HARVARD JOURNAL OF LAW & TECHNOLOGY

VOLUME 31, NUMBER 1 FALL 2017

CONTENTS

ARTICLE

TRUST BUT VERIFY: A GUIDE TO ALGORITHMS AND THE LAW

Deven R. Desai and Joshua A. Kroll***

TABLE OF CONTENTS

* 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, and from attendees at Privacy Law Scholars Conference, 2016 at George Washington University Law and at the Law and Ethics of Big Data Colloquium at University of Indiana, Bloomington, Kelley School of Business. I thank Jason Hyatt for excellent research assistance. This Article was supported in part by summer research funding 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 author alone and do not necessarily reflect the view of those who helped with and supported this work.

** Postdoctoral Research Scholar, UC Berkeley School of Information.

I. INTRODUCTION

> ACCORDING TO MY DEFINITION, A NUMBER IS COMPUTABLE IF ITS DECIMAL CAN BE WRITTEN DOWN BY A MACHINE.
>
> — ALAN TURING[1]

> IN 1953, HENRY RICE PROVED THE FOLLOWING EXTREMELY POWERFUL THEOREM, WHICH ESSENTIALLY STATES THAT EVERY INTERESTING QUESTION ABOUT THE LANGUAGE ACCEPTED BY A TURING MACHINE IS UNDECIDABLE.
>
> — JEFF ERICKSON[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, 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.
>
> — IAN BOGOST[3]

Someone is denied a job.[4] A family cannot get a loan for a car or a house.[5] Someone else is put on a no-fly list.[6] A single mother is denied federal benefits.[7] None of these people knows why that happened other

---

1. A. M. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem*, 42 PROC. LONDON MATHEMATICAL SOC'Y 230, 230 (1936).

2. JEFF ERICKSON, MODELS OF COMPUTATION 10 (2015) (ebook).

3. Ian Bogost, *The Cathedral of Computation*, THE ATLANTIC (Jan. 15, 2015), http://www.theatlantic.com/technology/archive/2015/01/the-cathedral-of-computation/384300/ [https://perma.cc/AA6T-3FWV].

4. *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).

5. *See, e.g.*, *id.* at 4–5 (discussing use of predictive analytics in credit scoring and loan decisions).

6. *See, e.g.*, Danielle Keats Citron, *Technological Due Process*, 85 WASH. U. L. REV. 1249, 1256–57 (2008).

7. *See, e.g.*, *id.*

than the decision was processed through some software.[8] Someone commandeers a car, controls its brakes, and even drives away.[9] A car company claims its cars have low emissions, but in fact its cars pollute significantly.[10] A voting machine is supposed to count votes accurately, but no one can tell whether the count is correct.[11] A car's battery seems not to have a good range, so its software is updated, but no one knows whether the update has fixed the problem or is compliant with government regulations.[12] Searches for black-sounding names yield ads suggestive of arrest records.[13] Why these things happen and whether the companies using the software have complied with regulations or used software to commit fraud is difficult to determine. In all of these cases, a common concern is that the algorithms or, more precisely, the software behind the decision or process, are at fault.[14] Often the claim is that society cannot understand or govern these outcomes because the decision process is a black box.[15]

A consistent theme is that unaccountable machines have taken center stage and now "are used to make decisions for us, about us, or with

---

8. *See, e.g.*, PASQUALE, *supra* note 4, at 4–5 (explaining that one "will never understand exactly how [one's credit score] was calculated"); *infra* Part II.

9. 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 stolen. *See* Davis Z. Morris, *Tesla Stealing Hack Is About Much More than Tesla*, FORTUNE (Nov. 26, 2016), http://fortune.com/2016/11/26/tesla-stealing-hack/ [https://perma.cc/GQ2N-Y3XC]. In one case, a group took over the car's braking system (and more) from 12 miles away. *See* Andrea Peterson, *Researchers Remotely Hack Tesla S*, WASH. POST (Sept. 20, 2016), https://www.washingtonpost.com/news/the-switch/wp/2016/09/20/researchers-remotely-hack-tesla-model-s/?utm_term=.7fc39f9f20f9 [https://perma.cc/X3EL-CFZM].

