# TRUST BUT VERIFY: A GUIDE TO ALGORITHMS AND THE LAW

BY

DEVEN R. DESAI* AND JOSHUA A. KROLL†

*The call for algorithmic transparency as a way to manage the power of new data-driven decision-making techniques misunderstands the nature of the processes at issue and underlying technology. Part of the problem is that the term, algorithm, is broad. It encompasses disparate concepts even in mathematics and computer science. Matters worsen in law and policy. Law is driven by a linear, almost Newtonian, view of cause and effect where inputs and defined process lead to clear outputs. In that world, a call for transparency has the potential to work. The reality is quite different. Real computer systems use vast data sets not amenable to disclosure. The rules used to make decisions are often inferred from these data and cannot be readily explained or understood. And at a deep and mathematically provable level, certain things, including the exact behavior of an algorithm, can sometimes not be tested or analyzed. From a technical perspective, current attempts to expose algorithms to the sun will fail to deliver critics' desired results and may create the illusion of clarity in cases where clarity is not possible.*

*At a high-level, the recent calls for algorithmic transparency follow a pattern that this paper seeks to correct. Policy makers and technologists often talk past each other about the realities of technology and the demands of policy. Policy makers may identify good concerns but offer solutions that misunderstand technology. This misunderstanding can lead to calls for regulation that make little to no sense to technologists. Technologists often see systems as neutral tools, with uses to be governed only when systems interact with the real world. Both sides think the other simply "does not get it," and*

* Associate Professor of Law and Ethics, Georgia Institute of Technology, Scheller College of Business; J.D., Yale Law School; Affiliated Fellow, Yale Law Information Society Project; former Academic Research Counsel, Google, Inc. I, and this Article, have benefitted from discussions with and input from Solon Barocas, Ariel Feldman, Brett Frischmann, Andrew Selbst, and Peter Swire. I am grateful for feedback on an early draft from participants at the Law and Ethics of Big Data Research Colloquium hosted by The Department of Law and Ethics, Kelley School of Business, University of Indiana and the Virginia Tech Center for Business Intelligence Analytics, and from participants at Privacy Law Scholars 2016 hosted by George Washington University Law School and UC Berkeley School of Law. This Article was supported in part by summer research funding for me from the Scheller College of Business and an unrestricted gift to the Georgia Tech Research Institute by Google, Inc. The views expressed herein are those of the authors alone and do not necessarily reflect the view of those who helped with and supported this work.

† Systems Engineer, Cloudflare, Inc., PhD, Princeton University; Affiliate, Center for Information Technology Policy, Princeton.

Electronic copy available at: https://ssrn.com/abstract=2959472

*important problems receive little attention from either group. By setting out the core concerns over the use of algorithms, offering a primer on the nature of algorithms, and a guide on the way in which computer scientists deal with the inherent limits of their field, this paper shows that there are coherent ways to manage algorithms and the law.*

Electronic copy available at: https://ssrn.com/abstract=2959472

# TRUST BUT VERIFY:
# A GUIDE TO ALGORITHMS AND THE LAW

BY
DEVEN R. DESAI AND JOSHUA A. KROLL

> *According to my definition, a number is computable if its decimal can be written down by a machine.*[1]
>
> *--Alan Turing*

> *The next time you hear someone talking about algorithms, replace the term with "God" and ask yourself if the meaning changes. Our supposedly algorithmic culture is not a material phenomenon so much as a devotional one, a supplication made to the computers people have allowed to replace gods in their minds, even as they simultaneously claim that science has made us impervious to religion.*[2]
>
> *--Ian Bogost*

# INTRODUCTION

Someone is denied a job.[3] A family can't get a loan for a car or a house.[4] Someone else is put on a no-fly list.[5] A single mother is denied federal benefits.[6] None of these people has an idea or reason why that happened other than the decision was processed through some software.[7] Someone commandeers a car, controls its brakes, or even drives away with the car.[8] A car company claims its cars have low emissions, but in

[1] Alan Mathison Turing, *On computable numbers, with an application to the Entscheidungsproblem*, 42 PROCEEDINGS OF THE LONDON MATHEMATICAL SOCIETY, 230, 230 (1936).
[2] Ian Bogost, *The Cathedral of Computation*, THE ATLANTIC, January 15, 2015, http://www.theatlantic.com/technology/archive/2015/01/the-cathedral-of-computation/384300/
[3] *See e.g.* FRANK PASQUALE, THE BLACK BOX SOCIETY: THE SECRET ALGORITHMS THAT CONTROL MONEY AND INFORMATION 34-35 (2015) (describing use of software and online data to make hiring decisions).
[4] *Id*. at 4-5 (discussing use of predictive analytics in credit scoring and loan decisions).
[5] Danielle Citron, *Technological Due Process*, 85 WASH. U. L. REV. 1249, 1256-57 (2008).
[6] *Id*.
[7] *See e.g.*, PASQUALE, *supra* 3 at 4-5 (2015) (one "will never understand exactly how [one's credit score] was calculated"); *infra* notes __ to __ (33-36) accompanying text.
[8] At least two groups have shown ways to take over a Tesla and open its doors, open its sunroof, and enable keyless driving so the car could be driven away, that is stolen. *See* Davis Z. Morris, *Tesla Stealing Hack Is About Much More than Tesla*, FOTUNE.COM, November 26, 2016 at http://fortune.com/2016/11/26/tesla-stealing-hack/. In one case, a group was able to take over the braking system and more from a great distance. *See* Andrea Peterson, *Reseachers Remotely Hack Tesla S*, WASHINGTON POST, September 20, 2016.

fact its cars pollute.[9] A voting machine is supposed to count votes accurately, but no one can tell whether the count is correct.[10] An electric car's battery seems not to have sufficient capacity, so its software is updated, but no one knows whether the update has fixed the problem or is compliant with government safety regulations.[11] Searches for black sounding names yield ads suggestive of arrest records.[12] For each decision, it is difficult both to learn precisely why these things happen and to determine whether the actors using the software—companies or government agencies—have complied with regulations or have committed fraud by only feigning compliance. A common concern is that the software behind the decision or process is at fault for any negative outcomes, either because it has bugs or because the algorithms, or methods for determining outcomes, are incorrect or faulty.[13] And often critics claim that those who are affected by such decisions can't understand or govern these outcomes, because the decision process is a black box.[14] The standard solution to this general problem is a call for transparency, and in this context specifically a call for what critics have called algorithmic transparency.[15] We argue that although the problems are real, for important computer

[9] *See e.g.*, Russel Hotten, *Volkswagen: The Scandal Explained*, December 10, 2015 BBC NEWS, (explaining the way Volkswagen used software to fake emissions results) at http://www.bbc.com/news/business-34324772; Alex Davies, *Here We Ago Again: EPA Accuses Chrysler of Selling Dirty Diesels*, January 12, 2017 WIRED.COM (noting EPA accused Fiat Chrysler of installing and not disclosing software that hid nitrous oxide emissions in its diesel cars) at https://www.wired.com/2017/01/epa-now-accusing-fiat-chrysler-selling-dirty-diesels/.

[10] *See e.g.*, J. Alex Halderman, *Want To Know If the Election Was Hacked? Look at the Ballots*, Nov. 23, 2016, MEDIUM.COM, at https://medium.com/@jhalderm/want-to-know-if-the-election-was-hacked-look-at-the-ballots-c61a6113b0ba#.gzpyt1dat

[11] *Cf.* Alex Davies, *Tesla's Plans To Kill Range Anxiety With A Software Update*, WIRED.COM, March 19, 2015 at https://www.wired.com/2015/03/teslas-plan-kill-range-anxiety/.

[12] *See e.g.*, Latanya Sweeney, *Discrimination in Online Ad Delivery*, 56 COMMUNICATIONS OF THE ACM 44, 52 (2013) ("These findings reject the hypothesis that no difference exists in the delivery of ads suggestive of an arrest record based on searches of racially associated names.").

[13] *See infra* Part I.

[14] *See* PASQUALE, *supra* note 3, at 165.

[15] S*ee e.g.*, Katherine Noyes, *The FTC Is Worried About Algorithmic Transparency, And You Should Be Too*, PC WORLD (April 9, 2015 8:36 AM) http://www.pcworld.com/article/2908372/the-ftc-is-worried-about-algorithmic-transparency-and-you-should-be-too.html (noting Christian Sandvig's view that

science reasons, this proposed solution will not work. Nonetheless there is, and we offer, a way to mitigate these problems so that society can continue to benefit from software innovations.

Put simply, current calls for algorithmic transparency misunderstand the nature of computer systems. This misunderstanding may flow in part from the religious, devotional culture around algorithms Ian Bogost describes. Both critics and advocates can stray into uncritical deference to the idea that the big data, mathematical models and algorithms used to make decisions in software are somehow infallibly scientific. We believe this problem is aggravated because, although algorithms are decidedly *not* mystical things or dark magic, the details of how software systems work are not well understood outside the technical community.[16] This paper thus examines the idea of algorithmic transparency, offers a primer on the construction and analysis of software as a way to bridge this gap, and presents concrete options for managing the problems automated decision-making presents to society.

Those who wish to rein in certain sectors' power by forcing transparency raise good questions about the structure of our society, fairness, and welfare, but as applied to algorithms and automated decision-making, a call for transparency alone is misguided. A consistent theme is that unaccountable machines have taken center stage and now "are used to make decisions for us, about us, or with us," in sensitive and subjective areas

transparency may not be viable because of the complexity of some algorithms and the data needed to test the algorithms); *accord* Christian Sandvig, Kevin Hamilton, Karrie Karahalios, and Cedric Langbort, *Auditing algorithms: Research methods for detecting discrimination on internet platforms* (2014) (using the "social scientific study" auditing to investigate algorithmically driven platforms).

[16] Bogost, *supra* note 2 ("The next time you hear someone talking about algorithms, replace the term with "God" and ask yourself if the meaning changes. Our supposedly algorithmic culture is not a material phenomenon so much as a devotional one."); *see also* JOSHUA A. KROLL, ACCOUNTABLE ALGORITHMS 2, n.1 (2015) ("The term 'algorithm' is assigned disparate technical meaning in the literatures of computer science and other fields.") available at https://www.cs.princeton.edu/~kroll/papers/dissertation.pdf.

such as health-care, employment, credit, national security, networked devices, news, and more.[17] Even before today's fascination with big data, algorithms, and automated systems legal scholars such as Paul Schwartz and Danielle Citron have noted that data processing and software used in the administrative state can undermine or take away due process rights.[18] A related fear is that the human designer of a program could have bad intent and seek to discriminate, suppress speech, or engage in some other prohibited act.[19] A more recent fear is that the rise of large data sets combined with machine learning (an area of computer science that uses the automated discovery of correlations and patterns to define decision policies that are not explicitly determined by humans) means that those who use such techniques may be able to wield power in ways prohibited by law or disfavored politically, but which would not be detectable.[20] Further, if software yields undesired results, its programmers may say that the system was not designed to act that way.

Transparency has been proposed as a solution to mitigating these possible outcomes. The claim is that someone "ought to be able to 'look under the hood' of highly advanced technologies like [] algorithms" as a way to police such behavior.[21] There are two interpretations of this position that raise different questions. On the one hand we need to know that a system, such as one for counting votes or allocating visas in a lottery, is doing what it is supposed to do and that there is a meaningful way to look under the

[17] *See e.g.,* Centre for Internet & Human Rights at European University Viadrina, Final Draft Background Paper, *The Ethics of Algorithms: From Radical Content to Self-Driving Cars*; PASQUALE, *supra* note 3, at 4 (2015); *cf*. Bogost, *supra* note 2 (explaining deference to algorithms resembles idolatry rather than following Enlightenment skepticism).
[18] *See* Paul Schwartz, *Data Processing and Government Administration: The Failure of the American Legal Response to the Computer*, 43 HASTINGS L.J. 1321 (1991); Citron, *supra* note 5, at 1249.
[19] *See infra* Part I.
[20] *See e.g.,* FTC REPORT, BIG DATA A TOOL FOR INCLUSION OR EXCLUSION: UNDERSTANDING THE ISSUES, (January 2016); Centre for Internet & Human Rights at European University Viadrina *supra* note 17; PASQUALE, *supra* note 3, at 4; Solon Barocas & Andrew D. Selbst, *Big Data's Disparate Impact*, 104 CALIF. L. REV. 671 (2016).
[21] PASQUALE, *supra* note 3, at 165; *but see* Noyes, *supra* note 15.

hood.[22] On the other hand, society may wish to ferret out undesired and possibly unintended results such as the use of an algorithm to discriminate in a hiring decision. Thus, critics audit parts of systems, decry apparent discrimination, and want to hold someone responsible for disfavored outcomes. This approach is tantamount to saying we need proof that the algorithm is not designed to engage in, nor has parts of it that lead to, discrimination or other undesired or prohibited acts.[23]

Both views seek a type of transparency. And both views relate to a deep, unstated and powerful view in the law; the builder of the proverbial better mousetrap will know precisely how it was built and what will happen when one presses a trigger or button in the invention. The device will do the same thing over and over until the springs wear out. The same, so reasons the law, must be true of software. As we shall see, while in many ways, it is, in some very important ways, it is not.

This view relates to more than just the machines used to make or aid in making decisions. Entire decision-making processes fit this view, including discretionary and rule-driven decisions whether made by people or by machines, and whether operated by the state or by private entities. As Jerry Mashaw has argued for administrative state systems, they should make "accurate," "cost-effective" judgments, but also give "attention to the dignity of participants."[24] The dignity element requires that those who are subject to such a process know or understand what reasons are behind a decision.[25]

[22] *See e.g.*, Halderman, *supra* note 10.
[23] *See infra* Part I.
[24] *See* JERRY L. MASHAW, BUREAUCRATIC JUSTICE 26, 95-96; *accord* Schwartz, *supra* note 18, at 1348.
[25] MASHAW, *supra* note 24, at 175; *accord* Schwartz, *supra* note 18, at 1349. Respecting and protecting dignity is important as a practical matter given the EU's approach to data processing. The current European Data Protection Supervisor—the office responsible "under Article 41.2 of Regulation 45/2001 'With respect to the processing of personal data… for ensuring that the fundamental rights and freedoms of natural persons, and in particular their right to privacy, are respected by the Community institutions and bodies', and '…for advising Community institutions and bodies and data subjects on all matters concerning

Thus, "attention to the dignity of participants" has a transparency dimension, as it relies on the idea that one can see how the mousetrap or system worked and understand it. The problem is that many recent implementations of decision-making in software—the ones that have raised concerns—do not map well to this assumption.

Put differently, transparency is a powerful concept and has its place. After all who can argue against sunlight? And yet to an extent, we will do exactly that, because from a technical perspective general calls to expose algorithms to the sun or to conduct audits will not only fail to deliver critics' desired results but also may create the illusion of clarity in cases where clarity is not possible.[26]

As part of a larger project on algorithms and the law, we argue that demands for transparency must confront the realities of computer science when it comes to testing and analyzing software. For example, because socially important computer systems have a large range of possible inputs and outputs, social science auditing methods can only test "a small subset of those potential inputs."[27] As legal matter, determining whether such methods can capture whether a prohibited practice has occurred, and to an extent, which is actionable, presents problems.

In addition, handing over code often will not yield the accountability results those in favor of so-called algorithmic transparency desire. That does not mean those who use

---

the processing of personal data'—has stated that dignity must be protected as fundamental human right in light today's privacy and personal data processing issues. *See* GIOVANNI BUTTARELLI, TOWARDS A NEW DIGITAL ETHICS: DATA, DIGNITY, AND TECHNOLOGY, Opinion 4/2015 of the European Data Protection Supervisor, September 11, 2015 available at https://secure.edps.europa.eu/EDPSWEB/webdav/site/mySite/shared/Documents/Consultation/Opinions/2015/15-09-11_Data_Ethics_EN.pdf.

[26] *Cf*. Jatinder Singh, Ian Walden, Jon Crowcroft, Jean Bacon, *Responsibility & Machine Learning: Part of a Process*, (October 27, 2016) available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2860048 ("algorithmic selection not only impacts the quality of the ML model, but the degree to which the inner workings of the ML algorithm and learned model can be interpreted and controlled depends on the technique used.").

