V. Interactive furniture layout using interior design guidelines. *ACM TOG (SIGG.) 30*, 4 (2011), 87:1–87:10.
10. Mitra, N.J., Pauly, M. Shadow art. *ACM TOG (SIGG. Asia) 28*, 5 (2009), 156:1–156:7.
11. Ovsjanikov, M., Li, W., Guibas, L., Mitra, N.J. Exploration of continuous variability in collections of 3D shapes. *ACM TOG (SIGG.) 30*, 4 (2011), 33:1–33:10.
12. Paczkowski, P., Kim, M.H., Morvan, Y., Dorsey, J., Rushmeier, H., O'Sullivan, C. Insitu: Sketching architectural designs in context. *ACM TOG (SIGG. Asia) 30*, 6 (2011), 182:1–182:10.
13. Parker, H., Ambrose, J. *Simplified Design of Wood Structures*. Wiley, Hoboken, NJ, 1997.
14. Popović, J., Seitz, S.M., Erdmann, M., Popović, Z., Witkin, A. Interactive manipulation of rigid body simulations. In *ACM SIGGRAPH* (2000), 209–217.
15. Shapira, L., Shamir, A., Cohen-Or, D. Image appearance exploration by model-based navigation. *CGF (EUROGRAPHICS)* (2009), 629–638.
16. Smith, J., Hodgins, J.K., Oppenheim, I., Witkin, A. Creating models of truss structures with optimization. *ACM TOG (SIGG.) 21*, 3 (2002), 295–301.
17. Talton, J.O., Gibson, D., Yang, L., Hanrahan, P., Koltun, V. Exploratory modeling with collaborative design spaces. *ACM TOG (SIGG. Asia) 28*, 5 (2009), 167:1–167:10.
18. Twigg, C.D., James, D.L. Backwards steps in rigid body simulation. *ACM TOG (SIGG.) 7*, 3 (2008), 25:1–25:10.
19. Umetani, N., Igarashi, T., Mitra, N.J. Guided exploration of physically valid shapes for furniture design. *ACM TOG (SIGG.) 31*, 4 (2012), 86:1–86:11.
20. Umetani, N., Kaufman, D.M., Igarashi, T., Grinspun, E. Sensitive couture for interactive garment modeling and editing. *ACM TOG (SIGG.) 30*, 4 (2011), 90:1–90:12.
21. Whiting, E., Ochsendorf, J., Durand, F. Procedural modeling of structurally-sound masonry buildings. *ACM TOG (SIGG. Asia) 28*, 5 (2009).
22. Yang, Y.-L., Yang, Y.-J., Pottmann, H., Mitra, N.J. Shape space exploration of constrained meshes. *ACM TOG (SIGG. Asia) 30*, 6 (2011), 124:1–124:12.
23. Yu, L.-F., Yeung, S.-K., Tang, C.-K., Terzopoulos, D., Chan, T.F., Osher, S.J. Make it home: Automatic optimization of furniture arrangement. *ACM TOG (SIGG.) 30*, 4 (2011), 86:1–86:12.

Nobuyuki Umentani and Takeo Igarashi, Department of Computer Science, Graduate School of information Science and Technology, The University of Tokyo/Japan Science and Technology Agency (JST), Exploratory Research for Advanced Technology (ERATO), Tokyo, Japan.

Niloy J. Mitra, Department of Computer Science, University College London (UCL), London, United Kingdom.

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.

California State Polytechnic University, Pomona
Computer Science Department
http://www.cpp.edu/~cs/

The Computer Science Department invites applications for **three tenure-track positions at the rank of Assistant Professor** to begin Fall 2016. We are particularly interested in candidates with specialization in Software Engineering, Cloud Computing, Databases and Data Mining, CS Education, Computer Graphics, Robotics, Artificial Intelligence and Machine Learning. Strong candidates from other areas are also encouraged to apply. Cal Poly Pomona is 30 miles east of L.A. and is one of 23 campuses in the California State University. The department offers an ABET-accredited B.S. program and an M.S. program. Qualifications: Possess, or complete by September 1, 2016, a Ph.D. in Computer Science or closely related area. Demonstrate strong communication skills, commitment to actively engage in the teaching, research, and curricular development activities of the department at both undergraduate and graduate levels, and ability to work with a diverse student body and multicultural constituencies. Ability to teach a broad range of courses, and to articulate complex subject matter to students at all educational levels. First consideration will be given to completed applications received no later than November 13, 2015. Contact: Faculty Search Committee, Computer Science Department, Cal Poly Pomona, Pomona, CA 91768. Email: cs@cpp.edu. Cal Poly Pomona is an Equal Opportunity, Affirmative Action Employer. Position announcement available at: http://www.cpp.edu/~faculty-affairs/open-positions/index.shtml. Lawful authorization to work in US required for hiring.