10. *See, e.g.*, Russel Hotten, *Volkswagen: The Scandal Explained*, BBC NEWS (Dec. 10, 2015), http://www.bbc.com/news/business-34324772 [https://perma.cc/H89Z-LNPC] (explaining how Volkswagen used software to fake emissions results); Alex Davies, *Here We Go Again: EPA Accuses Fiat Chrysler of Selling Dirty Diesels*, WIRED (Jan. 12, 2017, 4:06 PM), https://www.wired.com/2017/01/epa-now-accusing-fiat-chrysler-selling-dirty-diesels/ [https://perma.cc/5DYJ-NUQC] (noting that EPA accused Fiat Chrysler of installing and not disclosing software that hid nitrous oxide emissions in its diesel cars); *see also* Moritz Contag et al., *How They Did It: An Analysis of Emission Defeat Devices in Modern Automobiles*, 38 IEEE SYMP. ON SEC. & PRIVACY 231 (2017) (providing a detailed technical description of the emissions defeating software). For details of the charges against one of the engineers at Volkswagen, see Indictment at 15–23, United States v. Liang, No. 2:16-cr-20394, 2016 WL 5542732 (E.D. Mich. 2016).

11. *See, e.g.*, J. Alex Halderman, *Want to Know if the Election Was Hacked? Look at the Ballots*, MEDIUM (Nov. 23, 2016), https://medium.com/@jhalderm/want-to-know-if-the-election-was-hacked-look-at-the-ballots-c61a6113b0ba#.gzpyt1dat [https://perma.cc/9M7Y-PTRR].

12. *See, e.g.*, Alex Davies, *Tesla's Plan to Kill Range Anxiety with a Software Update*, WIRED (Mar. 19, 2015, 2:01 PM), https://www.wired.com/2015/03/teslas-plan-kill-range-anxiety/ [https://perma.cc/6YGY-7APR].

13. *See, e.g.*, Latanya Sweeney, *Discrimination in Online Ad Delivery*, 56 COMM. ACM, May 2013, at 44, 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.").

14. *See infra* Part II

15. *See* PASQUALE, *supra* note 4, at 165.

us,"[16] in sensitive and subjective areas such as healthcare, employment, credit, national security, networked devices, news, and more.[17] A more recent fear is that the rise of large data sets combined with machine learning ("ML") (an area of computer science that uses the automated discovery of correlations and patterns to define decision policies) might allow those who use such techniques to wield power in ways society prohibits or should disfavor, but which society would not be able to detect.[18] Further, if a computer yields undesired results, its programmers may say that the system was not designed to act that way.[19]

The standard solution to this general problem is a call for transparency, which in this context has been called "algorithmic transparency."[20] We argue that although the problems are real, the proposed solution will not work for important computer science reasons. 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, where algorithms might as well be God.[21] Both critics and advocates can stray into uncritical deference to the idea that big data and the algorithms used to process the data are somehow infallible science. We believe this problem is aggravated because although algorithms are decidedly *not* mystical things or dark magic, algorithms are not well understood outside the technical community.[22]

---

16. CTR. FOR INTERNET & HUMAN RIGHTS, THE ETHICS OF ALGORITHMS: FROM RADICAL CONTENT TO SELF-DRIVING CARS 1 (2015).

17. *See, e.g.*, PASQUALE, *supra* note 4, at 4; *cf.* Bogost, *supra* note 3 (explaining that deference to algorithms resembles idolatry rather than following Enlightenment skepticism).

18. *See, e.g.*, FED. TRADE COMM'N, BIG DATA: A TOOL FOR INCLUSION OR EXCLUSION? 1 (2016); CTR. FOR INTERNET & HUMAN RIGHTS, *supra* note 16, at 1; PASQUALE, *supra* note 4, at 4.

19. *Cf.* Solon Barocas & Andrew D. Selbst, *Big Data's Disparate Impact*, 104 CALIF. L. REV. 671, 674–75 (2016) (explaining that algorithms may unintentionally increase discrimination because of problems with data mining).