[27] *See* Joshua A. Kroll et. al., *Accountable Algorithms*, 165 U. PENN. L. REV. 633 (2016), available at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2765268.

computerized decision-making are ungovernable, but it does require that we understand what is and is not possible when we seek to regulate or monitor the use of these technologies.[28] Many of the current calls for transparency as a way to regulate automation do not address these limits, and so they may come up short on providing the sort of accountability they desire, and which we also support.[29] Instead, as software continues to grow in importance, especially when it makes use of machine learning, which separates the creation of algorithms and rules from human design and implementation, we argue that identifying harms, prohibiting outcomes, and banning undesirable uses of data or technologies are more promising paths.[30] In addition, in some cases, requirements that software be built to certain specifications that can be tested or verified will also be necessary. We believe that these restrictions can be effectively promulgated in the law, creating legal regulation of automated decision-making systems that is more powerful than transparency requirements alone.

In contrast to the current approaches to governance by auditing to find unacceptable behaviors or demanding algorithmic transparency, regulation via the law will realize four benefits from being informed by the way software and algorithms are tested and analyzed for correctness. First, legal regulation can avoid the problem of applying inapt approaches from past regulatory regimes or demanding outcomes that are not possible. Second, it can address the dynamism of the industry and the difficulties of

[28] *See infra* Part II and III.
[29] As one group of computer scientists has noted within machine learning "some algorithms are more amenable to meaningful inspection and management than others." Singh et. al., *supra* note 26 (offering that decision trees, naïve Bayes, and rule learners were the most interpretable, k-Nearest Neighbors (kNN) was in the middle, and neural networks and support vector machines were the least interpretable).
[30] *Cf.* FTC REPORT, *supra* note 20 (acknowledging potential beneficial and negative outcomes from using data analytics and noting that it is the use of data and data analytics in certain areas such as housing, credit, and employment that triggers concerns and potential liability, not the use of application of data analytics alone).

analyzing software by providing requirements that technologists and computer scientists understand and can implement. Third, as with past regulation of housing, credit, and employment, legal regulation of software and algorithms can offer clarity about what is actionable and when, as well as what, evidence must be offered to regulators to show compliance. Fourth, if those who are to be regulated object, the burden will be on them to show why proposed technically-informed solutions don't work. And that discussion will use the framework and terms within which they already operate, avoiding charges that the law creates unachievable mandates. As such, it should be less likely that objections based on feasibility will succeed. In short, smart regulation via the law allows the many gains from automation to be captured safely, while providing the assurances of governance necessary to assuage critics.

We begin with a discussion of the law and policy concerns over software systems that have been raised so far and some of the proposed approaches to addressing these concerns. This discussion shows that there are many different issues at play, and many of those issues are proxies for concerns about power and inequality in general, not software specifically. After setting out an understanding of the claimed problems, we turn to some fundamental questions about computer science such as what an algorithm is and whether policy can be general enough to cover all software in the same way.[31] Having set out a brief primer on the underlying computer science, we turn to the question of determining what a piece of software will do when it is run. It turns out that it is impossible to determine this reliably and for all programs. With that in mind, we turn to the way in which computer scientists have addressed this problem. Using that foundation, we offer recommendations on how to regulate public and private sector uses of software and

[31] *See e.g.*, Sandvig et al. *supra* note 15 ("virtually any [piece of software] may deserve scrutiny.").

propose a proposed legislative change to protect whistleblowers and allow a public interest cause of action, which will aid in increasing detection of overt misdeeds in designing software. In short, we conclude that a better understanding of how programs work and how computer scientists address the limits of software analysis affords policymakers the tools to manage the evolving world of algorithms and the law so that society can address justice and safety interests while also enabling many actors to use these new techniques to innovate and improve the world in which we live.

## I. ALGORITHMS, THE CONCERNS

Software and algorithms have gained much attention under the *premise* that they "exercise power over us,"[32] because they "[govern selection of] what information is considered most relevant to us, a crucial feature of our participation in public life,"[33] are "powerful entities that govern, judge, sort, regulate, classify, influence, or otherwise discipline the world,"[34] and are "black boxes."[35] In short, the general idea that computer systems are powerful and opaque has led to claims "that virtually any [piece of software] may deserve scrutiny."[36] Varying reports and studies raise concerns about complex systems and point to a more general concern: the possible power and avenues of abuse or

[32] *See e.g.,* Nicholas Diakopoulous, *Algorithmic Accountability Reporting: On the Investigation of Black Boxes* (Tow Center for Digital Journalism, Columbia Unversity 2014) ("What we generally lack as a public is clarity about how algorithms exercise their power over us.").
[33] Tarleton Gillespie, *The Relevance of Algorithms* IN MEDIA TECHNOLOGIES: ESSAYS ON COMMUNICATION, MATERIALITY, AND SOCIETY (Tarleton Gillespie, Pablo J. Boczkowski, and Kirsten A. Foot. eds., 2104).
[34] Solon Barocas, Sophie Hood, and Malte Ziewitz, *Governing algorithms: A provocation piece* (2013) at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2245322.
[35] PASQUALE, *supra* note 3, at 17.
[36] Sandvig et. al, *supra* note 15.

manipulation that go with practices that use computers. Despite their different methods and concerns, many commenters look to transparency as a key part of managing this new order, because we "cannot access critical features of [its] decision-making processes."[37] These issues arise in both the public and private sector context. And yet consensus on what sort of scrutiny is needed, whether different areas affected by computers require different solutions, and whether software, other factors, or both are the cause of the claimed problems, is lacking. We start with some examples of the public sector concerns and then turn to private sector ones.

## A. Algorithms, Public Sector Concerns

Public sector concerns are about power but involve questions on how power is governed that are different from private sector concerns. Public sector use of automated decision-making raises larger questions, because society regulates public power differently than the way it regulates the private sector.[38] Legal scholars have looked at

[37] PASQUALE, *supra* note 3, at 17. The call for or desire to have transparency as a way to resolve issues around automated decision systems can be strong. For example, Professor Latanya Sweeney has done work on racial discrimination and advertising. Although Sweeney's final paper did not invoke transparency, a draft claimed that answering why the advertising discrimination occurred "requires further information about the inner workings of Google AdSense." Latanya Sweeney, *Discrimination in Online Ad Delivery*, (January 28, 2013) preprint available at http://dataprivacylab.org/projects/onlineads/1071-1.pdf. We note this point only to indicate the draw of transparency, not to argue that Professor Sweeney advocates one way or the other on that strategy.

[38] *Cf.* EXECUTIVE OFFICE OF THE PRESIDENT, BIG DATA: SEIZING OPPORTUNITIES, PRESERVING VALUES at 10 (2014) https://www.whitehouse.gov/sites/default/files/docs/big_data_privacy_report_may_1_2014.pdf ("Public trust is required for the proper functioning of government, and governments must be held to a higher standard for the collection and use of personal data than private actors."). That private actors are taking on government functions in many areas is clear. *Cf.* Curtis Publ'g. Co. v. Butts, 388 U.S. 130, 163-64 (1967) (Warren, C.J. concurring) (noting that policy is set by "a complex array of boards, committees, commissions, corporations, and associations, some only loosely connected with the Government" rather

governmental use of computerized decision-making and identified several areas where software can aid the way the administrative state functions but at the same time run afoul of justice and due process requirements.

Twenty-five years ago, Paul Schwartz noted, "Computers are now an integral part of government administration."[39] Given the rise of the administrative state in managing and providing "social services," the state requires "detailed information on the citizen as client, customer, or simply person to be controlled. Moreover, the state gathers personal information to better manage itself."[40] When the state uses data to administer services, we want administration that "carries out legislative policy, acts in a just manner, and combats fraud."[41] Schwartz examined the Aid to Families with Dependent Children Program and Child Support Enforcement programs as exemplars of the administrative state. He argued that the nature of data processing undermined the ability to attain bureaucratic justice as developed by Jerry Mashaw and the ability to protect autonomy.[42] In that vision, the system should not only make "accurate," "cost-effective" judgments, but also give "attention to the dignity of participants."[43] The first two criteria relate to the use of data and data processing in that one needs to show a "connection between a particular decision, given the factual context, and the accomplishment of one or more of the decision maker's

than by "formal political institutions."); Deven R. Desai, *Speech, Citizenry and the Market: A Corporate Public Figure Doctrine* 98 MINN. L. REV. 455, (2013) ("the distinction between commercial and political has collapsed"). But whether a specific private actor (or sector) using a given piece of software is performing a public function must be determined to see whether they should be held to the same standard as the government.

[39] Schwartz, *supra* note 18, at 1322.

[40] *Id*. 1332; *cf*. FRANK WEBSTER, THEORIES OF THE INFORMATION SOCIETY 55 (John Urry ed., 3d ed. 1995) (""[I]f we as a society are going to respect and support the individuality of members, then a requisite may be that we know a great deal about them."); Jack M. Balkin, *The Constitution in the National Surveillance State*, 93 MINN. L. Rev. 1, 18 (2008) (arguing government needs to collect information "to ensure efficient government and national security" but must have "justifiable reasons" and procedures to protect against abuse of data collection and use).

[41] Schwartz, *supra* note 18, at 1333.

[42] *Id.* at 1351.

[43] *Id.* at 1348.

goals."[44] The dignity element requires that those who are subject to such a process know or understand what reasons are behind a decision.[45] Without that knowledge or understanding those subject to the decision-making process lose self-worth, and over time the legitimacy of the system will be in doubt, because of the lack of understanding and loss of dignity.[46]

Danielle Citron's work also calls out the way that computers have been used in the administrative state.[47] She focused on due process concerns.[48] She describes the "automated administrative state"[49] as using software to determine whether someone should receive "Medicaid, food stamp, and welfare" benefits, be on a no fly list, and be identified as owing child support.[50] According to Citron, "Automation jeopardizes the due process safeguards owed individuals and destroys the twentieth-century assumption that policymaking will be channeled through participatory procedures that significantly reduce the risk that an arbitrary rule will be adopted."[51]

Although these scholars use different metrics about why the use of software and computers is a problem, both identify the problem sphere as the administrative state.[52] And both Schwartz and Citron look to transparency, among other tools, as a way to

[44] *Id.*
[45] *Id.* at 1349.
[46] *Id*.
[47] Citron, *supra* note 5, at 1256-57.
[48] *Id*.
[49] *Id.* at 1281.
[50] *Id*. at 1256-57.
[51] *Id*. at 1281.
[52] Work by computer scientists has looked at software and accountability and also found the administrative state as a prime example of where algorithmic governance is needed. *See generally* Kroll et. al., *supra* note 27 (using government visa lottery programs and voting machines as examples where the use of algorithms intersect with the application of specific rules for decision-making that affect individual rights).

determine whether the state uses data and software-based processes in a way that hinders the ability of citizens to know what is happening within the system.[53]

Two other examples, voting machines and the auto industry illustrate a different, but related, public sector concern: verifying that a system is accurate in implementing its goals and works as desired. Voting machines track votes, and so the process in which the machines are used must be accurate about at least four things. First, it must be accurate about whether someone voted. That is, one might try to hijack an election by making it seem like someone, or many people, voted, when in fact they never voted at all. Typically, humans handle this step by asking voters to sign logbooks. The machines are not logging who voted. Second, the process must also verify that the person voting was eligible to vote. Third, the process needs to be accurate about recording for whom an eligible voter cast their ballot. Finally, the process must be accurate about tallying the set of all properly cast votes. Yet the machines and procedures used to make these guarantees are quite susceptible to being subverted to give outputs that are not accurate.[54] In one example, computer scientists showed that they could make a voting machine play a video game, Pac-Man, instead of tallying votes.[55] The point was not that officials would play on the machines or voters would be frustrated or enjoy a prank on election day, but that the machine can be tampered with, contravening the intuition and presumption at law that the

[53] Schwartz, *supra* note 18, at 1375 (calling for "The maintenance of transparent information processing systems"); Citron, *supra* note 5, at 1295 (noting lack of ability for "meaningful review" of rules and system put in place to deliver administrative state services).

[54] It is important to distinguish the very real problem of whether the machines and processes in use can be deliberately subverted from the distinct problem, never observed in the real world, of whether any elections have actually been subverted in this way. The mere fact that the process is subject to corruption is enough to undermine its legitimacy.

[55] *See* Kim Zetter, *Touchscreen E-Voting Machine Reprogrammed to Play Pac-Man*, August 24, 2010, WIRED.COM (two computer scientists "swapped out the machines PCMCIA card—where the voting software is stored—and replaced it with one loaded with Pac-Man. They pulled this off without disturbing the tamper evident seals on the machine.") at https://www.wired.com/2010/08/pac-man/.

machine is specialized for a purpose and cannot be made to do other things. Given this flexibility in the machine's behavior, a way to verify that the system had not been tampered with—or at least that the accuracy requirements described above are met—is vital.

Almost any industry in which devices are regulated and must behave in certain ways raises issues about software and verification. The auto industry provides a good overview of the issues. Cars have had software governing their operation for some time, but as software grows in importance for cars, so does the importance of accountability and analyzability for that software. Automobiles are subject to safety and environmental regulation. Part of that regulation involves knowing that cars work as claimed by automakers and required by law.

Two recent events in the auto industry—one involving Volkswagen, the other Tesla—illustrate the challenge. First, the recent fraud by Volkswagen illustrates how software can aid a company in evading regulations. Volkswagen used software that allowed the company to make its diesel cars seem to have low emissions when in fact the cars did not.[56] The ability to have accountable and analyzable algorithms in this sector would aid in detecting such fraud.[57]

Second, as Tesla and other car makers offer cars that are networked so that software can be updated after the car is purchased, the integrity and inviolability of software increases in importance. An automaker may claim that the software running in a vehicle performs as promised, but regulators will need ways to verify that the software

[56] *See e.g.*, Russel Hotten, *Volkswagen: The Scandal Explained*, December 10, 2015 BBC NEWS, at http://www.bbc.com/news/business-34324772.
[57] *See infra* Part IV.A.3 (explaining how to build accountable and analyzable algorithms).

and logs have not been altered from the version, which has been reviewed.[58] For example, Tesla now updates its cars regularly, claiming that these updates improve performance such as the range its cars can drive on a full battery charge. Regulators must be able to track whether those updates are described accurately to consumers, whether the new code functions as described, and whether that functionality adheres to safety and other kinds of regulations.[59] Furthermore, as self-driving or autonomous cars continue to be put on the road and evolve, regulating their software will be even more important. For example, if there is a standard that requires a self-driving car to obey traffic laws about how long to signal before changing lanes, , what happens when the automaker pushes an update to the fleet? How can regulators be sure that the updated software complies with the standard? Unlike an update to a computer or mobile phone game, the automaker's change affects not only the user but others on the road. The automaker may, in good faith, assert that the update is within the standards already approved, but the regulating agency—and anyone using the roads—needs a way to verify that claim. Further, regulators may want to ensure that only approved, standards-compliant updates can be installed in vehicles already on the road.

## B. Algorithms, Private Sector Concerns

[58] *Id*. (explaining audit logs and verification of such logs).
[59] Issues with cars' software updates and security have been revealed in at least two cases. *See* Morris, *supra* note 8 (reporting that a security group took over a Tesla and opened its doors, opened its sunroof, and enabled keyless driving so the car could be driven away or stolen) at http://fortune.com/2016/11/26/tesla-stealing-hack/; *See* Peterson, *supra* note 8 (describing researchers able to take over the braking system and more remotely).

Although the private sector is regulated differently than the public sector, calls for transparency as it relates to software-based decision-making in the private sector abound. For example, in light of the importance of recent technologies, Frank Pasquale has argued that the code for important software such as Google's search algorithm or a broadband carrier's method for network management "should be transparent to some entity capable of detecting" the potential misdeeds or harms these services may create.[60] In the same vein, other studies and investigations have identified a range of examples where software was part of undesired or troubling outcomes and have called for methods to detect such issues.

An important area of concern is whether certain software is enabling or aggravating illegal discrimination on the basis of a protected attribute such as race or gender. One study by Professor Latanya Sweeney looked at online search and advertising to test whether a search for "racially associated names" returned "ads suggestive of an arrest record."[61] The study rejected the hypothesis "that no difference exists" in the delivery of such ads, because under its method, a search for a "black-identifying first name," yielded an ad for a company that sold public records and included the word "arrest" in the ad text for "a greater percentage of ads … than [a search] for white-

[60] Frank Pasquale, *Beyond Innovation and Competition: The Need for Qualified Transparency in Internet Intermediaries*, 104 NW. U. L. REV. 1, 166 (2010). Faced with the challenges of data processing and computation a quarter century ago, Paul Schwartz argued that a key factor in managing problems from those practices required, "the establishment of a government body capable of studying the effects and implications [of software-based decisions]." Schwartz, *supra* note 18, at 1379. That approach was part of addressing state actions, and the approach looked at transparency as a feature to limit government action and to make the system "open and understandable to the data subject." *Id*. at 1375. The connection between Pasquale and Schwartz is conceptual: Both seek transparency as a way to enable a third party to aid in scrutiny and to aid the ability to challenge a practice.