California State University, East Bay (Hayward, CA)
Department of Computer Science
Faculty Position in Computer Science

2 POSITIONS (OAA Position No 15-16 CS-DATA/CLOUD/CORE-TT) The Department invites applications for **2 tenure-track appointments as Assis-**

University of Glasgow

University of Glasgow
College of Science and Engineering
School of Computing Science

Lecturer/Senior Lecturer

Vacancy Ref: 010852
Lecturer Grade 7/8 Salary Range £33,242-£37,394/£40,847-£47,328
Senior Lecturer Grade 9 Salary Range £48,743-£54,841

The post holders will pursue a world-class research programme in Computing Science and will teach related topics in Computing Science at all undergraduate and postgraduate levels as well as carrying out allocated administrative tasks.

The School seeks applications for four academic posts from outstanding candidates to develop and lead research of international standard in one or more of our priority areas including ***algorithms and complexity; applied machine learning; autonomous and socially intelligent robotic systems; and machine learning in computational biology.***

You will have a minimum of 2 years postdoctoral research experience and a developing track record of publication in leading international conferences and journals. In addition, you will have the ability, drive and organisational skills to secure research grant funding, lead a research team and develop an international research profile as well as the ability to teach effectively at undergraduate and postgraduate levels.

For appointment at Senior Lecturer you will typically have 5–7 years postdoctoral research experience with a proven track record of undertaking a sustained programme of research of international standard in Computing Science and an established record of gaining research grant funding to support the long-term growth of the activity.

Further information can be found at:
www.glasgow.ac.uk/computing/worldchangerswelcome

Informal enquires may be made to

Professor Chris Johnson
Head of School
Email: christopher.johnson@glasgow.ac.uk
Tel: +44 141 330 6053

Apply online at www.glasgow.ac.uk/jobs

Closing date: 30 September 2015.
Interviews are expected to be held in the period 12–17 November 2015.

The University has recently been awarded the Athena SWAN Institutional Bronze Award.

The University is committed to equality of opportunity in employment.

The University of Glasgow, charity number SC004401.

Athena SWAN Bronze Award

www.glasgow.ac.uk

tant Professor in Computer Science (considering all areas of computer science, capable of teaching in emerging areas) starting Fall 2016. For details see http://www20.csueastbay.edu/about/career-opportunities/. For questions, email: cssearch@mcs.csueastbay.edu.

Stanford University
Graduate School of Business
Faculty Positions in Operations, Information and Technology

The Operations, Information and Technology (OIT) area at the Graduate School of Business, Stanford University, is seeking qualified applicants for full-time, tenure-track positions, starting September 1, 2016. All ranks and relevant disciplines will be considered. Applicants are considered in all areas of Operations, Information and Technology (OIT) that are broadly defined to include the analytical and empirical study of technological systems, in which technology, people, and markets interact. It thus includes operations, information systems/technology, and management of technology. Applicants are expected to have rigorous training in management science, engineering, computer science, economics, and/or statistical modeling methodologies. The appointed will be expected to do innovative research in the OIT field, to participate in the school's PhD program, and to teach both required and elective courses in the MBA program. Junior applicants should have or expect to complete a PhD by September 1, 2016.

Applicants should submit their applications electronically by visiting the web site http://www.gsb.stanford.edu/recruiting and uploading their curriculum vitae, research papers and publications, and teaching evaluations, if applicable, on that site. **For an application to be considered complete, all applicants must have three letters of recommendation, CV and job market paper submitted by November 15, 2015.** For questions regarding the application process, please send an email to Faculty_Recruiter@gsb.stanford.edu.

Stanford University is an equal opportunity employer and is committed to increasing the diversity of its faculty. It welcomes nominations of and applications from women, members of minority groups, protected veterans and individuals with disabilities, as well as from others who would bring additional dimensions to the university's research, teaching and clinical missions.