20. S*ee, e.g.*, Katherine Noyes, *The FTC Is Worried About Algorithmic Transparency, and You Should Be Too*, PC WORLD (Apr. 9, 2015, 8:36 AM), http://www.pcworld.com/article/2908372/the-ftc-is-worried-about-algorithmic-transparency-and-you-should-be-too.html [https://perma.cc/N3Z2-5M3E] (discussing Christian Sandvig's view that transparency may not be viable because of the complexity of some algorithms and the data needed to test the algorithms). For Sandvig's position on algorithmic transparency, see Christian Sandvig et al., *Auditing Algorithms: Research Methods for Detecting Discrimination on Internet Platforms*, (May 22, 2014), http://www-personal.umich.edu/~csandvig/research/Auditing%20Algorithms%20--%20Sandvig%20--%20ICA%202014%20Data%20and%20Discrimination%20Preconference.pdf [https://perma.cc/GJS4-YWP3] (using "social scientific study" auditing to investigate algorithmically driven platforms).

21. *See* Bogost, *supra* note 3.

22. *See id.* ("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 et al., *Accountable Algorithms*, 165 U. PA. L. REV. 633, 640 n.14 (2016) ("The term 'algorithm' is

Put differently, transparency is a powerful concept and has its place. After all, who can argue against sunlight? And yet, to an extent, we 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.[23] For example, as discussed *infra*, fundamental limitations on the analysis of software meaningfully limit the interpretability of even full disclosures of software source code. This Article thus examines the idea of algorithmic transparency, offers a primer on algorithms as a way to bridge this gap, and presents concrete options for managing problems automated decision-making presents to society.

This Article begins 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, the Article turns 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.[24] Having set out a brief primer on the underlying computer science, the Article addresses 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, the Article reviews the

---

assigned disparate technical meaning in the literatures of computer science and other fields . . . .").

23. It is common for open-source advocates to cite "Linus's Law" — the dictum that "[g]iven enough eyeballs, all bugs are shallow" — meaning that transparency of code to a sufficient number of experts implies that any problem will seem obvious to someone and can be remedied. ERIC S. RAYMOND, THE CATHEDRAL AND THE BAZAAR 9 (1999). However, many pieces of open-source software struggle to get the attention of enough skilled reviewers, creating what has been called a "major eyeball shortage." Edward W. Felten & Joshua A. Kroll, *Heartbleed Shows Government Must Lead on Internet Security*, SCI. AM. (July 1, 2014), https://www.scientificamerican.com/article/heartbleed-shows-government-must-lead-on-internet-security1/ [https://perma.cc/X7WD-CUZJ]. The much-publicized "Heartbleed" vulnerability provides an object lesson: a trivially simple coding error in a widely used security tool exposed the private information of the majority of web servers on the Internet, including encryption keys, website passwords, and sensitive user information, for several widely used web applications. For a detailed account, see Zakir Durumeric et al., *The Matter of Heartbleed*, 14 ACM INTERNET MEASUREMENT CONF. 475 (2014). Contrast the situation in systems that use machine learning, where the rule that is being applied by a model may not be understandable, even if the system's developer knows that the model is performing well. *See* Jatinder Singh et al., *Responsibility & Machine Learning: Part of a Process*, (Oct. 27, 2016), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2860048 [https://perma.cc/EDG2-TNTF] ("[A]lgorithmic 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." (emphasis omitted)).

24. *See, e.g.*, Sandvig et al., *supra* note 20, at 3.

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 propose a legislative change to protect whistleblowers and allow a public interest cause of action as a way to aid in increasing detection of overt misdeeds in designing software. In short, a better understanding of how programs work and how computer scientists address the limits of their field creates tools to manage the evolving world of algorithms and the law. This, in turn, allows society to 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.

Thus, in contrast to the current approaches to governance by auditing to find unacceptable behaviors, 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 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 what 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 do not work. And that discussion will use the framework and terms within which they already operate, avoiding charges of 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.

II. ALGORITHMS: THE CONCERNS

Software and algorithms have gained much attention under the premise that they "exercise power over us"[25] because they "select[] what information is considered most relevant to us, a crucial feature of our participation in public life,"[26] are "powerful entities that govern, judge, sort, regulate, classify, influence, or otherwise discipline the

25. NICHOLAS DIAKOPOULOUS, ALGORITHMIC ACCOUNTABILITY REPORTING: ON THE INVESTIGATION OF BLACK BOXES 2 (2014) (emphasis omitted).