[61] Sweeney, *supra* note 12, at 52 ("These findings reject the hypothesis that no difference exists in the delivery of ads suggestive of an arrest record based on searches of racially associated names.").

identifying first names." [62] According to Sweeney, this finding intersects with discrimination problems, because when one competes for many things such as "an award, a scholarship, an appointment, a promotion, or a new job … or [is] engaged in any one of hundreds of circumstances for which someone wants to learn more about you," ads appear in online searches.[63] Another study on search, webpage visitation history, and advertising found that when ad preference settings were set to female, a user saw "fewer instances of an ad related to high paying jobs than [when preferences were set] [] to male."[64] The specific ad was for a career coaching service promising to aid someone in obtaining a job that paid more than $200,000 a year.[65] These studies have identified some outcomes that may not meet the legal[66] or "normative" definition of discrimination but

[62] Sweeney, *supra* note 12, at 52.
[63] *Id*. at 44.
[64] Amit Datta, Michael Carl Tschantz, and Anupam Datta, *Automated Experiments on Ad Privacy Settings*, 1 PROCEEDINGS ON PRIVACY ENHANCING TECHNOLOGIES 92, 92 (2015).
[65] *Id*.
[66] As Peter Swire has observed in an initial investigation of online, data-driven marketing, several statutes prohibit discrimination in specific sectors such as lending, housing, and employment. PETER SWIRE, LESSONS FROM FAIR LENDING FOR FAIR MARKETING AND BIG DATA (2014), https://www.ftc.gov/system/files/documents/public_comments/2014/09/00042-92638.pdf. These statutes apply to online practices but how they apply for each sector and which practices within each sector are prohibited is not settled. *Id*. Sweeney's study may not fit into these sectoral approaches as they appear to be about an indirect, yet possibly powerful, way to affect hiring decisions. That is, the ads at issue in Sweeeny's study are not about an employment opportunity; rather they may affect an employer's impression of or decision about someone without being the explicit criteria on which the decision is made. In contrast, the employment ads in the other study fall under Title VII which governs employment ads. Yet, as Swire explains even when an advertisement falls under a statute:

> One important statutory issue, which is a subject for future research, is what would meet the statutory requirement that the advertisement "indicates any preference, limitation, or discrimination" concerning a protected class. For online advertising, this issue will be important for [] advertisement campaigns that narrowly target a specific population. For instance, it will be important to clarify whether and when the Act covers advertisement purchasing decisions that will reach members of a protected class far more or less often than other demographic groups.

Id.

raise questions about "the pervasive structural nature of [] discrimination in society at large."[67]

The studies cannot, however, find one party to blame in part because of the many factors at play.[68] As Sweeney states, "We do not yet know" "why" [this type of discrimination] is "occurring" or whom to blame. The source of the problem could be the ad buyer, the ad seller, or "society."[69] The other study also admitted that it could not assign blame or determine the cause of the outcomes as being from the advertising network (Google), "the advertiser, or complex interactions among them and others."[70] As such, Sweeney turns to technical solutions to address the issues and argues, "we can use the mechanics and legal criteria described [in her paper] to build technology that distinguishes between desirable and undesirable discrimination in ad delivery."[71] The other study offers a tool to allow the ad network and the advertiser "to audit [each] other" to detect undesired ad behaviors.[72] That study suggests that in the future there may be "machine learning algorithms that automatically avoid discriminating against users in

[67] Datta et. al *supra* note 64, at 105. In addition, advertisers and marketers can be deemed credit reporting agencies under the Fair Credit Reporting Act. The FTC has brought claims for violating the Fair Credit Reporting Act against at least two companies that used data profiles from a range of sources for marketing and advertising activities. *See* United States v. Spokeo, Inc., No. 2-12-cv-05001-MMM-SH (C.D. Cal. June 12, 2012), https://www.ftc.gov/sites/default/ files/documents/cases/2012/06/120612spokeoorder.pdf; Instant Checkmate, No. 3:14-cv-00675-H-JMA (S.D. Cal. Apr. 1, 2014), https://www.ftc.gov/system/files/documents/cases/1 40409instantcheckmateorder.pdf. *See also* Press Release, Fed. Trade Comm'n, Spokeo to Pay $800,000 to Settle FTC Charges Company Allegedly Marketed Information to Employers and Recruiters in Violation of FCRA (June 12, 2012), http://www.ftc.gov/news-events/press-releases/2012/06/spokeo-pay-800000-settle-ftccharges-company-allegedly-marketed; Press Release, Fed. Trade Comm'n, Two Data Brokers Settle FTC Charges That They Sold Consumer Data without Complying with Protections Required under the Fair Credit Reporting Act (Apr. 9, 2014), https://www.ftc.gov/news-events/press-releases/2014/04/two-data-brokers-settle-ftc-charges-they-sold-consumer-data.

[68] Datta et. al *supra* note 64, at 105 ("blaming one party may ignore context and correlations that make avoiding such discrimination difficult").

[69] Sweeney, *supra* note 12, at 52.

[70] Datta et. al *supra* note 64, at 105.

[71] Sweeney, *supra* note 12, at 53.

[72] Datta et. al *supra* note 64, at 106.

unacceptable ways and automatically provide transparency to users."[73] Of course, whether these techniques will be convincing to users, will require their own interrogation for correctness, or can provide convincing evidence of non-discrimiation while still serving the purpose of identifying relevant advertisements remain to be seen. Insofar as the techniques are based on machine learning, the irony may be that the techniques will be as inscrutable as the systems they mean to analyze and thus will fall short of providing accountability. And while there is an emerging class of "interpretable" machine learning models meant to provide explanations of their decisions, it is not known whether such models would function effectively in these cases or could provide sufficient evidence of fairness. Regardless, it is important that actors deploying systems of concern (e.g., search engines, advertising networks, and other users of discrimination-prone automated decision making) be able to understand the requirements placed upon them and the workings of tools used to enforce or surface compliance with those requirements. That is, we seek convincing evidence that such systems function without undesirable discrimination, and evidence generated in inscrutable ways can fail to be sufficiently convincing.

Academics are not the only ones to think about normative concerns as applied to software. Journalists have also investigated the use of automation with similar results and conclusions. Rather than investigating questions about ad networks, where several actors are involved and each may or may not be responsible for outcomes, journalists and technologists have looked at personalization of commerce and search features to see whether a single actor's implementation of an algorithm poses problems.

[73] *Id*. at 106.

An investigation by Wall Street Journal reporters found that the e-commerce they examined lends itself to a legal practice known to economists as price discrimination—the practice of trying to match the price for a good or service to specific market segments or people. Several companies "were consistently adjusting prices and displaying different product offers based on a range of characteristics that could be discovered about the user."[74] For example, Staples, Inc., the office supply company, charged different prices for the same item depending on where Staples thought the consumer was.[75] Although the practice of altering prices based on whether a good is online, in-store, and the geography of the shopper is common, it can reinforce inequality if it allows retailers to charge lower prices (or choose to offer certain services only) to those who lived in ZIP codes with higher weighted average income and charge higher prices (or choose not to offer certain services) to those in ZIP codes with lower weighted average income.[76] Even if one accepts the argument that a retailer accounts for different costs at local, physical stores, if the orders are fulfilled and shipped from a central warehouse, costs associated with physical retail stores should not be an issue. Although the outcomes of price discrimination would seem to indicate that inequality could be reinforced, the price discrimination is not illegal in the retail sector. If personalization is used, however, by a credit card or other regulated, financial company to steer people to more expensive financial services based on race, gender, or other protected class status, price

[74] Jennifer Valentino-Devries, Jeremey Singer-Vine, and Ashkan Soltani, *Websites Very Prices, Deals Based on Users' Information*, THE WALL STREET JOURNAL, (DEC. 24. 2012) ("The Journal identified several companies, including Staples, Discover Financial Services, Rosetta Stone Inc. and Home Depot Inc., that [engaged in the activities]") HTTP://WWW.WSJ.COM/NEWS/ARTICLES/SB10001424127887323777204578189391813881534.
[75] *Id*.
[76] *Id*. This situation could occur naturally if the retailer or its competitors had no stores in the lower-income ZIP codes, but charged based on the proximity to its own or its competitors' stores, as Staples did.

discrimination becomes prohibited discrimination,[77] and so public sector regulation will be triggered, and the need to understand the system behind the outcome becomes like our discussion of other regulated industries above.

Another investigation tried to test the autocomplete feature for Google's and Bing's search services to see how each one handled searches for sensitive topics such as "illicit sex" and "violence."[78] At the time of the report, Bing's autocomplete did not offer autocomplete suggestions for "homosexual," and both Bing and Google did not offer autocomplete suggestions for "110 sex-related words."[79] According to the author, this blacklisting raises the specter of censorship, because "we look to algorithms to enforce morality."[80]

This position is puzzling, because whether users of a product should defer to algorithms and/or a company's manual choices about what to blacklist for morality enforcement (assuming they do that) is answered as, "No," by us and by most who discuss the issue.[81] In addition how much anyone truly "look[s] to algorithms to enforce morality" is unclear. Some may believe algorithms should be constructed to provide moral guidance or enforce a given morality. Others claim that moral choices are vested with a system's users and that the system itself should be neutral, allowing all types of use and with moral valences originating with the user. In either case, choices about morality demand certain outcomes from computer systems such as search engines. Such

[77] Swire, *supra* note 66, at 7-8 (discussing Fair Housing Act prohibition on "steering"—the practice of "deliberately guiding loan applicants or potential purchasers toward or away from certain types of loans or geographic areas because of race.").
[78] Nicholas Diakopoulos, *Sex, Violence, and Autocomplete Algorithms*, SLATE (August 12, 2013, 11:43 AM) http://www.slate.com/articles/technology/future_tense/2013/08/words_banned_from_bing_and_google_s_autocomplete_algorithms.html.
[79] *Id*.
[80] *Id*.
[81] *See e.g.*, Bogost, *supra* note 2; Deven R. Desai, *Exploration and Exploitation: An Essay on (Machine) Learning, Algorithms, and Information Provision*, 47 LOYOLA U. CHICAGO L. J. 541 (2015).

deference to technology as the source of morality creates precisely the problems a demand against censorship seeks to address. In that sense the author's deference to algorithms is a type of "worship" that reverses the skepticism of the Enlightenment.[82] Asking algorithms "to enforce morality" is not only a type of idolatry, it also presumes we know whose morality they enforce and can define what moral outcomes are sought.[83] That is another path to censorship and control.[84] Even allowing a neutral use of a technology is itself is a moral choice, because for example a computer system can allow uses that its operators or users might consider abusive. Nonetheless, in its best light, the argument seems to be that society defers in a default way to morality enforcement by algorithm; and so we must cope with that fact. This view is, however, subverted by the fact that those algorithms will not be perfect at enforcing their chosen values, because "filtering algorithms will always have some error margin where they let through things we might still find objectionable."[85] The somewhat circular logic is that because society defers to algorithms to enforce morality, "with some vigilance we can hold such algorithms accountable and better understand the underlying human (and corporate) criteria that drive such algorithms' moralizing."[86] Of course, the better step is not to defer to such systems, and even if such deference is inevitable—as the study seems to believe—exactly what sort of accountability and understanding is possible is not answered by the critique.

---

[82] *See* Bogost, *supra* note 2.
[83] *Cf.* Desai, supra note 81, at 571-573 (explaining the difficulty for online information providers to show a given user a "good" song or correct entry for a term such as Darwin because of the range of users and each one's view of what a correct result is).
[84] *See* Desai, supra note 81, at 561-562 (noting that someone has to choose what to show users and the history of politicians using media content rules to filter information rather than expand access to it).
[85] Diakopoulos, *supra* note 78.
[86] *Id.*

These investigations also assume that the personalization is well-controlled by the party personalizing the content, but that is not always the case. As one group of computer scientists has noted regarding the use of machine learning algorithms, the notion of what constitutes control and transparency in a given system varies depending on a range of things.[87] As the authors explain, a given application of machine learning is better understood as consisting of abstract "machine learning techniques," "training and operational data," "machine learning outputs," and "the broader systems context; i.e. the workflows, processes, and system supply chains surrounding and integrating the ML" *that work together as a system* and so each component offers different possibilities for control and responsibility.[88]

Focusing on only one part of such a system, the data, shows the ways the idea of control becomes nuanced. Using "data in the wild"—that is deploying a system into the world at large, even if it was known to have been accurate and useful when built—requires ongoing monitoring and evaluation to ensure the model remains accurate given that the real world changes.[89] These changes can create what is called "concept drift" where the "once accurate model [is] obsolete."[90] The change may be because of a general change in the world, or because of active work to defeat the model such as in "spam, intrusion and fraud detection."[91] Inputs can also lead a benign program to render undesired outputs such as what happened with Microsoft's Twitter bot, Tay. That system was designed to have a teenage millennial persona and use slang, but when it was fed

[87] *See e.g.*, Singh et. al., *supra* note 26, at 3-4; *see also* id. at 11-12 (noting how using data in the wild requires ongoing monitoring and evaluation to ensure the model remains accurate given that real world changes and that input in the wild can lead a benign program to render undesired outputs).
[88] *See e.g.*, Id. at 3-4.
[89] *Id*. at 11-12.
[90] *Id*.
[91] *Id*.

data by Internet trolls became "foul-mouthed and racist"[92]—an outcome quite different than intended or expected.

Computer scientists have also looked at personalization and documented the ability for a third-party to launch a "pollution attack"—which "allows third parties to alter the customized content the services return to users who have visited a page containing the exploit."[93] The study examined Amazon, Google, and YouTube's personalization offerings and showed that they were vulnerable to such an attack. In the specific cases one could increase the visibility of YouTube channels, "dramatically increase the [Google] ranking of most websites in the short term" and manipulate Amazon recommendations to display "reasonably popular products of the attacker's choosing."[94] Although the attack was not "powerful, broadly applicable, or hard to defeat," the larger implication is that other sites that use personalization could be vulnerable in similar ways. As the authors put it, "With increasingly complex algorithms and data collection mechanisms aiming for ever higher financial stakes, there are bound to be vulnerabilities that will be exploited by motivated attackers. The age of innocence for personalization is over; we must now face the challenge of securing it."[95]

To summarize, there are broad descriptive claims of a range of differing problems appearing in the public and private sectors and flowing from a range of applications of software techniques. Some of these criticisms assume more control over the systems at issue than may exist. All of these criticisms converge on the notion of transparency as part of a viable solution and yet have different visions of what the term entails and how it

[92] *Id*.
[93] Xinyu Xing, et. al., *Take This Personally: Pollution Attacks on Personalized Services*, 2013 USENIX SECURITY 671, 671 (Aug. 14, 2013).
[94] *Id*.
[95] *Id*.

would work in practice. In contrast, we argue that whether transparency is useful in a particular case or part of a viable solution turns on the context of a given automated process at issue. The next section addresses the nature of the algorithms—or rather the nature of the software—underlying these systems as a step to show why that is so.