The College of St. Scholastica
Computer Information Systems Faculty

The College of St. Scholastica seeks a tenure-track faculty member to teach undergrad courses, advise students, participate in the continued development of our curriculum, and actively participate in the other faculty responsibilities.

A graduate-level degree in a computer related field is required, as is excellent teaching and communication skills. Demonstrated cross-cultural communication skills and multicultural competency is necessary. A Ph.D. in a computer related field is preferred but not required. For more information and to apply, visit www.csshrjobs.com. EOE

[CONTINUED FROM P. 128] sity of California at) Davis—and I proposed an application called identity-based encryption.

Thanks to identity-based encryption, any arbitrary string, such as an email address, can function as a user's public key.

Identity-based encryption is just one application. Our work, and the work of others, kicked off a flood of results: shorter digital signatures, ways to search on encrypted data, content protection, privacy mechanisms, and so on. More and more applications are discovered every day.

Your later work expanded the possibilities even further by describing a generalization of pairings called a multilinear map.

In a 2003 paper with Alice Silverberg, we suggested that if one could construct a multilinear map, which is a generalization of pairings, then it would open the door to many new applications. For many years, the challenge was to construct multilinear maps. I'm thrilled to say that a few years ago, Sanjam Garg, Craig Gentry, and Shai Halevi came up with the first candidate multilinear map that lets us to do many of the things we had hoped for and much more.

Such as?

One of the most beautiful applications of multilinear maps is the work of a group of researchers at IBM, UCLA (the University of California at Los Angeles), and U.T. (the University of Texas at) Austin, who showed how to use them for software obfuscation. Software obfuscation lets you hide secrets in code: I can embed a secret key in software and then give you the software. You can look at the code and run it, but you will never be able to get the key out of the code.

In addition to your theoretical work, you've been involved with a number of very practical projects in computer security.

I've always been interested in the bigger picture of computer and data security. What I like to do with my students is identify new ways of defending systems against attacks that have not been thought of yet. So we try to predict what tomorrow's attacks will look like and how we can defend against them. It's fun for me, it's fun for the students, and it's a great way to teach security—see if you can break this thing, and then we all fix it.

"Another way to think of a MOOC is as a 21st-century textbook; it does not replace on-campus learning, but enhances it."

One recent paper of yours demonstrated a way to turn the power meter on a cellphone into a GPS.

Cellphones recently added a power meter so that applications can see how much power is being used and adapt how much they draw. But it turns out the amount of power your cellphone's radio drains depends on obstacles between it and the base station.

So you can calculate the phone's location and route, provided you know the configuration of nearby cellular towers.

Exactly. The software systems we build these days are so complex it's very difficult to reason about all the possible ways things can go wrong. You add a power meter and all of a sudden, you realize it can be used as a GPS.

Let's talk about some of your educational initiatives. You run a popular MOOC (massive open online course) on cryptography, for instance.

If you build an application that touches user data, you're going to have to use cryptography to protect it. Cryptography, however, is one of those fields where a little knowledge can be dangerous—the application may work fine, but if the crypto is used incorrectly, the data will be insecure. We want every CS major to be exposed to some crypto so they know what tools exist, what they do, and more importantly, how to use them correctly. I teach an undergraduate course that takes a hands-on look at how to use crypto to protect real data, and I've made that class into a free online course, a MOOC.

More than 600,000 have signed up for it.

It's been very rewarding. Another way to think of a MOOC is as a 21st-century textbook; it does not replace on-campus learning, but enhances it. My on-campus teaching has improved as a result of the MOOC. In the past, I covered every topic at the same level of detail; now, if there's one topic I think is not as exciting as another, I'll discuss it quickly in class and refer students to the MOOC for more information. This allows me to spend more time in class on deeper topics that deserve more attention, rather than spend lecture time on mundane topics.

You also run a popular computer security class.

It's important to teach CS students to always think about security as they write software, because the code they write will come under attack. We give students assignments where the first part is to break something by finding flaws; the second part is to fix all the vulnerabilities. The students enjoy the first part, but complain about the second; they fix all the bugs they can find, but they worry about losing points because of some bug they didn't find. Well, that's the real world. C

Leah Hoffmann is a technology writer based in Piermont, NY.