26. Tarleton Gillespie, *The Relevance of Algorithms*, *in* MEDIA TECHNOLOGIES: ESSAYS ON COMMUNICATION, MATERIALITY, AND SOCIETY 167, 167 (Tarleton Gillespie et al. eds., 2014).

world,"[27] and are "black boxes."[28] In short, the general idea that computer systems are powerful and opaque has led to claims "that virtually any algorithm may deserve scrutiny."[29] Varying reports and studies raise concerns about complex systems and point to the more general concern: the possible powers and avenues of abuse or manipulation that go with practices that use computers. Despite commenters' different methods and concerns, transparency is often raised as a key part of managing this new order, because one "cannot access critical features of [its] decision-making processes."[30] 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. These issues arise in both the public and private sector context. We start by examining transparency and accountability from a legal-political view and a computer science view, proceed to some examples of the public sector concerns, and then turn to private sector ones.

A. Transparency and Accountability: Two Complementary Views

Both legal-political and computer science scholars wish to ensure that automated decision systems are not enabling misdeeds or generating undesired outcomes. Both fields use the terms transparency and accountability, but have different meanings and functions for them. This vocabulary clash can muddy the understanding of what is desired as an end result and of how to create systems to realize whatever end result is agreed upon. As such, this section parses the different, yet related, aspects of the terms.

There is a deep, unstated and powerful view in the law and much of society in general; the builder of the proverbial better mousetrap will know precisely how it was built and what will happen when one presses

27. Solon Barocas, Sophie Hood & Malte Ziewitz, *Governing Algorithms: A Provocation Piece*, GOVERNING ALGORITHMS ¶ 9 (Mar. 29, 2013), http://governingalgorithms.org/resources/provocation-piece/ [https://perma.cc/M8RY-73YX].

28. PASQUALE, *supra* note 4, at 17.

29. Sandvig et al., *supra* note 20, at 3.

30. PASQUALE, *supra* note 4, at 17. The call for or desire to have transparency about the inner workings of automated decision systems as a way to resolve issues around outcomes from those systems can be strong. For example, Professor Latanya Sweeney has done work on racial discrimination and advertising. *See* Sweeney, *supra* note 13. As Professor Cynthia Dwork noted when interviewed about that work, "[t]he examples described in that paper raise questions about how things are done in practice." *See* Claire Cain Miller, *Algorithms and Bias, Q&A with Cynthia Dwork*, N.Y. TIMES: THE UPSHOT (Aug. 10, 2015), https://www.nytimes.com/2015/08/11/upshot/algorithms-and-bias-q-and-a-with-cynthia-dwork.html (last visited Dec. 19, 2017). 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.

a trigger or button in the invention.[31] The device will do the same thing over and over until the springs wear out. The related presumption is that if a layperson who did not build the mousetrap has the plans, that person, perhaps with the aid of a hired expert, will be able to understand how the mousetrap works and probe the plans for flaws.[32] The same, so reasons the law, must be true of software. As we shall see, in many ways, it is, and in some ways, it is not. Nonetheless, the key idea is that seeing the internals of a system leads to understanding of the workings of that system and the consequences associated to the system's operation. This is the core of the law's conception of transparency, and it is deployed to serve many goals.

Transparency has been proposed as a solution to mitigating possible undesired outcomes from automated decision-making. Even before today's fascination with big data, algorithms, and automated systems, legal scholars such as Paul Schwartz and Danielle Citron identified important ways in which data processing and software used in the administrative state can undermine or take away due process rights.[33] 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.[34] Transparency in this context is the claim that someone "ought to be able to 'look under the hood' of highly advanced technologies like . . . algorithms"[35] as a way to police such behavior.[36] It is the idea that with the plans to the mousetrap, one can identify flaws, willful errors, and perhaps ferret out undesired and possibly unintended results such as discrimination in a hiring decision.[37] Thus, different critics audit parts of systems, decry apparent discrimination, and want to hold someone responsible for bad outcomes.[38] This approach is tantamount to saying that the way to get proof that the algorithm is not designed to engage in, nor has parts of it that lead to, discrimination or

---

31. *Cf.* DAVID NYE, TECHNOLOGY MATTERS: QUESTIONS TO LIVE WITH 162 (2006) ("By c. 1900 [people] increasingly demanded scientific explanations . . . . Engineers could explain how a bridge failed or what component of a steam engine was responsible for a burst boiler.")