## II. ALGORITHMS: A PRIMER

The word, algorithm, conjures dark wizardry, but that is because algorithms are not well understood outside the technical community; not because they are a dark art.[96] Some are simple, and some are complex.[97] Regardless, an algorithm is a step-by-step process and "each of the steps must be precise, requiring no human intuition or guesswork."[98] Thus we can call the steps for brushing teeth an algorithm. However, most of the time, including in the concerns addressed in this work and most of the work we describe, we are concerned not with the conceptual steps but with their reduction to practice as an implementation in computer code. Indeed, there is a difference between

[96] Bogost, *supra* note 2 ("The next time you hear someone talking about algorithms, replace the term with "God" and ask yourself if the meaning changes. Our supposedly algorithmic culture is not a material phenomenon so much as a devotional one."); *see also* KROLL, *supra* note 27, at 2, n.1 (2015) ("The term 'algorithm' is assigned disparate technical meaning in the literatures of computer science and other fields.")
[97] Ian Bogost has pointed out that just as manufacturing seems "automated" but requires "confluence" of raw materials, machines, human labor, and transportation to reach consumers, algorithms such Google's search is a "confluence of physical, virtual, computational, and non-computational stuffs—electricity, data centers, servers, air conditioners, security guards, financial markets." Bogost, *supra* note 2.
[98] JOHN MACCORMICK, NINE ALGORITHMS THAT CHANGED THE FUTURE at Kindle loc. 113 (2012); *accord* THOMAS H. CORMEN, ALGORITHMS UNLOCKED at Kindle loc. 129, 147 (2013).

when humans follow instructions and a computer does.[99] Humans "might be able to tolerate it when an algorithm is imprecisely described, but a computer cannot."[100]

The idea of an algorithm as a recipe shows the problem. Recipes seem to be quite precise, but they are not.[101] As Brain Kernighan explains, the "*Joy of Cooking* says that to poach an egg, 'Put in a small bowl: 1 egg' but fails to specify that the egg should be broken and the shell removed first."[102] Humans can handle this ambiguity, because humans are apt to fill in details or otherwise guess at what to do when presented with a partially specified process, while, as a machine, a computer can only follow precise instructions from the set of operations wired into it. Software is simply a long sequence of these instructions.[103]

As Thomas Cormen puts it, "given an input to a problem, [a computer algorithm] should always produce a correct solution to the problem."[104] Correctness, however, is not so simple; an algorithm's correctness can only be established relative to a specification of its behavior.[105] This point raises two issues. One can fail to choose as one's specification

[99] CORMEN, *supra* note 98, at Kindle loc. 135.
[100] *Id*. 147-48 (2013). As Thomas Cormen puts it, "We want two things from a computer algorithm: given an input to a problem, it should always produce a correct solution to the problem, and it should use computational resources efficiently while doing so." *Id*.
[101] *Cf*. DOMINGOS, *supra* note 123, at 3 ("A cooking recipe is not an algorithm because it doesn't exactly specify what order to do things in or exactly what each step is.").
[102] BRIAN D. KERNIGHAN, BRIAN. D IS FOR DIGITAL at Kindle loc. 1149-1150 (2012).
[103] An irony is that the machine learning and neural network software behind many critiques of algorithms arguably came about because of the specification problem. Specification, until recently, was a wall to advances in artificial intelligence. For example, humans are rather good at making visual distinctions such as between a dog and cat and between one type of cat and another. Computer software was limited in such tasks in part because the specifications could not be precise for each instance or picture of a cat in such a way that software could process well. Recent advances in machine learning and neural networking have allowed software systems to take less precise specifications combined with large data sets so that now the systems can accomplish the distinction task almost as well as a human.
[104] CORMEN, *supra* note 98, at Kindle loc. 147. To be clear, Cormen's full point is that we want a computer algorithm to be correct and efficient, but correctness is the key concern for our discussion here. *Id*. (we want an algorithm to "use computational resources efficiently" while reaching the correct solution).
[105] *See e.g.* Douglas D. Dunlop & Victor R. Basili, *A Comparative Analysis of Functional Correctness*, 14 ACM COMPUTING SURVEYS 229, 229 (June 1982) (defining "functional correctness [as] a methodology for verifying that a program is correct with respect to an abstract specification function").

the correct solution to one's problem or one might fail to implement the solution faithfully. Software that provides directions on GPS systems illustrates the problem.

If one thinks of a GPS navigation system giving a route, there may be several criteria for what the correct route should be.[106] Although people may not often alter how their GPS computes routes, many GPS systems allow one to do so, and in that sense accommodate the driver's preferred approach to determining the correct route. In many cases, a driver simply wants the fastest route given all possible routes.[107] But some drivers will instead want the shortest route, as in least distance, which may not be the fastest.[108] Still others will want the fastest route that also avoids highways or toll roads.[109] Regardless, all correct routes will connect the origin to the destination. But which of the many possible routes is "correct" depends on which of the above options we have selected.

Yet after choosing from the above options, we run into a new problem. Suppose we choose, as our definition of the "correct" route, the fastest route given all possible routes criterion. The routing algorithm must have a way to determine fastest, which means without real time traffic data, the algorithm and its outputs will be incorrect.[110]

Correctness is generally determined with respect to the specification alone, and independently of the correctness of data used as input. Suppose that, instead of real-time traffic data, we provide as input to the algorithm traffic data from the day before. Some of the time, changes in traffic will mean that the algorithm gives a result which is not

[106] CORMEN, *supra* note 98, at Kindle loc. 157-64.
[107] *Id.*
[108] *Id.*
[109] *Id.*
[110] *Id.*

actually the fastest route. The algorithm is nonetheless correct *given the input it had*.[111] The routing algorithm itself "is correct," even if it does not return the fastest result, because "even if the input to the algorithm is not [correct]; for the input given to the routing algorithm, the algorithm produces the fastest route."[112] Thus, the algorithm itself is correct regarding the specification, which in this case is to produce the fastest route given traffic data. However, given incorrect input data, the algorithm may produce an incorrect output.

Software can also have bugs as it implements an algorithm. Staying with our GPS example, suppose that the software sometimes returned the shortest route by distance instead of the fastest route given traffic. Here, the specification is precise and correct (we want the system to return the fastest route, taking traffic conditions into account), and the algorithm we have chosen to implement that specification is correct (it produces the fastest route given traffic data), but the software itself is incorrect with respect to the specification (it occasionally gives an incorrect answer, namely a shorter distance route that was slower because of traffic, because of a bug).

Thus, a computer system can be incorrect either because the approach is incorrect (that is, the specification does not describe a solution to the problem at hand), because the input data are incorrect (that is, the system would be correct if given the right data, but different data were provided), or because a programmer introduced a bug when converting it to computer code. In sum, *correctness is not as precise as policy critics would like*. We must ask whether a solution has been specified correctly and whether that

[111] *Id.*
[112] *Id.*

specification has been correctly reduced to practice, both when data are chosen and when software implementing the solution is written.

As such, to ask that an algorithm not "discriminate" or yield some other result prohibited by legal rules, requires that a precise statement, or specification, be provided so that the request is workable for computer scientists and can be effectively translated into software.[113] And yet the policy process is rarely prepared to provide a complete specification for any rule, let alone thorny questions such as discrimination.[114] Even in the possibly simpler realm of administrative law where rules abound, the administrator "faces decisions for which external standards provide no binding, perhaps no relevant, guidelines."[115] Thus some argue that the "administrative process, like the judicial and legislative process, [is] somehow in pursuit of justice and the general welfare; [and] 'administration,' like 'democracy' and the 'rule of law,' [should be understood] as a motivating ideal."[116] In short, there are ideals that guide the law, but the precise way those ideals manifest themselves is a bit messy and particular to a given application. This dynamic appears in computer science as well.

The idea that all computer systems are designed by first deciding on a precise set of steps, which are completely specified, and then mechanically reducing this precise idea

[113] *Accord* Kroll et. al., *supra* note 27, at 2 ("[C]omputer scientists tend to want a full, technical specification of all the desired properties of an algorithm, but policy processes tend not to declare such precise rules in advance.").
[114] Although some rules, such as the 80/20 rule for employment discrimination, may be precise enough to be a workable specification, it is not clear that other areas of discrimination are as precise. In addition, when one considers claims of censorship versus free speech or whether posting a video clip is fair use, rules or tests are broad and imprecise. Nonetheless, as we argue later, when a clear rule is set by the state, we can ask whether that rule is faithfully reflected in software. Even simple rules, such as rules about how to record and count votes, can be subject to interpretation and challenge, and do not lend themselves to precise specification in all scenarios.
[115] MASHAW, *supra* note 24, at 16.
[116] *Id*, at 14.

to software is wrong. Many systems spring more simply from an "informal notion."[117] Instead of a single formula that can set out or describe a specific result (e.g., $E=MC^2$), large modern computer systems often are developed based on on high-level goals that "generally describe how to solve a problem."[118] For example, it would be difficult or impossible to write a complete recipe for serving a targeted advertisement or arranging posts on a social network timeline. While, at base, these activities are executed by well-specified mechanized processes—similar to administrative law as put into practice—the outcomes came to be through abstract and messy processes that are often reflected in their complexity and in unexpected, unplanned behaviors.[119]

As a colleague in robotics and machine learning put it to one of us, imagine engineers building a robotic arm. Part of the approach applies physics and mechanical engineering to figure out how to drive the arm. But for the arm to work well in the field, where its movements are not precisely determined ahead of time, other engineers and computer scientists apply machine learning and develop models of movement—discovering a specific algorithmic approach to solving the underspecified problem of controlling the arm's movements outside the lab.

The algorithms and models help define a rule for how to move, but it is a rule that is not coded directly by a programmer or designed with the intent to reach the precise rule

[117] MACCORMICK, *supra* note 98, at Kindle Loc. 122; *accord* JAMES GLEICK, THE INFORMATION at Kindle Loc. 3652-3654 (2011) ("An example of an intellectual object that could be called mechanical was the algorithm: another new term for something that had always existed (a recipe, a set of instructions, a step-by-step procedure) but now demanded formal recognition. Babbage and Lovelace trafficked in algorithms without naming them. The twentieth century gave algorithms a central role.").
[118] MACCORMICK, *supra* note 98, at Kindle Loc.122.
[119] *Cf.* Singh et. al., *supra* note 26, at 12 ("large-scale computing environments, for example IoT-enabled smart cities, that entail 'systems of systems' [] have many 'moving parts' – including a range of different software, services, agents (and people!) – all of which might use or be affected by a range of ML models. Managing responsibility in these environments presents a significant challenge. There will be feedback loops between systems, where the outputs/actions of one system can feed into others in real-time. The interactions can be direct, e.g. competing for resources, or more indirect, through 'butterfly effects', where (subtle) actions of a system can (potentially dramatically) affect others.").

discovered. Rather, the discovery of the particular rule eventually put in place is happenstance, although ideally would always generate very similar, very useful rules. The arm movement is not coded based only on a simple equation about force, mass, and acceleration. Instead, a model of the arm's movement comes either from the use of machine learning or simply by capturing sequences of movements of the actuators (for example, when the arm is operated by remote control). This model then defines the control of the arm. While the model is, ultimately, a precise algorithm, it is not one that the arm's creators specified directly. Nor, in many cases, could the arm's creators precisely specify the algorithm in use by the model after-the-fact. The best they can do is specify the process used to create it. This lack of direct coding is especially useful when the arm is to move and interact within a dynamic environment.[120] In fact, some models are developed by guessing parameters at random and testing the model's performance. It can also be beneficial to incorporate randomness into a system's rules (at the time they are run, in addition to the process used to discover them) to address the limits of a static system.[121] None of which is to say that the underlying physics don't matter and aren't used to control the arm. In practice, the physics provide the scaffolding for the arm's movements, defining how settings of the actuators within the arm will change its position or cause it to move, while learned models provide finesse, control, and intelligence to achieve goals, such as how to reconfigure the actuators to move the arm's tip from one place to another.[122] The arm's movements are a combination of formulaic physics and

[120] *Accord* Kroll et. al *supra* note 27, at 21.
[121] *Accord* Id. (noting if one "hard-coded" the movement of a robot like the Roomba vacuum, "an unusual configuration" might end up with the device trapped in a corner but "randomized motion allows it to escape these patterns and work more effectively.").
[122] The use of learned models solves yet another problem, that of interpolation. A precisely specified rule would have to account for how to move the arm from any configuration to any other, while a model can fill in details or learn to move the arm around obstacles in its environment.

models that do not correspond to a pre-written formula, but rather are extracted by discovering patterns in large sets of data.

The key is to optimize the arm's movement, which is done by learning.[123] The arm starts with a model that has unknown parameters. The initial parameters might be set to a good guess of the final model, or they might be chosen at random. The goal of learning, then, is to find the optimal parameters. But as conditions change, the model encounters situations with which it cannot yet deal. When the model gives less-than-optimal results, a learning algorithm will modify the model's parameters and measure whether that modification improves performance in that situation.

Thus we see that the idea that all algorithms are designed when a programmer chooses a precise set of steps is limited, especially in the case of machine learning and other model-driven approaches. While the steps for training the model and for computing its predictions are precisely specified, the model's parameters, which control what guidance it gives, are not specified directly by a programmer. Rather, they are optimized given a set of training data.[124]

Even the environment matters for explaining how a system operates. Suppose we have a program that measures how long it takes for some function to run. Let's say we wish to test how long an algorithm that sorts a large list of numbers takes to run. To test this, a program implementing the algorithm starts a timer before sorting a large list of numbers and then checks how much time has elapsed when the sort is complete. Now,

[123] There are several different approaches to machine learning. For a short overview of the approaches *see* Singh et. al., *supra* note 26, at 4-8; *see generally* PEDRO DOMINGOS: THE MASTER ALGORITHM: HOW THE QUEST FOR THE ULTIMATE LEARNING MACHINE WILL REMAKE OUR WORLD (2016).
[124] *Cf.* Singh et. al., *supra* note 26, at 9 "ML is data driven: (1) the data involved in the training/learning phases determines the model, and (2) the live data on which the model is applied determines the results/outcomes").

suppose the program is run on two computers, one where it is the only program running and one where there are 1000 other programs running, all of which are demanding many resources and reading/writing the disk heavily. Clearly, this one program will behave in two very different ways depending on its environment.

Another feature of algorithms and software that might surprise policymakers is that even if we can posit that both a precise specification and a complete system are available for review, it can nonetheless be impossible to analyze or test whether the system will yield a certain outcome. Although a program is a precise description of its own behavior, it can nonetheless be impossible in some cases to interpret from a description of that program whether it has specific behaviors. This fact runs contrary to the law's mechanistic, Newtonian view of engineering, making it critical to policy discussions about governing algorithms. At a high level, recognizing these boundaries will clear cognitive dissonance between computer scientists and non-computer scientists about the nature of the practices under scrutiny. With such an understanding in hand, critics will be better placed to offer concerns and solutions that computer scientists appreciate as based in sound science. As a practical matter, this fact implies that a general testing program that tests other programs to show that they do not "discriminate" or yield some other result prohibited by legal rules cannot be workable. To support this argument, we turn in Part III to a core part of mathematical theory having to do with decidability and computer science known as the Halting Problem.

## III. TO HALT OR NOT TO HALT

The halting problem implies that some interesting problems are impossible to solve because of fundamental limits that challenge many aspects of computer science. These limits indicate that insofar as law and policy seeks a general transparency tool that analyzes disclosed algorithms for compliance with desired norms, one will not be possible.[125] This challenges the classic view of policymakers that disclosure of a system's internals will lead to a complete understanding of its behavior. The halting problem does not mean algorithms cannot be governed; rather, understanding these limits enables policymakers to craft demands that are workable and achieve the desired ends of policy.

There are certain questions for which there is no algorithm; these questions are called undecidable. That is "[T]here are problems for which it is *provably* impossible to create an algorithm that *always gives a correct answer*."[126] It is possible to prove such things, because computer science at its core is a kind of "mathematical reasoning."[127] In that sense, the deepest ideas in computer science are not about programming software or designing hardware, but abstract concepts and issues that exist beyond real-world instantiations.[128] Understanding what can and cannot be computed abstractly gives insight into the practical question of why demanding software code and input data to test for hidden agendas or prohibited outcomes will not work in at least some cases, perhaps important ones.

[125] When one discusses the idea of algorithmic transparency with computer scientists, there can be a palpable rejection of the idea. The law professor author of this paper encountered versions of this response numerous times, and it was part of what stimulated his interest in this project.
[126] CORMEN, *supra* note 98, at Kindle loc. 4106 (emphasis added).
[127] MACCORMICK, *supra* note 98, at Kindle Loc. 2794-2795.
[128] *Id*. at Kindle Loc. 132 (2012).