DOI:10.1145/2800615

Leah Hoffmann

Q&A
A Passion for Pairings

Dan Boneh on pairing-based cryptography, multilinear maps, and how an 1,800-year-old "intellectual curiosity" became the foundation of all secure network traffic.

DAN BONEH, RECIPIENT of this year's ACM-Infosys Foundation Award, discovered his passion for computers early; by the time he got to Princeton University, where he earned his Ph.D., he had zeroed in on his line of research: cryptography. Now a professor at Stanford University—and head of the university's Applied Cryptography Group—Boneh has made groundbreaking contributions to the field, pioneering the use of pairings to solve cryptographic problems and working on a range of broader computer security questions. Here, he discusses his work.

What drew you to computer science?

I fell in love with computers at a very early age. In high school, I learned about the mathematics of cryptography and was completely taken by it. It is a beautiful area involving elegant mathematics, and at the same time, the results are practical and used in real-world systems.

Can you take us through your work in pairing-based cryptography, which is among your best-known contributions to the field?

Let's start with some context. The story begins 1,800 years ago with a mathematician named Diophantus, who lived in Alexandria and whose work gave rise to many things we learn in high school algebra. The ideas he developed to solve one particular problem described something that is heavily studied in mathematics and called an elliptic curve.

"Cryptography is a beautiful area involving elegant mathematics, and at the same time, the results are practical and used in real-world systems."

At first, the work was theoretical, but eventually, researchers discovered that elliptic curves are extremely useful for cryptography.

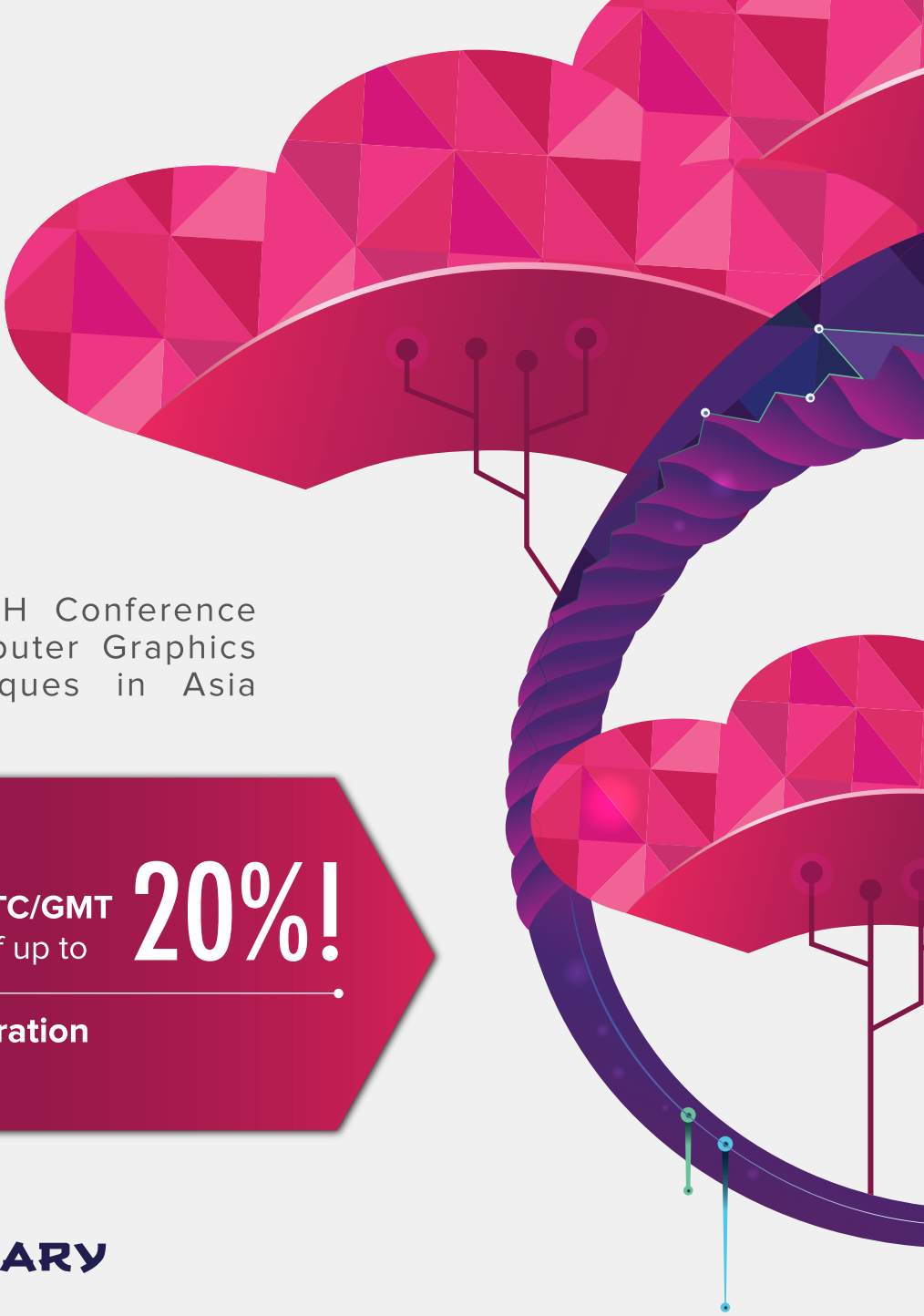
If a browser wants to talk to a bank, it needs to exchange keys to encrypt the communication. Whitfield Diffie and Martin Hellman, working at Stanford, invented an ingenious way to do this. The problem is that to make their method secure, you have to make the numbers involved quite large, and that makes it slow.