32. *Id.* ("Formerly inscrutable events became legible to the safety engineer, the tort lawyer, the insurance agent, and the judge.").

33. Paul Schwartz, *Data Processing and Government Administration: The Failure of the American Legal Response to the Computer*, 43 HASTINGS L.J. 1321, 1343–74 (1992); Citron, *supra* note 6, at 1281–88 (discussing how automation threatens citizens' due process rights such as notice and opportunity to be heard).

34. *See infra* Part II.B.

35. PASQUALE, *supra* note 4, at 165; *cf.* Citron, *supra* note 6, at 1308 ("Automated systems must be designed with transparency and accountability as their primary objectives, so as to prevent inadvertent and procedurally defective rulemaking . . . . [V]endors should release systems' source codes to the public. Opening up the source code would reveal how a system works.").

36. *But see* Noyes, *supra* note 20.

37. *Cf.* Barocas & Selbst, *supra* note 19, at 694 (arguing that "[c]urrent antidiscrimination law is not well equipped to address the cases of discrimination stemming from" data mining and related algorithmic practices in employment).

38. *See infra* Section II.B.

other undesired or prohibited acts is to review its internals.[39] From this proof, then, comes the ability to hold the system's creator or operator accountable.

Put differently, if one can see into the system and identify bad behaviors and outcomes, one can hold someone accountable in the legal-political sense of the word. Accountability as a legal-political concept is about openness regarding the way government operates so that the people may know what its representatives are doing, hold government responsible, and participate in government by voting, expressing their views to representatives and regulators, or influencing policymaking in a variety of ways.[40] As David Levine explored a decade ago, trade secrets — such as those surrounding voting machines' software — clash with political accountability.[41] If one cannot determine whether rules have been followed, one cannot engage through the legal-political system to advocate for change. Specifically, one cannot hold someone responsible through lawsuits or by voting someone out of office. In short, accountability about software as conceptualized by law scholars such as Schwartz,[42] Levine,[43] Citron,[44] and Pasquale[45] is traditionally about accountability that enables holding political and private actors responsible.

The desire to see how a system works and to understand the system relates to more than just the machines used to make or aid in making decisions. Entire decision-making processes, whether operated by the state or by private entities, fit this view. As Jerry Mashaw has argued for administrative state systems, machines that make significant decisions should make accurate and cost-effective judgments.[46] However, they should also give "attention to the dignity of the participants."[47] The dignity element requires that those who are subject to such a process know or understand what reasons are behind a decision.[48] In that sense,

---

39. *See infra* Section II.B.

40. *See* David Levine, *Secrecy and Accountability: Trade Secrets in Our Public Infrastructure*, 59 FLA. L. REV. 135, 180 (2007); Daniel J. Weitzner et al., *Information Accountability*, 51 COMM. ACM, June 2008, at 82, 86 (arguing for a technical architecture that enables holding "the individuals and institutions who misuse it accountable for their acts").

41. Levine, *supra* note 40, at 158 (discussing transparency and accountability as foundational democratic values in contrast to secrecy).

42. Schwartz, *supra* note 33.

43. Levine, *supra* note 40.

44. Citron, *supra* note 6.

45. PASQUALE, *supra* note 4.

46. JERRY L. MASHAW, BUREAUCRATIC JUSTICE 26 (1983).

47. Schwartz, *supra* note 33, at 1348; *accord* MASHAW, *supra* note 46, at 95–96.

48. *See* Schwartz, *supra* note 33, 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 has stated that dignity must be protected as a 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 2 (2015). The European Data Protection Supervisor is the office responsible "with respect to the processing of personal data . . . for ensuring that the fundamental rights and freedoms of natural persons,

the demand for dignity presupposes 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 such as the way online ads are delivered, the prices online shoppers are shown, and whether a car is performing as promised[49] — do not map well to this assumption.