We can think of prohibited outcomes such as unlawful discrimination as crashes—problems any programmer wishes to avoid.[129] That is, "Very occasionally, even high-quality, well-written software [will have a bug and therefore] can do something it was not intended to do." That unintended outcome can be thought of as a crash, whether it actually terminates the program or not. So if in fact we could detect when a program is discriminating, any program could be turned into one that crashes when it discriminates simply by running a routine that detects discrimination and crashing if that routine finds discrimination. And then to ask if the original program discriminates, we would only have to ask whether the modified program crashes. One might guess that it would be possible to write an analytic tool which could find just these sorts of bugs. But that hope is a false dream. Software testing has improved such that today many bugs are caught, but it is still impossible to catch all bugs after a program is written.[130] It is, however, possible to write bug-free programs using advanced technical tools that prevent programmers from writing bugs in the first place.[131]

Indeed, for the question of whether a program has bugs that will cause it to crash, society desires the precision of the physicist, mathematician, logician, and critic of the power of algorithms, but it turns out that *one precise thing that can be shown is that we cannot show certain things about software programs including whether a specific*

[129] *Id*. at Kindle Loc. 2775-2778.
[130] *Id*. at 2782-2785 ("A natural question to ask would be: will the automated software-checking tools ever get to the point where they can detect all potential problems in all computer programs? This would certainly be nice, since it would eliminate the possibility of software crashes once and for all. The remarkable thing that we'll learn in this chapter is that this software nirvana will never be attained."); *accord* MICHAEL SIPSER, INTRODUCTION TO THE THEORY OF COMPUTATION, 3d Ed., 165, 201 (2013) ("The general problem of software verification is not solvable by computer."). However, there are approaches to building software such that it has no bugs in the first place. While it is provably impossible to detect all bugs in an existing program, it is demonstrably possible to build programs that have no bugs at all.
[131] This is only one of the common approaches in software verification, the area of computer science concerned with the development of bug-free software. Another is to design the program so that it is possible to test exhaustively all of the states it can ever take and to evaluate the desired property in each of these states, rejecting the program as buggy if the desired property is ever untrue.

*program will crash*. In the specific case of software, detecting a potential crash is an undecidable problem,[132] meaning that "it is provably impossible for any software-checking tool to detect all possible crashes in all programs."[133] But given that such a tool is very desirable, and to counter the argument that this limit is narrow, it is important to see how it applies to computing and software in general and to the problems we are concerned with here in specific.

A more general framing of this problem, and a more powerful theorem about what is undecidable, originates in work by Alan Turing and is known as the halting problem.[134] Turing, often considered "the founder of theoretical computer science" was not writing software or testing for bugs as "no electronic computer had even been built yet."[135] When Turing discovered the proof in 1937, he was "interested in whether or not a given computer program would eventually produce an answer."[136] Specifically, he was interested in determining what was *computable*, or possible to generate with a computer. Because this work applies to what can be known about algorithms in general, it matters to those who wish to regulate them.

Turing offered an abstract, ideal machine that can stand in for any "digital computer" as a way to answer whether there are numbers that are "nameable, definable,

[132] MACCORMICK, *supra* note 98, at Kindle Loc. 3128-3131 ("We proved the undecidability of the Crashing Problem, but you can use essentially the same technique to prove the Halting Problem is also undecidable. And, as you might guess, there are many other problems in computer science that are undecidable.").
[133] *Id*. at Kindle Loc. 2784-2785.
[134] *Id*. at Kindle Loc. 3123-3126 ("A closely related question is: will a given computer program ever terminate— or, alternatively, will it go on computing forever, without producing an answer? This question of whether a given computer program will eventually terminate, or "halt," is known as the Halting Problem. Turing's great achievement was to prove that his variant of the Halting Problem is what computer scientists call "undecidable.'").
[135] *Id.* at Kindle Loc. 3121-3123.
[136] *Id.*

and *not* computable."[137] Turing's machine, called *U*, as in universal, represents a simple yet powerful model of computing machines.[138] In short, "A Turing machine can do everything that a real computer can do."[139]

By connecting the idea of writing down a machine's description with the operation of his universal machine, Turing gave a definition of computation and related it to the definition of algorithm used in computer science; in fact, a Turing machine "capture[s] all algorithms."[140] By extension, whatever applies to Turing machines and the algorithms they can run, applies to the real-world software and machines at issue here. Because Turing machines completely encompass the functionality of real computers, *anything a Turing machine cannot do is also a limitation on real computers*. That is, Turing machines circumscribe the limits of real computers, and abstract limits on what a Turing machine can do cannot be circumvented by building a bigger or more powerful computer or writing better software.[141] The limit we care about here is called the halting problem.

The halting problem captures the problem of decidability for software, answering the question of whether all problem statements which have answers also have the

[137] GLEICK, *supra* note 117, at Kindle Loc. 3683-3684, 3738-3740.
[138] Turing, *supra* note 1, at 241-242; *accord* SIPSER, *supra* note 130, at 202 ("The Turing machine U is interesting in its own right. It is an example of the *universal Turing machine* first proposed by Alan Turing in 1936. This machine is called universal because it capable of simulating any other Turing machine from the description of the machine. The universal Turing machine played an important early role in the development of stored-program computers.").
[139] A Turing machine has "unlimited and unrestricted memory," and "is a much more accurate model of a general purpose computer" than earlier models had been. SIPSER, *supra* note 130, at 165. In particular, Turing machines are a more similar theoretical model to modern electronic computers than even equivalent earlier logical models such as Alonzo Church's Lambda Calculus. *Id*.; *accord* GLEICK, *supra* note 117, at Kindle Loc. 3740-3742 ("No matter how complex a digital computer may grow, its description can still be encoded on tape to be read by U. If a problem can be solved by any digital computer—encoded in symbols and solved algorithmically—the universal machine can solve it as well.").
[140] SIPSER, *supra* note 130, at 184; *accord* GLEICK, *supra* note 117, at Kindle Loc. 3687 (Turing "defined calculation as a mechanical procedure, an algorithm.").
[141] SIPSER, *supra* note 130, at 165 ("even a Turing machine cannot solve certain problems. In a very real sense, these problems are beyond the theoretical limits of computation.").

property that those answers can be computed algorithmically. Specifically, the halting problem is an example of a well-defined problem for which no algorithm can find the answer. The halting problem asks whether there is some method which analyzes a given program running on a certain input and determines whether the input program "halts" or "runs to completion;" Turing proved that no such analysis exists.[142] That is, there is no Evaluator we can use to test a set of instructions (the Standard Description or "S.D" [sic]) on a Machine and see whether a specific outcome will occur.[143] Cormen illustrates the problem this way:

> In the halting problem, the input is a computer program A and the input x to A. The goal is to determine whether program A, running on input x, ever halts. That is, does A with input x run to completion? Perhaps you're thinking that you could write a program—let's call it program B—that reads in program A, reads in x, and simulates A running with input x. That's fine if A on input x actually does run to completion. What if it doesn't? How would program B know when to declare that A will never halt? Couldn't B check for A getting into some sort of infinite loop? The answer is that although you could write B to check for some cases in which A doesn't halt, it is provably impossible to write program B so that it always halts and tells you correctly whether A on input x halts.[144]

We can convert the problems in which we are interested into Turing's halting problem by considering the sort of misbehavior we are concerned with as a kind of crash, which causes a program never to halt,[145] as described above. Given that reformulation, we discover that there does not exist an analysis that will always correctly identify the misbehavior we wish to limit simply by reviewing an algorithm's source code and inputs.

---

[142] CORMEN, *supra* note 98, at Kindle loc. 106-4117; SIPSER, *supra* note 130, at 216-217.
[143] Turing, *supra* note 1, at 248. In Turing's more general language, "We can show further that there can be no machine E which, when applied with the S.D [sic] of an arbitrary machine M, will determine whether M ever prints a given symbol (0 say)." *Id*. For a detailed description of the theorem of undecidablity and step-by-step ideas that lead to the proof *see* SIPSER, *supra* note 130, at 202-209.
[144] CORMEN, *supra* note 98, at Kindle loc. 4106-4117; *accord* SIPSER, *supra* note 130, at 216-217.
[145] By convention, computer scientists consider a program to "halt" if and only if it runs to completion and returns an answer. If the program gets into an infinite loop (i.e., repeats the same set of instructions over and over again forever) or crashes, the program has not "halted" even though it is no longer making any progress towards an answer. Instead, such conditions are often described as the program being "stuck".

Yet these are precisely the elements for which advocates of transparency demand disclosure, so that someone can "look under the hood" of an algorithm.

Put more simply, if the goal or dream is to test, for example, an online ad network, and see whether a specific outcome—like race or gender discrimination in the placement of an ad—will occur, there is no analysis that will always determine that. At this point, it might seem that the author of a piece of software could say that any outcome is unintended, and they could not have done anything to predict that outcome. *That is not so*.

In the Cormen example, B is an analysis program that simulates input program A on its input x. If A halts, B can just report that. But if A does not halt, B doesn't know when to declare that A is stuck. It's not that B is hopeless—it's that there's always some input pair (A, x) such that B will be confused. That is, the system for representing programs and describing their behavior is rich enough to provide a representation that is always inscrutable in this particular way. Turing's theorem says that such a program/input pair *exists*. And that's a very different proposition from the theorem saying that *no analysis can be done*. It's just that *any* analysis will not work *in general*. That is, any sufficiently complicated analysis will be wrong (or unable to reach a conclusion) at least some of the time. Thus, as Cormen puts it, as a general matter we cannot create a program "that determines whether another program meets its specification."[146]

Computer science has met this reality in part by looking for weaker tools that are still useful. Although we cannot compute certain things, we can estimate those things in a way that will be wrong some of the time. In particular, we can limit the failures of

[146] CORMEN, *supra* note 98, at Kindle loc. 4113-4115. A related undecidable problem comes from Rice's theorem, which states that "determining *any* non-trivial property of the languages recognized by Turing machines in undecidable." *See* SIPSER, *supra* note 130, at 219, 240 Problem 5.16. For Rice's paper *see* Henry Gordon Rice, *Classes of Recursively Enumerable Sets and Their Decision Problems*, 74 TRANSACTIONS OF THE AMERICAN MATHEMATICAL SOCIETY 358 (1953).

analysis methods in two ways. First, we can demand that the analysis be *complete*, meaning that all cases of a particular misbehavior will be detected by the analysis. Because the analysis must be wrong some of the time, any complete analysis of an undecidable problem will have *false positives*, or cases where it reports misbehavior where there is none. False positives can cause users of such a tool to become frustrated with it, so minimizing the number of false positives is of paramount importance when building such *unsound* analyses. Second, we can demand the property of *soundness*, which says that any reported misbehavior truly will be misbehavior. Sound analysis methods will, of necessity, miss some cases of misbehavior. No undecidable problem has an analysis that is both sound and complete; many sound and complete analysis methods may exist for problems which are computable, however. It is therefore important to understand whether a property of policy interest is computable, since it will affect whether that property can be established based on the disclosure of software source code, the system's operating environment, and input data.

Put differently, despite the halting problem and issues of undecidability, all is not lost. It is possible to write programs (and their specifications) in sufficiently restricted languages that it is possible to prove that they meet their specifications. In short, computer science has found ways to control for the effect of the limits we have described both in theory and in applications. The next section draws on the stable of techniques with which computer scientists address these challenges to see what policy can learn from those methods and how those solutions can address the regulatory issues critics have raised.

# IV. PRACTICAL SOLUTIONS FROM COMPUTER SCIENCE

Returning to the idea of a result that we do not want, we can use technology to mitigate such outcomes. For example, one might ask for a guarantee that certain software was built a specific way and to have a way to verify that promise. As one of the authors has argued, such a promise is possible if one starts with the goal of building programs that are analyzable, and a sense of for what properties those programs should be analyzed.[147] That is, one must build software with an eye to what must be analyzed, and by whom, because it may be impossible, difficult, or unconvincing to show those things otherwise. Systems must be constructed to produce evidence that they operate as desired, either by doing the things we want or by not doing the things we do not want. Such evidence may involve the disclosure of aspects of the system, but it is often possible without such disclosures.

With a design that enables the production of such evidence, computer science offers ways to give a complete guarantee that something is true about a piece of software under certain circumstances. The problem is that such certainty is impossible for software that shows up from an unknown source, such as malware or software disclosed under a transparency regime and which was developed without regard for the need to produce convincing evidence.

In addition, although we cannot discover 100% of bugs in existing software, we can write software that provably meets a specification and is therefore provably 100% bug-free. Further, it is often possible to achieve high confidence outcomes for properties,

[147] *See generally*, Kroll et. al *supra* note 27.

which are difficult or expensive to specify—such as the high availability or reliability guarantees software companies now offer in some areas.[148] These options should work to address many concerns about software. And, we argue these goals are the ones law and policy should pursue when appropriate.

Nonetheless, these options may still not assuage worries that a computer system will be able to avoid using a specific, prohibited method such as using gender or race in a hiring decision—and so meet the high threshold for legal compliance—and yet still discriminate or otherwise generate unacceptable outcomes. Instead, to demonstrate that the evidence tying a computer system's behavior to its specification is meaningful for policy goals, auditing software systems and the algorithms underlying them using in-the-field tests and social science methods after the systems are deployed can test whether these systems generate undesired outcomes.[149] Although these methods have a history of uncovering discrimination, or at least signs of disparate impact that require an explanation, reliance on these methods faces some difficulties when applied to computer systems. The next section explains when and how certain testing methods can allow someone to test a given computer system and the limits—technical and practical—of those techniques.[150]

## A. Reviewing Algorithms, Software, and Decisions

[148] *Cf.* CORMEN, *supra* note 98, at Kindle loc. 3149 ("[U]ndecidability has limited practical effects: it turns out that we can often do a good job of solving undecidable problems most of the time."). Beyond undecidability, some desirable properties (such as a system always being available, or a system being sufficiently secure) may be specifiable and decidable but hard to analyze, test for, or achieve in practice.
[149] Sandvig et. al., *supra* note 15.
[150] *See supra* notes __.

The practical issues computer scientists face in evaluating software and algorithms show the limits of transparency as a solution on its own[151] and that different instantiations and applications of algorithms require different approaches to regulation. There are two common settings in which one tests software, white-box and black box. In white-box settings, the analyst has access to the source code. While that is the dream scenario of advocates for source code disclosure, approaches using white-box testing still have limits. Black-box settings, in which the analyst is restricted to only see the inputs and outputs of the system but not its internal operation, pose more problems. Some limitations apply in both settings. In either setting, there are two categories of analysis: static analysis, which examines the program's structure or code without actually running it; and dynamic analysis, which runs the program and observes its behavior on certain inputs.

All dynamic analysis is only as good as the "number of input–output pairs that are available to be observed or tested."[152] Once one considers that many of the algorithms of concern can operate across a massive number of inputs, it becomes apparent that one can only test a (perhaps insignificantly) small subset of those inputs.[153] And the inputs and outputs available for review or generated by such testing may or may not match the set of inputs and outputs that matter for policy analysis.[154]

[151] This discussion is indebted to and draws on Kroll et. al's paper. In addition, we offer thanks to Ariel Feldman, for his generosity in exploring these issues.
[152] *Cf.* Kroll et. al, *supra* note 27, at 16.
[153] *Id.*
[154] For example, in the black-box studies of search engines and advertising networks mentioned above (*supra*, notes 60-67), while researchers could view problematic *outputs* of particular queries, they could not see the output for every search query or every possible advertisement considered by the system. Because of this, they could not conclude what the cause of the problematic behavior was, nor recommend a specific remedy.

Finally, testing is impotent to determine whether some result came from a particular system in the field. Consider again the problem of securing the voting process. While vote counting is a very simple operation requiring no complex specification and admitting to straightforward testing and even full verification to produce software with no bugs, there is no way to turn the output of such software alone into evidence that it resulted from the inputs (i.e., ballots) approved by the surrounding voter verification processes (e.g., id verification, log books, comparison to voter registration data, etc). Instead, evidence of the correctness of the count must in some way relate the inputs to the final result. Otherwise, there is no telling whether the approved, correct software was swapped out with different, possibly malicious software in the voting machines.

### 1. White Box Testing

White-box testing is commonly done during the development of software systems, but can be done at any time. Kroll et. al describe the ways one may test a program in a white-box setting:

> Computer scientists evaluate programs using either static methods—which look at the code without running the program—or dynamic methods—which run the program and see how the program behaves for certain inputs. Dynamic methods can be subcategorized into 1) methods that rely on observation of how to program operates in the field with naturally occurring inputs and 2) more powerful methods that include testing, where an analyst chooses inputs and submits them to the program.[155]

Although powerful, these testing and analysis methods do not solve all the issues around algorithmic transparency. Static methods are not "perfect." Experts can easily miss simple problems hidden in complicated code, and there is a theoretical limit on all

[155] *Id*. 13.

program analysis, both static and dynamic, that the halting problem implies: one cannot always predict whether a certain outcome will occur.[156]

In addition, while white-box testing is more powerful than black-box testing (since any black-box test can also be run in a white-box setting), it may not be obvious which input/output pairs will provide the most information to a dynamic analysis, since nearly any interesting static analysis is not computable and therefore will either be unsound or incomplete, as described above. For this reason, it is generally necessary when developing software to limit the scope of what inputs a program will allow, if only to make that program more easily testable.