So in the 1980s, mathematicians Victor Miller and Neal Koblitz suggested using elliptic curves to enable the parameters to be smaller and the key exchange to run faster. And now every time you connect to Google, you are using this modified version of the Diffie-Hellman exchange. It is a remarkable success story of mathematics—something that for 1,800 years has only been studied as an intellectual curiosity suddenly forms the underpinning of all secure network traffic.

Where does your own work come in?

Mathematician André Weil, who studied elliptic curves in the 1960s, discovered something called a pairing, as a side tool in one of his proofs. As it turns out, pairings are a godsend for cryptographers. We already use elliptic curves for key exchange. Now it turns out they have this additional structure, namely pairings, that enables new applications and this was recognized by several researchers. In 2001, Matt Franklin—who is a professor at UC (the Univer- [CONTINUED ON P. 127]

IMAGE COURTESY OF DAN BONEH



SIGGRAPH ASIA 2015 KOBE

The 8th ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia

Register online before
18 September 2015, 15:00 UTC/GMT
& enjoy early bird discounts of up to **20%!**

sa2015.siggraph.org/registration

[RE]VOLUTIONARY

CONFERENCE 2 - 5 November 2015
EXHIBITION 3 - 5 November 2015
KOBE CONVENTION CENTER, KOBE, JAPAN

SA2015.SIGGRAPH.ORG

Sponsored by
ACM SIGGRAPH

15th International Conference on MODULARITY

March 14–17, **Málaga**, Spain

Modularity 2016 (formerly AOSD) invites papers for its Research Results and Modularity Visions tracks presenting new research and compelling insights into modularity in information systems, including its nature, forms, mechanisms, consequences, limits, costs, and benefits. Proposals for workshops on the conference-related topics that are novel or of emerging importance are also solicited.

General Chair

Lidia Fuentes
Universidad de Málaga, Spain

Local Organizers

Mónica Pinto
Universidad de Málaga, Spain

Mercedes Amor
Universidad de Málaga, Spain

Research Results Chair

Don Batory
University of Texas, USA

Modularity Visions Chair

Krzysztof Czarnecki
University of Waterloo, Canada

Workshops Chairs

Francisco Ortín
Universidad de Oviedo, Spain

Hidehiko Masuhara
Tokyo Institute of Technology, Japan

Demos Chair

Ruzanna Chitchyan
University of Leicester, UK

Student Events Chair

Tobias Pape
Hasso Plattner Institute, Germany

Web Technology Chair

Nadia Gámez
Universidad de Málaga, Spain

Social Media Chair

Inmaculada Ayala
Universidad de Málaga, Spain

Publicity Chair

José María Conejero
Universidad de Extremadura, Spain

Important dates

Abstracts (recommended):
October 30, 2015 (Fri)

Papers:
November 6, 2015 (Fri)

Workshop Proposals:
October 30, 2015 (Fri)

acm In-Cooperation

acm SIGPLAN SIGSOFT

aosa Aspect-Oriented Software Association

uma UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

http://2016.modularity.info