Thus, we argue that demands for transparency must confront the realities of computer science considering 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."[50] As a legal matter, using such methods to determine whether a prohibited practice has occurred to an actionable extent, presents problems, because it is difficult to measure which inputs are the important ones to test.[51]

In addition, handing over code often will not enable the political accountability results those in favor of so-called algorithmic transparency desire.[52] For example, simply disclosing or open-sourcing source code does nothing to show that the disclosed software was used in any particular decision unless that decision can be perfectly replicated from the disclosures. And that is rarely the case. With this in mind, we turn to the computer science view of accountability.

Computer science accountability, by contrast, is a technical concept about making sure that software produces evidence allowing oversight and verification of whether it is operating within agreed-upon rules.[53] For example, if one cannot determine whether a credit bureau adhered to the Fair Credit Reporting Act's restrictions on data gathering and use, one would have trouble detecting whether the credit bureau disobeyed the law.[54] Similarly, if one wishes to ensure that a system for counting votes or allocating visas in a lottery is doing what it is supposed to do, one needs a meaningful technical way to look under the

---

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 the processing of personal data." Council Regulation 45/2001, art. 41, 2001 O.J. (L 8) 18.

49. *See infra* Section II.C.

50. Kroll et al., *supra* note 22, at 650.

51. *See infra* Part IV.

52. *See infra* Part IV.

53. *See, e.g.*, Andreas Haeberlen, Petr Kuznetsov & Peter Druschel, *PeerReview: Practical Accountability for Distributed Systems*, 41 ACM SIGOPS OPERATING SYS. REV. 175, 175 (2007) ("[A]n accountable system maintains a tamper-evident record that provides non-repudiable evidence of all nodes' actions. Based on this record, a faulty node whose observable behavior deviates from that of a correct node can be detected eventually. At the same time, a correct node can defend itself against any false accusations."); Kroll et al., *supra* note 22, at 662–65.

54. *Cf.* Daniel J. Weitzner et al., *supra* note 40, at 86 (advocating for technical architecture that enables detecting whether someone has complied with laws governing data use such as in the credit bureau context).

hood.[55] Thus, "technical accountability" requires that systems generate reliable evidence to verify the system functioned correctly. Such evidence must have integrity, which in this context means the production of "a tamper-evident record that provides non-repudiable evidence" of relevant actions by the automated system.[56] Such evidence would provide records of what actions were taken and why, with a focus on how that evidence will be used to hold the system's creators or operators accountable for those actions. Of course, evidence that satisfies the requirements of technical accountability by itself does not provide legal-political accountability.[57] Rather, technical accountability enables legal-political accountability by providing a way to understand whether, how, to what extent, and why misdeeds occurred, as well as who (or what part of a system) is responsible for them. It is then up to political and legal processes to use that evidence to hold actors responsible, meting out punishments when warranted. And as Kroll et al. note, where rules cannot be agreed upon in advance because of the necessary ambiguities in policymaking, oversight may be necessary as the mechanism for defining the specific contours of rules ex post, even as system designers do their best to comply with their view of the rules ex ante.[58]

In short, technical accountability is a necessary step to enable political accountability. For it is only after one has verifiable evidence, relevant to the inquiry at hand, that one can have the possibility of holding both public and private actors who use automated systems responsible for their actions.[59]

None of the above means those who use 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.[60] Many of the current calls for transparency as a way to regulate automation do not address such limits, and so they may come up short on providing the sort of legal-political accountability they desire, and which we also support.[61] Instead, as software (and especially machine learning systems, which separate the creation of algorithms and rules from human design and implementation) continues to

---

55. *See, e.g.*, Halderman, *supra* note 11.

56. Haeberlen et al., *supra* note 53, at 175.

57. *See* Weitzner, et al., *supra* note 40, at 84 ("Transparency and accountability make bad acts visible to all concerned. However, visibility alone does not guarantee compliance.").

58. Kroll et al., *supra* note 22, at 678.

59. *See* Weitzner, et al., *supra* note 40, at 84 ("[W]e are all aware that we may be held accountable through a process that looks back through the records of our actions and assesses them against the rules.").

60. *See infra* Parts III, IV.