Another problem is that the fact that software is discrete and so testing what happens for an input x tells you essentially nothing about what happens for input y—it might be completely, radically different. Unless you happen to know something about the structure of the program, you'd have to test all possible inputs, which is rarely feasible for programs of interest.[157] No amount of sophistication in the testing process or tools, nor any unstructured disclosure of information from the creator or operator of a piece of software, can circumvent these fundamental limits of software bug testing.[158]

## 2. Black Box Testing

[156] Kroll et. al, *supra* note 27, at 16.
[157] Still, this approach is sometimes done in the form of "model checking."
[158] *Id*.

Black-box testing suffers all of the limits on white-box testing, with the additional limitation that the software source code cannot be inspected. A further limit concerns the difference between analysis, in which a reviewer looks at the behavior of a system on naturally occurring inputs, and testing, in which the reviewer has the power to introduce new, synthetic inputs in order to observe differences in the system's behavior between inputs. Many large systems of interest, such as search engines and social networks, are hybrid between these scenarios: to interrogate the system, some inputs will be able to be changed or have new choices inserted, while others will stay the same. For example, it is possible to change the queries submitted to a search engine, but not possible to replace the results it has crawled from the Internet at large.

As Kroll et al. explain, for black box testing, because of the "astronomical number of hypotheses about the behavior the program that [] fit the observed or tested input output pairs," "it is impossible to use inductive reasoning to determine the decision procedure generating outcomes or to predict the behavior of the program for inputs that have not been observed or tested."[159] Even if one combined static and dynamic testing methods, "Not every algorithm will be able to be fully analyzed" because of the intrinsic limit of the halting problem, as described above.[160] As they note, if an algorithm was not "designed with future evaluation and accountability in mind," no amount of software testing—even aided by total transparency—will always work to elucidate any particular

[159] *Id*.
[160] This point comes more directly from a close cousin of the halting problem, Rice's Theorem, which holds that for any non-trivial property of a language (or program) of a Turing machine, "determining *any property* of the language[] … is undecidable." *See* SIPSER, *supra* note 130, at 219; *accord* Kroll et. al at 5, n. 14. For Rice's paper *see* Rice, *supra* note 146. For a statement of theorem see SIPSER, *supra* note 130, at 240, problem 5.16. For the proof *see* Id. at 243, Solution 5.16.

question.[161] To combat the problems and limits of transparency and testing, Kroll et. al offer a different solution, but it, too, has limits.

### 3. A Third Way: Ex-post Analysis and Oversight

In simple terms, the goal of governing automated decision making is to review a software-driven action after-the-fact of the action, to see if it comports with applicable social, political, or legal norms. That goal runs into the problems of transparency and software testing we have described, as well as the problem of determining whether the action originated from a specific piece of software. There are many options to meet those problems. One option is full transparency – a computer is, after all, a precise machine, and with a full understanding of its construction, inputs, programming, and operating state, we can reproduce its actions. But transparency is only one way to achieve the evidence necessary to explain how and why a computer system produced a certain decision. Other methods create convincing evidence without complete disclosure of a system's internals. Full transparency, including transparency of a program's operating environment, can yield complete explanations for a system's behavior (or at least be able to reliably reproduce that behavior simply by running the disclosed system again). But such complete transparency is rarely possible or desirable. Often such systems may be subject to trade secret or other concerns that run counter to full transparency. And even when it is possible, we run into the limits discussed in Part II above, for full transparency will in many cases require disclosing significant detail about a program's operating environment, such as the full contents of databases a program interacts with and

[161] Kroll et. al, *supra* note 27, at 24.

information about other programs running on the same computer.[162] Again, such detail is rarely desirable, or even feasible, to disclose. But we do not need such detail to achieve the goals of governing software systems.

Insofar as reviewing an outcome after-the-fact is about the ability to evaluate properties of the software being used, we do not need full transparency. Instead as Kroll et. al point out, one can use a suite of technical tools including cryptographic commitments[163] and zero-knowledge proofs[164] to allow for an automated decision process to be used and at the same time "provide accountability" in the sense that a reviewer can check that even undisclosed elements are duly recorded, or are applied equally among decision subjects as appropriate (e.g., all decisions are rendered from the same decision policy).[165] These techniques allow for ex-post verification even when the entire system is not transparent, or not transparent to all concerned parties, functioning as a kind of self-executing software escrow. Skeptical decision subjects, oversight entities, and concerned citizens can later verify that the software which was meant to be used and which was committed to—or, in the escrow analogy, was deposited into escrow—was actually used for a decision. Further, these techniques allow selective transparency: a system may not need to be transparent to everyone, so long as members of the public are confident that the system's decisions about them correspond to the version of the system reviewed by a

[162] *See supra* Part II.
[163] *See* Kroll et. al, *supra* note 27, at 30 ("A cryptographic commitment is the digital equivalent of a sealed document held by a third party or in a safe place. … Commitments are a kind of promise that binds the committer to a specific value for the object being committed to (i.e., the object inside the envelope) such that the object can later be revealed and anyone can verify that the commitment corresponds to that digital object.").
[164] *Id*. at 32 ("A zero-knowledge proof is a cryptographic tool that allows a decisionmaker, as part of a cryptographic commitment, to prove that the decision policy that was actually used (or the particular decision reached in a certain case) has a certain property, but without having to reveal either how that property is known or what the decision policy actually is.").
[165] Kroll et. al, *supra* note 27, at 27.

reviewer with the authority to compel transparency—and again, in the escrow analogy, a trusted agent could determine that these conditions were met without actually disclosing the escrowed software. Finally, these techniques directly verify the relationship between inputs, policy, and decision output, sidestepping the need to disclose or record the entire operating environment of a computer system. Thus, they allow disclosure of only the relevant facts (such as the particular inputs and decision policy used) to only the relevant parties (such as only competent oversight authorities, e.g. a court issuing a duly executed order for discovery) without compromising the ability of others (i.e., decision subjects, the public at large) to verify that oversight can occur. These techniques also reduce the scope of disclosure, replacing the need for certain disclosures with direct evidence.

If one wants to review any action after-the-fact, one needs an audit log or audit trail, that is a time-stamped record that documents actions that affect an operation, procedure, or event.[166] In human-driven, bureaucratic processes, the audit log serves to identify who did what when, and why any intermediate decisions were taken. However, it is not possible to reconstruct a human's precise thought process simply from notes. On the other hand, digital audit logs can improve on the governance of decision making because the process that led to a decision can be precisely reconstructed and analyzed. If one wants to know that the audit log in a computer system corresponds to what actually happened, one can either re-run the entire system (disclosed via a suitable transparency regime) and compare the inputs and outputs or one can use these cryptographic methods to receive and verify direct evidence of how the outcome was generated. In addition, even a passive observer who is a member of the public (and not just privileged regulators with

[166] The reliance on such trails is known and used in the law. For example, the Food and Drug Act requires the use of such trails in electronic record keeping. *See* 21 C.F.R. § 11.10(e).

the power to compel disclosure of all or parts of the system) can determine that the audit log is correct if it is created properly at the time the decision is made. Further, with sufficient evidence, an observer will be able to determine that the actions recorded in the audit log correspond to the system disclosed to a regulator. These methods can also show an outsider that the audit log corresponds to a process with certain desirable properties, e.g., showing that the same decision policy was applied in all cases.

## B. Dynamic Systems and the Limits of Ex-Post Testing

Although the above techniques hold promise for many areas of concern, they do not cover all problems that might undermine public trust in computer systems. Dynamic systems that change over time pose two problems for ex-post review. First, even if such systems are created with the above requirements in mind, whatever analysis is possible may aid in determining whether, when, and how things changed and to isolate the effects of changes. But how well these approaches could aid such analysis remains an open research area. Second, dynamic systems already in place but not designed to support this sort of review remain difficult to interrogate.

The systems that of most concern—those that govern "search engine rankings, spam filters, intrusion detection systems, … website ads, … [and which] social media posts to display to users"[167]—are not covered by these solutions. That is, software that uses certain types of machine learning or is modified frequently by its operators to

[167] Kroll et. al, *supra* note 27, at 24-25.

respond and adapt to dynamic inputs and user behavior does not lend itself to a tidy analysis of a single, coherent policy uniformly applied. Many systems change often, either because of "regular" changes by designers (for example, to enhance their functionality or combat abuse) or because they use automated processes such as *online* machine learning models "which [can] update their [. . .] predictions after each decision, incorporating each observation as part of their training data."[168] The approach of creating an audit log showing that everyone is subject to the same decision policy is less useful when systems are dynamic and change over time, because the system may (desirably) change between decisions.

As a general matter, Kroll et. al's solutions address issues looking forward, requiring that decision makers decide on a single policy and publish it in advance of making any decisions. These methods, at least as described, do not directly apply to situations where the decision policy changes often. Instead, these methods must be adapted to address the idea that the decision policy changes over time, as we discuss below in Part V, Section 3.

Algorithms and the software systems that bring them to the real world vary, and regulatory approaches to controlling their power must be shaped by who uses them, how they are used, and for what purpose they are fielded. As such, we next look at the sort of systems and contexts where the approaches offered by Kroll et al. fit well and then turn to the open questions of systems already in place or that may be less amenable to these approaches.

[168] *Id*. at 25.

# V. A TAXONOMY OF POTENTIAL SOLUTIONS

Given that software-based decision-making appears to be here to stay and that there are limits to traditional transparency solutions for such decision-making, this section sets out a taxonomy of when certain approaches for accountable algorithms work well from both a technical and legal standpoint. The nature of the appropriate mechanism for accountability will depend on the nature of the software system at issue. In some cases, the extent to which accountability is needed will turn on whether and how the use of such systems is regulated. In some cases, robust accountability will be a law and policy requirement; in others, building accountability into software systems will be a best practice, at least under current policy. We begin this Part by looking at public sector decision-making and explain why accountability is necessary as matter of due process. We then turn to private sector, regulated industries which also require accountability. Next we examine unregulated industries and offer best practices for accountability. In addition, we offer a possible statutory change—the passage of law to encourage and protect whistleblowers who know of prohibited practices—to aid in policing the use of software in decision-making.

## A. Public Systems

We start with the easier case, public systems. As described above, in general accountable public systems promote the dignity of citizens and support a well-functioning society. By extension, citizens need some ability to verify that public systems operated in a certain way, and complied with social norms, political realities, and legal obligations. Recall that a large barrier to accountability and evaluation occurs when a system is built without the requirement to generate this evidence in mind from the start. The easiest approach to a solution is that when the government chooses to use or purchase software, it must use software that meets the standards the government sets and the standards should include full transparency.[169] And while full transparency of the software source code used to build government systems would certainly be useful, both for understanding software-mediated decision-making by the government and as an end in itself, such disclosure is insufficient on its own to provide governance. Indeed, as we have argued, calls for transparency often miss the point, going too far for some requirements and not far enough for others.

We offer instead that when the state is using software for sensitive decision-making that raises due process concerns or where the integrity of the state's process may be questioned (e.g., when using voting machines) the state must use software that allows for accountability to the public and evaluation by citizens at large. The state may build such software in-house or buy it, but the requirement applies in both cases. Given that the

[169] Citron, for example, advances a more complete view of this argument, describing how it would be achieved and saying that "open code governance provides a means to make agency decisions bound up in information systems more transparent, democratic, and legitimate" in her article "Open Code Governance", University of Chicago Legal Forum 355 (2008). Citron goes further than transparency, arguing that government-procured code should be made open-source, enabling "new opportunities for participation by a broad network of programmers, who can contribute to the development of accurate and secure systems." Indeed, many scholars and advocates have proposed that government-procured software should be open source, both for reasons of transparency and also because openness fosters participation and treats such software as a public resource, consonant with the doctrine that government works are not entitled to copyright protections.

government, especially at the state or local level, may not be able to build the software it needs. When using outside vendors to provide the software or software based-analysis in these areas, the government can and should define complete requirements for procured systems based on the goals those systems are meant to fulfill. In addition, it should demand in procurement requirements and contracts that the resultant system be able to generate evidence verifiable to a competent observer that it satisfies those goals. This evidence could be generated per-decision or once for the entire system if the application scenario allows it. Further, what constitutes sufficient evidence will depend on the goals of the system overall. For example, in electronic voting an auxiliary voter-reviewed paper record may suffice as evidence of correctness, as counts can later audit the correspondence between electronic and paper ballots. Conversely, in another counting application such as the census, paper records alone cannot describe sampling procedures or interpolation from returned questionnaires, and do not constitute sufficient evidence of the system's correctness for its designated purpose. To have sufficient accountability, it must be possible to review the correctness of every individual decision produced by the system.

Given that the government already vets software for security and privacy issues among other criteria, it is hardly peculiar to demand that systems produce evidence of the correctness of each decision they make. Private sector companies already employ testing regimes to convince themselves that software (both for internal use and for sale to customers) operates as expected. And in many cases, software vendors must provide evidence beyond simply attesting to the correctness of their software. For example, many online service providers specify an application programming interface (API) which

describes precisely how software operating remotely on the provider's computers is meant to operate (deviations from the promised behavior in the API are thus clear bugs). In addition, enterprise software vendors create custom software built to client-specified requirements all the time. The government is a large, albeit somewhat special, customer, but it is nonetheless a customer that can list its needs for the market to meet. Indeed, at least one agency, the Food and Drug Administration, sets out technical requirements by requiring those who "create, modify, maintain, or transmit electronic records shall employ procedures and controls designed to ensure the authenticity, integrity, and, when appropriate, the confidentiality of electronic records, and to ensure that the signer cannot readily repudiate the signed record as not genuine."[170] The FDA also specifies that audit trails be a part of "such procedures and controls."[171] Our argument is that Kroll et. al's approach enhances such requirements, by helping to ensure that the audit trails have not been tampered with or faked and tying the evidence in them to the actual actions taken. Those assurances help to fulfill Mashaw's three demands for bureaucratic justice.

Thus we argue the state must still adhere to Mashaw's three points: making "accurate," "cost-effective" judgments, while giving "attention to the dignity of participants,"[172] but a modification of the demands of dignity is needed. Although the subject of such a process may not have the literal ability to know or understand what reasons are behind a decision,[173] when a sensitive decision is being made—or as Mashaw, Schwartz, and Citron have stated in different ways, when the state is making decisions

[170] *See* 21 C.F.R. § 11.10.
[171] *See* 21 C.F.R. § 11.10 ("Such procedures and controls shall include the following: … Use of secure, computer-generated, time-stamped audit trails to independently record the date and time of operator entries and actions that create, modify, or delete electronic records. Record changes shall not obscure previously recorded information. Such audit trail documentation shall be retained for a period at least as long as that required for the subject electronic records and shall be available for agency review and copying.").
[172] Schwartz, *supra* note 18, at 1348.
[173] *Id*. at 1349.

that raise due process concerns—the state must use software that produces sufficient evidence to allow for accountability and evaluation by a competent authority after the fact of the decision. Further, it must be possible *even without oversight* for a decision subject to verify the integrity of this evidence (i.e., that it is correct, corresponds to the decision, and does not contradict itself). In general, this requirement assures that the subject of any such processes can determine that the rules and procedures have been followed. Thus, even if a subject cannot know or fully understand the software and data behind a process, she can know and understand the rules and procedures were followed in a way that protects her dignity, as Mashaw has developed that idea.

Further examining this approach illustrates its benefits. At the general level, software firms will have a requirement that they can build against—the production of specific evidence that allows for accountability and evaluation of the system's performance with respect to specific goals by customers and, ultimately, end users and decision subjects. Insofar as the government claims that a process uses certain methods, meets certain legal requirements, etc., the government can offer clear requirements that a vendor can implement. Of course as Citron notes "Policy is often distorted when programmers translate it into code."[174] That problem is real. The benefit of accountability mechanisms is that they provide a way to test the code ex post and see whether such a problem has occurred so that it can be fixed. Government programs that Schwartz and Citron critiqued or that Kroll et. al offer as good cases for accountability have limits on which factors they can and cannot consider and on the methods they can and cannot use to make a decision. When an agency is deciding who should receive "Medicaid, food stamp, and welfare" benefits, whether someone should be on a no fly list, who should be

[174] Citron, *supra* note 5, at 1261.

identified as owing child support,[175] who should be granted a Visa, whom to audit for tax compliance, whom to search for security checks at airports, and so on, it can set out what criteria were expected to be used *and* show that in fact it used those criteria (and only those criteria) if software was part of the decision-making.

Of course, many decisions in this area involve the discretion of an agency employee, a person, rather than vesting the full agency of the decision in software. But that does not change the need for having sufficient evidence to analyze the software component or to hold the entire process accountable.[176] Insofar as the parameters of a particular application allow for little discretion, when they are set out and applied via software with audit-logs and the sort of direct evidence Kroll et. al describe, citizens can have better assurance that the decision-maker followed the parameters. Even if the decision-maker has discretion, if software was used to process data within prescribed limits for how a decision was made, the decision-maker can show that the stages up to her discretionary decision adhered to required criteria and thus meet the demands of due process. Further, the decision-maker can record in an audit log what information was present at the stage where discretion was applied, and any details about how and why discretion was applied. Later, this information can be used to adjudicate any dispute about the appropriate application of discretion in a particular case.

[175] Citron, *supra* note 5, at 1256-57.
[176] *See* MASHAW, *supra* notes 115-116, and accompanying text (noting difference between exact rules and demands of justice for administrative law).

## B. Private Systems

Although due process concerns explain why government use of software for certain decision-making applications requires the creation of the sort of evidence we have described, whether the same is true for private sector uses turns on the nature of the private activity at issue. Private sector actors may want to offer, of their own accord, evidence of how a decision was made to engender trust with their customers. However, the extent to which a given actor or area of industry will be required by law to do so varies depending on the actor or industry.

Whether software making decisions in the private sector must produce evidence of its correct operation turns on a few issues. If the private sector industry in question is regulated, such as the transportation or pharmaceutical industry, the creation of evidence by automated decision processes is a natural requirement. If the sector in question is not regulated, it would seem that the accountability such evidence would enable is not required, but it is certainly a useful best practice and may be a de facto requirement under current policy. Given that the Federal Trade Commission or other consumer protection agencies may have questions about how a process worked or the truthfulness of a claim about the quality of an offering, we argue that any company should consider their requirements for designing software to produce evidence or audit trails of how decisions were made as a best practice, since it can improve trust in their products and services. In addition, we offer that, to the extent that disclosing such evidence explains how a system works, such an approach can allow community feedback and encourage the identification of unexpected errors in ways that can help companies built better products. In the security

space, bug testing challenges, open-source development, and bug bounty programs currently already fulfill this role. Last, we offer that whistleblower protection may be useful to aid in revealing whether a company is intentionally using prohibited discriminatory methods, such as criteria based on race or gender in software making decisions where these factors are proscribed.

## 1. Explicitly Regulated Industries

The use of software by the private sector in regulated industries raises a tension between trade secrets and the need to know whether a system adheres to agreed-upon or required standards. The recent discovery that Volkswagen used software so that its cars passed emissions tests but performed differently on the road is one example of a problem where evidence of how the software operates would enable evaluation, promote accountability, and ensure compliance. An automaker could be required to provide the software to a government tester. The software could be required to be designed to produce sufficient evidence of how it operates to make it straightforward for the tester to analyze the software and determine if it complies with the standard under test. Using Kroll et. al's idea about zero-knowledge proofs, the automaker could accomplish this while keeping proprietary methods secret, yet still be required to commit to using certain methods and to produce sufficient evidence so that testers could verify that the system performed as promised.

A larger problem arises with networked systems, which turn almost anything into a PC or browser that can be updated often if not daily. For example, Tesla announced it

would address issues about the range its cars can travel on one battery charge by a software update. The update is designed to allow the range meter (which indicates how far one can go before needing to recharge the battery) to be more accurate by accounting for real time driving conditions such as terrain and wind and by checking the distance from a charging station to let drivers know whether they are moving out of range of the closest station. And while that new functionality is important and useful, updates can radically change the behavior, and therefore compliance properties, of a computer system.

The key point is that Tesla, and other automakers pursuing the next generation of automotive functionality, are treating the car more like a computer than previous vehicles, in the sense that features are implemented as software running on generic hardware rather than as dedicated-to-a-purpose mechanical components that would have to be physically swapped out to change their behavior. Tesla alone issues updates every few months to improve 0 to 60 performance and add safety features such as "active safety features like automatic emergency braking, blind spot warning, and side collision warning" on models that have the hardware needed for the systems to work.[177] Other updates claim to improve radio reception and create a guest or "valet" driver mode to limit the way the car can be driven and access to confidential information.[178] And the so-called auto-pilot mode was delivered as a software upgrade too. Future Teslas, Volkswagens, and really any automaker's cars will rely more and more on software for most of the way the car operates, and on network connections to the world outside the vehicle for updates and real-time data. This calls out a problem for any networked, software driven device—what is often called "the Internet of Things"—that requires some approval before being

[177] *See* Davies, *supra* note 11.
[178] *Id*.

deployed or which must comply with some safety standard, since software-driven features of such devices can change radically when the software is updated.

Insofar as any device makes a claim about performance or adds a feature in a regulated context, that device must be tested before being put into the stream of commerce and should be built to provide the requisite level of evidence and assurance to facilitate accountability and evaluation. If a company wants to update systems that normally undergo review by a regulatory agency such as the NHTSA or FDA, it must not be allowed to push the update until the agency has analyzed the changes and verified the continued compliance of the updated device, subject to the original, approved parameters. This includes verifying that sufficient procedures are in place to allow the agencies to review the operation of devices in the field or on an interaction-by-interaction basis (if this is necessary). One ancillary benefit of carefully defining evidence for showing key compliance properties of a piece of software is that—unless an update transgresses the limits of the agreed-upon original testing parameters—companies can update rapidly as long as they can demonstrate that their update falls within those parameters. Further, clarity about what a device must do (and must show it does) can help companies focus their software development efforts. With such guarantees from a company in place, an agency will be able to test and approve updates faster than it could in a world where every product revision had to be reviewed *de novo*. Thus, updates could be pushed out more quickly, because oversight bodies will have the technical evidence showing whether the updated product was still within the regulations, or has changed sufficiently to require a more robust period of testing and approval.

## 2. Building Trust: Implicitly Regulated Industries or Activities

Battery life for cell phones shows how the question of transparency is different in unregulated sectors. Like a car battery range indicator, the charge indicator on a cell phone is based on a model, rather than reporting a measurement determined in hardware. Like a car battery charge, the way one uses the device affects the length of the charge. As a general matter, users and non-users, will want to know whether the indicator is accurate within certain parameters, and may want a way to check the truthfulness of the indicator as part of consumer protection. As with the Tesla example, if a company makes a claim about the length of time a charge lasts, the more third parties that can verify the claim, the better. Again, insofar as a system is somewhat unchanging and the public wishes to know whether the system adheres to certain methods and practices, the Kroll et al. criteria should work well. And if a company wishes to update or enhance a feature that does not require government testing and approval before launch, but the company wishes later to argue that its logs and data indicate all was well with a system or operation under scrutiny, it will need to offer a way for third parties to verify that evidence proffered to this end is as it was when the issue arose, and has not been doctored.

Thus, starting with the goal of producing evidence of chosen desired properties, to support accountability to customers or regulators and evaluation by the public, as software is developed and deployed aids a company in two ways. First, the approach allows a company to address the growing trend of journalists, online communities, and perhaps cranks (i.e., purveyors of fake news) to be provocative with grand claims of

failure or misdeeds. Second, should criticisms catch the attention of an agency like the FTC or a class action suit be filed, companies will be in a better place to respond to legitimate inquiry without having to stick their head in technical sand and claim that trade secrets, complexity, technical infeasibility or some combination of these issues means the public must simply defer to and trust the company that they are being honest and obeying rules.

Although designing software that gives evidence that it meets chosen requirements is a powerful solution to problems stemming from software-based decision-making, two issues pose a problem for this approach. We turn to those next.

## 3. The Challenge of Dynamic Systems

At this point we must turn to a problem that affects both public and private sector uses of software: highly dynamic systems, especially those relying on *online* machine learning techniques, may not be amenable to the accountability and evaluation criteria for which we argue. As above, whether such systems should be allowed in certain contexts or in what way they may be used turns on whether the public or private sector is using such systems and the purpose for which they are deployed.

The baseline reasons for requiring that public sector uses of software generate evidence that they can be evaluated by oversight entities and so provide accountability for decision-makers, namely concerns around due process, mean that the government may not be allowed to use such dynamic systems for certain purposes. Regardless of

whether the government builds or outsources the creation of the software in question, even when accountability and evaluation are possible and the software adequately addresses compliance requirements, due process and justice concerns specific to the application of these tools by government indicate that such systems cannot be used. The counter-argument that preventing the government from using the latest and greatest techniques in machine learning means that government is hampered and will be unable to do its job well is, of course, an overstatement. If an agency wants to use a highly dynamic system, the agency will bear the burden of proving that the system satisfies due process and justice concerns. And even if there is a case where the is no system that satisfies these requirements, the agency would have to show that in that extraordinary case, the trade off means it should be allowed to use it nonetheless It also misses the point of Mashaw's triumvirate for government decision-making. Decisions must strive to be accurate, cost-effective, *and* to give attention to the dignity of the subject, and although those three criteria may "compete with one another, [] bureaucratic justice becomes impossible without respect for all three of them."[179] Sometimes, ongoing modification of a rule is desirable and falls within the requirements of bureaucratic justice. Thinking about airlines and anti-terrorist safety measures shows the problem.

Other than perpetrators, no one is in favor of hijacking or blowing up a plane, and a goal of national security efforts is to prevent such actions. New machine learning techniques may be better than humans alone at detecting patterns that indicate someone may be trying to hijack or blow-up a plane. Such a system might create or take as input from human analysts a list of suspect people, either for enhanced security review or to be banned from flying. On the one hand, the process must be cost-effective. Given the

[179] Schwartz, *supra* note 18, at 1349.

hundreds of millions of people who travel within just the United States each year, having humans try to cull through just the passports and visas would be an enormous task. Even if society had enough people to do that work, it is unlikely that they could be trained to pick up patterns, do so consistently, and do so fast enough that the pattern found is still relevant to the task of finding terrorists the next day or week. In addition, the raw dollar cost of such manual review would be quite high. If the task further requires cross-referencing the paper trail with pubic online social media posts, facial recognition, and more, we see that the task is not at all a good one for humans but an excellent candidate for automation. In particular, such a task seems well suited for a machine learning system. But if such a list of suspected people is generated, and yet no one can explain how the list came to be, the goals of bureaucratic justice are not served. Even if a model can claim to satisfy requirements of due process by applying to all potential suspects in an equal way, no "ground truth" exists for the criterion of the list (an unmeasurable risk), so testing whether the list is accurate is an open question. Furthermore, if someone is on the list, believes that he or she should not be, and wants to challenge the choice, any lack of analyzability of the model which generates the list undermines the dignity interest that completes the triumvirate. Thus, to satisfy the requirements of bureaucratic justice, government should not be able to use a technique that does not meet the three criteria and by extension, a technique that does not allow for the evaluation of how decisions are made and accountability for these justifications. Yet following our approach saves the process.

Although a system that created a dynamic rule for selecting passengers would facially raise concerns about the dignity of screened passengers, it could conceivably be

constrained to update its rule in a way that ensures desirable properties. For example, a system might guarantee that the chance of having to undergo more stringent security procedures was independent of certain protected statuses, such as age, gender, or ethnicity. As long as updates to the rule do not violate this requirement and do not stray outside the envelope defined by a sufficient rulemaking, small updates to the rule to detect new patterns indicating risk might be allowable.

Private sector uses of such systems will run into problems insofar as they are regulated or need to show that they adhere to a certain set of rules or procedures. This is especially true of the many systems under scrutiny already built, in place, and working. Four areas—social networking, search, online news, and online advertising—have come under sustained criticism for several years and reveal the problem. We will not restate claims from the literature, as they are varied and proffered objections to automated practices are disparately accurate and substantive. For the purposes of this Paper, we remark only that companies in any of these areas have been accused of misdeeds, and a key question in all the cases is whether a given system behaves in a certain way. This question highlights the problem with highly dynamic systems, because such systems evolve the rule in effect for any given decision, creating challenges in determining why a system behaved as it did. That is, whether a company using dynamic systems will be able to work within the criteria for which we have called, namely for accountability and the creation of evidence as to how a system functions, will depend on precisely how and when the decision rules are updated.

As in the public sector example, so long as decision rule updates can be shown not to change an acceptable rule into an unacceptable one, or to change a rule too fast for

others to adapt to, rule changes may be acceptable and even desireable. For example, information providers such as search engines and social networks often change their ranking algorithms to combat abuse by low-quality sites or posts that want to show up higher in the results. These changes benefit both users, who see better results, and legitimate site owners or posters, who need not worry that their place will be usurped by dint of others' bad behavior. However, large changes to the ranking algorithms invariably demote some legitimate sites which were naturally better suited to the rules prior to the changes. In search, because many websites use advertising as their primary revenue source, large changes in traffic can have commensurate effects on sites' bottom lines. Naturally, sites which are demoted in ranking updates suffer corresponding decreases in the number of visitors they see, and thus generate less revenue for their owners. Website operators therefore often see large rule changes as arbitrary and capricious. For social networks, users may believe their posts ought to be seen by their contacts, and so changes to the way posts are ranked and displayed which alter that expectation are often greeted with skepticism. Changes can raise questions about fake news or bias about which news or posts are promoted. If, however, the changes by information providers happened in small steps over time or could be shown to only change the rules within certain bounds or could explain to users how and why the changes affect content rankings, rule changes would be more palatable and could be weighed against the surplus created by changing the rule.

Until recently the need to assess such systems has not been urgent, but as they become more common, the people they affect are demanding the ability to assess them. The terms "machine learning" and "artificial intelligence" suffer from a problem similar

to the term algorithm: there is an almost mystical or monolithic view of what the terms mean. Artificial intelligence seeks to build systems that solve general and abstract problems in the way humans do, or in yet-undiscovered better ways. However, AI remains in the distant future and the stuff of science fiction, even as the complexity of systems that do not generalize their approaches makes computers seem ever smarter. These seemingly intelligent systems can often generalize to apply to data they have not encountered yet through machine learning, but not to new approaches to problems or to general reasoning. For example, a voice recognition system can interpret phrases it has never heard before, but cannot (without significant modification) learn an entirely new language or grammar. But artificial intelligence will continue to grow its reach as ever more specific models are constructed and stitched together. Machine learning is the stuff of these models, and the best path known to build artificially intelligent systems (whether those systems have specific intelligence like a speech recognition engine or general intelligence like Commander Data). As a field, machine learning employs many specific, non-mysterious algorithmic tools such as decision trees; rule learners; and classification techniques such as naïve Bayes, nearest-neighbor classification, neural networks, and support vector machines[180] because "in practice [] each of these algorithms is good for some things but not others."[181] And as a general matter, one group of computer scientists has noted within machine learning "some algorithms are more amenable to meaningful inspection and management than others."[182] Decision trees, naïve Bayes classification, and rule learners were the most interpretable, kNN was in the middle, and neural

[180] Singh et. al., *supra* note 26, at 4. A full explanation of these techniques is beyond the scope of this Paper. For a general background on these techniques and their limits *see* DOMINGOS *supra* note 123.
[181] DOMINGOS *supra* note 123, at Kindle Loc. 179.
[182] Singh et. al., *supra* note 26, at 4.

networks and support vector machines were the least interpretable.[183] We offer generally that machine learning systems need not be inscrutable.