61. As one group of computer scientists has noted within machine learning, "[s]ome ML algorithms are more amenable to meaningful inspection . . . and management than others." Singh et al., *supra* note 23, at 4 (offering that decision tree, naïve Bayes, and rule learners were the most interpretable, [*k*-nearest neighbors] was in the middle, and neural networks and support vector machines were the least interpretable).

grow in importance, society may find, and we argue, that identifying harms, prohibiting outcomes, and banning undesirable uses is a more promising path.[62] In addition, in some cases, we argue that society may require that software be built to certain specifications that can be tested or verified.

B. 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.[63] Legal scholars have looked at governmental use of computerized decision-making and identified that software can aid the way the administrative state functions but at the same time run afoul of justice and due process requirements.[64]

Twenty-five years ago, Schwartz noted that "[c]omputers are now an integral part of government administration."[65] 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."[66] When the state uses data to administer services, however, we want administration that "carries out legislative policy, acts in a just manner, and combats fraud."[67] Schwartz

---

62. *Cf.* FED. TRADE COMM'N, *supra* note 18, at 5–12 (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 data analytics alone).

63. *See* EXEC. OFFICE OF THE PRESIDENT, BIG DATA: SEIZING OPPORTUNITIES, PRESERVING VALUES 10 (2014) ("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. *See* Curtis Publ'g. Co. v. Butts, 388 U.S. 130, 163 (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 than by "formal political institutions"); Deven R. Desai, *Speech, Citizenry, and the Market: A Corporate Public Figure Doctrine*, 98 MINN. L. REV. 455, 467 (2013) ("[T]he 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.

64. *See, e.g.*, Schwartz, *supra* note 33, at 1325; Citron, *supra* note 6, at 1256–57.

65. Schwartz, *supra* note 33, at 1322.

66. *Id.* at 1332; *see also* FRANK WEBSTER, THEORIES OF THE INFORMATION SOCIETY 208 (John Urry ed., 3d ed. 1995) ("If 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 that 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).

67. Schwartz, *supra* note 33, at 1333.

examined the Aid to Families with Dependent Children Program and Child Support Enforcement programs as exemplars of the administrative state, and he argued that the nature of data processing undermined the ability to attain bureaucratic justice as developed by Mashaw and the ability to protect autonomy.[68] In that vision, the system should not only make accurate, cost-effective judgments, but also give "attention to the dignity of participants."[69] 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 goals."[70] The dignity element requires that those who are subject to such a process know or understand what reasons are behind a decision.[71] 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.[72]

Danielle Citron's work also calls out the way that computers have been used in the administrative state, focusing on due process concerns.[73] She describes the "automated administrative state"[74] as using software to determine whether someone should receive "Medicaid, food stamp, and welfare" benefits, be on a no fly list, or be identified as owing child support.[75] According to Citron, "[a]utomation 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."[76]

Although these scholars use different metrics to argue that the use of software and computers is a problem, both identify the problem sphere as the administrative state.[77] And both Schwartz and Citron look to transparency as part of how to address whether the state uses data

68. *See id.* at 1360.
69. *Id.* at 1348.
70. MASHAW, *supra* note 46, at 49.
71. *See* Schwartz, *supra* note 33, at 1348–49.
72. *See id.*
73. Citron, *supra* note 6, at 1256–57.
74. *Id.* at 1281.
75. *Id.* at 1256–57.
76. *Id.* at 1281.
77. Scholars have examined software and accountability in the computer science context and found that the administrative state is a prime example of where algorithmic governance is needed. *See* Kroll et al., *supra* note 22 at 674–76 (using government visa lottery programs as an example where the use of algorithms intersects with the application of specific rules for decision-making that affect individual rights); *see also* Michael Veale, *Logics and Practices of Transparency and Opacity in Real-World Applications of Public Sector Machine Learning*, 4 WORKSHOP ON FAIRNESS ACCOUNTABILITY & TRANSPARENCY MACHINE LEARNING, at 2 (2017), https://arxiv.org/pdf/1706.09249.pdf [https://perma.cc/8X2L-RXZE]. (explaining how developers of public-sector machine learning tools use transparency and accountability measures to build consensus and approval for those tools among both colleagues and people at large).