Even if a system was not built with the goal of supporting analysis to start, there are ways to mitigate this fact. Any company using a machine learning approach to building some system needs to understand how that system works so that the company can manage the system and see how it performs in the field. This need opens the door to a type of evidence that can support outside analysis: the evidence the company uses to convince itself that the system is operating as desired could be reviewed for disclosure or made available to oversight entities. Along with evidence about how the software is built and run, this evidence can be convincing as to what system made a particular. In some cases a company may use an approach such as neutral networks that is not easy to interpret, even if its internals are fully disclosed. Nonetheless, the company could reduce that complexity by translating the inscrutable internals of the complicated model into a more interpretable decision tree system providing sufficiently similar decisions so that the company can work with and interpret the output more easily.[184] A related area of work looks to make machine learning interpretable.[185] This approach seeks to offer a way "to explain the predictions of any classifier or regressor" by "presenting textual or visual artifacts that provide qualitative understanding of the relationship between the instance's

[183] *Id*.
[184] While it is known that models designed for interpretability have only slightly lower performance than their inscrutable cousins, it is not known whether the additional errors are distributed evenly across all inputs or are biased towards a particular subgroup. That is, the interpretability meant to help the public understand whether the model is discriminating unduly may itself be a cause of discrimination in some cases. Research continues in this area, but we note that it should be possible to characterize the distribution of errors in any particular application, particularly in cases where more complex-but-inscrutable models already exist.
[185] For an example, *see* Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*, arXiv:1602.04938 [Cs, Stat], 16 February 2016, available at http://arxiv.org/abs/1602.04938.

components (e.g. words in text, patches in an image) and the model's prediction."[186] In addition, companies are constantly updating and building their software systems, and that provides the chance to modify the software as it evolves in ways that produce evidence to support outside analysis and ultimately promote accountability for automated decisions. Even when that is not possible, insofar as a company may need to explain what its system does, it should either use a method that is amenable to "meaningful inspection and management" or "interpretation," or—if using a method that is not as amenable—reduce that method to one that is better suited to "meaningful inspection and management" or "interpretation" while being careful to maintain fidelity to the original system both for actual decisions and for the distribution of errors.

There is an additional reason companies should offer ways for third parties to test and understand a company's software and its outputs. It may be that a dynamic system designed in the best of faith will yield an undesired result. If, however, a company finds ways for third parties to test its software and offers bounties for finding such outcomes (much as is done for security bugs), rather than an antagonistic policing game of "Gotcha!", in which critics cry foul in public, a more constructive norm of helping improve the system could emerge. For example, one popular image sharing site accidentally classified photos of African Americans under the tag "gorillas" using an automated system intended to determine the contents of a photograph.[187] A system of testing and bounties could ferret out such problems before they become problematic product issues and negative news stories.

---

[186] *Id.*

[187] *See e.g.*, Alastair Barr, *Google Mistakenly Tags Black People as 'Gorillas,' Showing Limits of Algorithms*. DIGITS BLOG. WALL STREET JOURNAL. 1 July 2015 at http://blogs.wsj.com/digits/2015/07/01/google-mistakenly-tags-black-people-as-gorillas-showing-limits-of-algorithms/

Of course, the use of evidence as a way to describe what systems are doing presupposes consensus on what the system should be doing. And absent methods for defining these characteristics precisely ahead of time, critics will still cast aspersions at systems they believe still act to produce discriminatory or illegal outcomes. For that possibility we offer that a whistleblower and public interest right of action law may be needed.

## C. Legislative Changes to Improve Accountability

Even with robust technical controls, companies using software-driven decision-making might still design their software to cause negative, discriminatory, or unfair outcomes while hoping that no one would know or that the perpetrators could deny that intent. Indeed, the precise outlines of what constitutes illegal or discriminatory behavior will always be the purview of messy, after-the-fact debates. Actors who wish to skirt the boundary of acceptable behavior may in some cases take liminal actions open to legitimate interpretation. Thus, we offer that, in addition to technical measures, policy-based controls are also appropriate. In particular, we propose that a whistleblower and public interest cause of action law would improve accountability.

Taken together, improving whistleblower protection and enabling a private right of action would help build a legal system to manage the problem of intentional, yet difficult to detect, discrimination via software. Given the federal government's roles as regulator and buyer, a federal whistleblower statute modeled on Sarbanes-Oxley gives

one way to think about whistleblower protection. Part of the reason for passing Sarbanes-Oxley was that unlike government employees, employees of publicly traded companies had no protection when they "act in the public interest by reporting wrongdoing, [that is] who blow the whistle."[188] Further, the Senate stated that "[w]ith an unprecedented portion of the American public investing in these companies and depending upon their honesty, this distinction does not serve the public good."[189] Similarly, with an unprecedented portion and an increasingly important portion of decision-making being processed through or influenced by software, with due process and vital verification interests at stake, there needs to be protection for employees who blow the whistle on software companies who knowingly violate the law. Other reasons behind passage of Sarbanes-Oxley further support a similar provision in the software context. First, without a federal law, a potential whistleblower is governed by whatever law, or lack of law, addresses the action in a given state or country where the whistleblower is.[190] Just as financial services companies can and do work across borders and set up subsidiaries or use other structures to shield against liabilities or prosecution, so too do software companies. Second, making retaliation a felony would increase protection for whistleblowers.[191] That protection won't by itself prevent negative repercussions such as blacklisting, but it may be possible to set up a bounty system so that a whistleblower will be able to cope with the difficulty in getting a new job after reporting violations.[192] When software that makes important decisions is faulty, issues of civil rights, due process, and the integrity of our democratic

[188] 148 Cong. Rec. S7418-01, S7420 (daily ed. July 26, 2002) (statement of Sen. Leahy).
[189] *Id*.
[190] *Id*.
[191] Sarbanes-Oxley Act of 2002 § 1107, 18 U.S.C. § 1513(e) (2012).
[192] *See* Dodd-Frank Wall Street Reform and Consumer Protection Act, § 922(a), 15 U.S.C. § 78u-6(b)(1)(A)-(B) (2012); *see also* Stavros Gadinis and Colby Mangels, *Collaborative Gatekeepers*, 73 WASH. & LEE L. REV. 797, 829 (2016) (explaining the problems of blacklisting and Dodd-Frank bounty system design as a way to mitigate that problem).

systems affect the United States government. First, the government is affected directly in its pocketbook (in that the government would have paid for illegal software, lost time and money, and have to remedy the failure). Second, the government suffers generally in its perception with U.S. citizens and even people around the world when it uses faulty software to make important decisions or when it stands by while such practices occur in the private sector. Protecting whistleblowers at a federal level shows a commitment by the government to take these issues seriously and to stop them.

On top of whistleblower protection, a public interest cause of action that balances the government's interest in pursuing a case against a private citizen's ability to do so, would aid governing the use of software.[193] Our model derives from *qui tam* actions under the False Claims Act of 1863,[194] which allows "any person to prosecute a civil fraud—in the name of the United States—against any person who allegedly makes a false

[193] Some argue that such an approach causes an "explosion" of or "gold rush" effect on litigation, but a recent empirical study indicates such claims are overstated. *See* David Freeman Engstrom, 114 COLUMBIA L. REV. 1913, 1922 (2014).

[194] See Julie Ann Ross, *Citizen Suits: California's Proposition 65 and the Lawyer's Ethical Duty to the Public Interest*, 29 U.S.F. L. REV. 809, 810 n.4 (1995) (noting relationship of qui tam to "modern day citizen suit"); accord Michael Ray Harris, *Promoting Corporate Self-Compliance: An Examination of the Debate Over Legal Protection for Environmental Audits*, 23 ECOLOGY L.Q. 663, 710, n. 317 ("A growing trend in environmental law is the use of the qui tam provisions of the False Claims Act of 1863 (FCA) to force corporations to abide by environmental reporting requirements."). The False Claims Act is codified at 31 U.S.C. §§ 3729-3732. It should be noted that the use of qui tam actions is subject to some debate and criticism. See Trevor W. Morrison, *Private Attorneys General and the First Amendment*, 103 MICH. L. REV. 589, 607-618 (2005) (examining private attorney general actions and noting pro arguments--"[P]rivate attorneys general are a cost-effective means of both pursuing the public welfare and returning power to the people themselves. For legislatures that value cheap, robust regulatory enforcement, private attorneys general may present an attractive option." Id. at 610. Arguments against include: private attorneys may be self-interested and motivated by financial gain, may free-ride by waiting to follow the government's lead for opportunistic litigation rather than being true investigators, may accept settlements too easily, and may undermine the provision of consistent agency or government enforcement. Id. at 607-619 (finding "no definitive resolution to the policy debate over private attorneys general" and that legislatures resolve the matter differently based on context.)); John Beisner, et. al., *Class Action Cops: Public Servants or Private Entrepreneurs?*, 57 STAN. L. REV. 1441, 1457-1460 (2005) (comparing class action private attorney general actions to qui tam suits and noting that though qui tam actions may allow citizens to bring suits that abuse or are not part of the policy objectives behind the act enabling a given suit, as opposed to class actions, the legislature can alter the parameters of the enabling act to prevent such abuses).

claim to the United States government"[195] and California's Safe Drinking Water and Toxic Enforcement Act of 1986, which addresses environmental concerns by allowing a public interest cause of action balanced against governmental interests in pursuing such matters.[196] To work for software, this model would have to be precise about what would be subject to the public cause of action, provide for the ability to file suit, and set out the procedure under which such a suit would be filed. We think that the clearest examples of when a public cause of action should be allowed would be in public sector use of software written by an outside vendor and regulated private sector cases.[197] Voting machines and the auto industry offer good examples of how this approach could work.

Voting machines and software in cars are excellent examples of software systems that need to be trusted and tested to ensure that they function as desired. A public interest cause of action statute to govern either industry would enable government and private oversight by including injunctive remedies, specific and daily civil penalties, detailed notice and minimum threshold requirements for private action, and time limits to allow a public monitoring function to exist while maintaining the government's ability to pursue cases.[198] The California approach sets the penalty at $2,500 per violation. Some statutory amount is needed because no money is at issue under California law for the behavior in question. We do not take a position on the correct amount for statutory penalties for software harms. Rather we note that the amount must be large enough to provide an

[195] Saikrishna Prakash, The Chief Prosecutor, 73 GEO. WASH. L. REV. 521, 531 (2005) (examining the history of qui tam actions and the executive's final authority in such actions).
[196] *See generally* Cal. Health & Safety Code § 25249.5-.13 (West 1999 & Supp. 2005).
[197] If the government built its software, there may be sovereign immunity issues that prevent the lawsuit by private citizens. That would not undermine a legislature requiring software that is analyzable and that the process in which it is used supports accountability for decisions made. It only undermines the use of private civil actions as a mechanism for policing the software's correctness.
[198] *See e.g.*, Cal. Health & Safety Code § 25249.5-.13 (West 1999 & Supp. 2005) (using the same levers for environmental protection).

incentive for the action. Staying with our example industries, given the number of voting machines in use and autos on the road, $2,500 may be sufficient. In other cases where important but lower scale use of software is at issue, the statute may need to set a higher amount. Regardless of the amount set, with civil penalties, courts would look at (A) The nature and extent of the violation; (B) The number of, and severity of, the violations; (C) The economic effect of the penalty on the violator; (D) Whether the violator took good faith measures to comply with regulations and the time these measures were taken; (E) The willfulness of the violator's misconduct; (F) The deterrent effect that the imposition of the penalty would have on both the violator and the regulated community as a whole; (G) Any other factor that justice may require.[199] If the statute is passed by a state, as a way to limit frivolous suits, only an Attorney General, a district attorney, or a city attorney for a city with a sufficient population of (perhaps more than 750,000 people) could bring a suit on their own.[200] Government attorneys in smaller jurisdictions could bring suits provided they receive permission of their district attorney.[201]

Under this approach, non-governmental actors can bring an action in the public interest under certain conditions.[202] To start, the potential plaintiff would have to give notice to the government attorney with jurisdiction over the potential case and to the alleged violator of the Act and wait out a period of deferral to the government attorney with jurisdiction (this period is sixty days in the California law) before bringing the suit.[203] The waiting period could be shorter or longer, but it allows the state to decide

[199] *Cf.* Id. § 25249.7(b)(2).
[200] *Cf.* Id. § 25249.7(c).
[201] *Cf.* Id.
[202] *Cf.* Id. § 25249.7(d).
[203] *Cf.* Id. § 25249.7(d)(1).

whether to act. If the State picked up the case and was "diligently prosecuting" it, the private action would no longer be allowed.[204]

As another way to limit frivolous lawsuits, the person bringing the suit would have to provide a certificate of merit which would verify that the person executing the certificate had consulted with one or more persons with relevant and appropriate experience or expertise who reviewed facts, studies, or other data regarding the claimed violation that is the subject of the action, and that, based on that information, the person executing the certificate believes there is a reasonable and meritorious case for the private action.[205] Furthermore "factual information sufficient to establish the basis of the certificate of merit . . . [would be required to] be attached to the certificate of merit that is served on the Attorney General."[206] The factual material in the certificate would not be discoverable unless it was "relevant to the subject matter of the action and is otherwise discoverable,"[207] which would help protect potential whistleblowers who might provide such material.

After a case is over, however, the alleged violator may move for, or the court of its own volition may conduct, an in camera review of the certificate of merit and the legal theories of the plaintiff to see whether the basis was real.[208] If the court deems the basis to be false, it may bring sanctions for a frivolous lawsuit.[209] California's provision on which we draw states, "If the court finds that there was no credible factual basis for the certifier's belief that an exposure to a listed chemical had occurred or was threatened,

[204] *Cf.* Id. § 25249.7(d)(2).
[205] *Cf.* Id. § 25249.7(d)(1).
[206] *Cf.* Id.
[207] *Cf.* Id. § 25249.7(h)(1).
[208] *Cf.* Id. § 25249.7(h)(2).
[209] Id.

then **the action shall be deemed frivolous** within the meaning of Section 128.6 or 128.7 of the Code of Civil Procedure, whichever provision is applicable to the action."[210] These two sections and their adaptation to our approach are vital, because one section allows the court to order the instigator of the frivolous lawsuit to pay attorney's fees and reasonable expenses,[211] and the other section provides authorization for the court to impose sanctions and punitive damages including but not limited to attorney's fees with the stated goal of having the sanctions deter frivolous suits.[212] A heightened pleading requirement or procedural limit, such as exhausting certain remedies, might be incorporated to limit further concerns regarding an explosion of suits. In addition, the statute could require a certain level of pattern and practice before such a suit would be allowed thereby preventing numerous small-scale, nuisance suits and focusing on larger scale systemic issues.

## CONCLUSION

During the Cold War, Ronald Reagan liked to use a Russian proverb, "trust but verify," as a way to describe his approach to U.S.-Soviet relations; today we need to trust automated systems to make many critical decisions, but must also verify that big data and sophisticated technologies do not raise new problems for those subject to automation. We can and should trust that many computer scientists and engineers wish to use their skills to improve the world. Yet, in the public sector, difficulty in understanding how and why

[210] Id.
[211] Cal. Civ. Proc. Code § 128.6(a) (West Supp. 2005).
[212] Cal. Code Civ. Proc. § 128.7(h) (West Supp. 2005) ("It is the intent of the Legislature that courts shall vigorously use its sanctions authority to deter that improper conduct or comparable conduct by others similarly situated.").

decisions are made—especially in the occasional cases where no answer is available—fuels distrust. For example, concerns over the integrity of voting machine counts make it harder to trust election outcomes.[213] In the private sector, evidence of misuse of software by the auto industry, such as with Volkswagen and recently at Fiat-Chrysler, and concerns around discrimination in insurance markets, credit scoring, and private systems used in the administration of criminal justice erode trust and increases general fears about software. In addition, while experiments have tested how news or so-called fake news is promoted on social networks and how online ads may be targeted in ways that are discriminatory or illegal, those experiments cannot find a consistent or clear answer as how or why the results occurred. That, too, eats away at the public's ability to trust these systems. The classic response to these types of problems is a demand for transparency. Although such an approach seems reasonable—once one can see all that went on in the process, one can root out bad behavior and verify that rules have been followed—it cannot function alone to meet the goals its proponents wish to advance. As we have shown, transparency is a useful goal, and will in many cases be necessary, but it does not solve these problems.

Instead, by understanding what can and cannot be done when evaluating software systems, and by demanding convincing evidence that systems are operating correctly and within the bounds set by law, society can allow the use of sophisticated software techniques while also having meaningful ways to ensure that these systems are governable. In some cases, determining compliance and effecting governance will require oversight by a competent authority, in which case software systems must create sufficient

[213] A proverb popular among election security researchers and attributed to computer security expert Dan Wallach is "The purpose of an election is not to name the winner, it is to convince the losers that they lost." This underscores the importance of integrity as a security concern.

audit trails to support that oversight. As we have shown, although current software systems pose large challenges for those who wish to understand how they operated, computer science offers a way out for software engineered to provide such assurances. One can require that software be built to produce evidence that allows it to be analyzed, and the facts gleaned from this analysis can promote accountability. Or rather, software can be built so that we can trust, but verify.