and software-based processes in a way that hinders the ability to know what is happening within the system.[78] To be clear, given the nature of the administrative state, both scholars are correct that transparency is a normal, required part of due process. That is why Citron's point-by-point examination of the Administrative Procedure Act ("APA"), the Freedom of Information Act ("FOIA"), and similar state laws is powerful.[79] The APA requires that new rules undergo notice and comment,[80] and FOIA requires that the public have access to "basic information about the conduct of agencies."[81] But opaque code and the process behind developing code challenge the way in which "procedural due process and formal and informal rulemaking provided a common structure for debating and addressing concerns about the propriety of administrative actions."[82] Thus, these problems force the question of what is, as Citron puts it, "Technological Due Process?" Citron offers that "[a]utomated systems must be designed with transparency and accountability as their primary objectives, so as to prevent inadvertent and procedurally defective rulemaking."[83] Of late, legislators have started to look to public participation before software is built for public systems and to open code to facilitate "algorithmic audits" so people can test how an algorithm operates.[84]

As discussed *infra*, we hope to add to the idea of accountability, and to what Jenna Burrell has explained are the problems of opacity that arise for algorithms that operate at a certain scale of application and the limits of interpretability that go with that scale.[85] Specifically, we seek to challenge whether disclosure of software source code and data alone facilitates debating the function of that code. In that sense, Mashaw, Schwartz, and Citron, in different, connected ways, raise deep questions about whether and how the use of software by the state is compatible with due process.

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

78. Schwartz, *supra* note 33, at 1375 (calling for "[t]he maintenance of transparent information processing systems"); Citron, *supra* note 6, at 1295 (noting lack of ability for "meaningful review" of rules and system put in place to deliver administrative state services).

79. Citron, *supra* note 6, at 1288.

80. *See id.* at 1289–91.

81. *Id.* at 1291.

82. *Id.* at 1252.

83. *Id.* at 1308.

84. *See, e.g.*, Jim Dwyer, *Showing the Algorithms Behind New York City Services*, N.Y. TIMES (Aug. 24, 2017), https://www.nytimes.com/2017/08/24/nyregion/showing-the-algorithms-behind-new-york-city-services.html (last visited Dec. 19, 2017) (discussing a New York city councilman's bill to mandate that computer code used for government decision making be open for inspection and testing).

85. Jenna Burrell, *How the Machine 'Thinks': Understanding Opacity in Machine Learning Algorithms*, BIG DATA & SOC'Y, Jan.–June 2016, at 5, 9, http://journals.sagepub.com/doi/pdf/10.1177/2053951715622512 [https://perma.cc/UQG8-NL52].

machines track votes, and so the process in which the machines are used must be accurate about at least four things. First, the process 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. Second, voting machines themselves need to verify that the person voting was eligible to vote before recording the voter's intent. Third, voting machines need to be accurate about recording how an eligible voter voted. 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.[86] In one example, computer scientists showed that they could make a voting machine play a video game, Pac-Man, instead of tallying votes.[87] The point was not that officials would think Pac-Man voted, but that the machine can be tampered with, contravening the intuition and presumption at law that the machine is specialized for a purpose and cannot be made to do other things. Given this flexibility in the machine's behavior, having 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.

Given the pervasive use of software in industry, almost any industry in which devices are regulated and/or must behave in certain ways can raise issues about software and verification. The auto industry provides a good example. Software has governed the operation of cars for some time, but as software grows in importance for cars, so does the importance of technical 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.[88]

Two recent events in the auto industry — one involving Volkswagen, the other Tesla — demonstrate the challenge. First, the recent fraud by Volkswagen illustrates how software can aid a company in evading regulations. Volkswagen used software that allowed the

86. It is important to distinguish the very real problem of whether the machines and processes in use *could* 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.

87. *See* Kim Zetter, *Touchscreen E-Voting Machine Reprogrammed to Play Pac-Man*, WIRED (Aug. 24, 2010, 2:25 PM), https://www.wired.com/2010/08/pac-man/ [https://perma.cc/ZWD9-V2PA] (Describing how 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.").

88. As a microcosm of regulating automotive software, the final rule for software-controlled brakes and electronic stability control (ESC) systems is fascinating. *See* 49 C.F.R. §§ 571, 585 (2007) (establishing Federal Motor Vehicle Safety Standard No. 126 requiring the installation of a compliant ESC system in all passenger cars).

company to make its diesel cars seem to have low emissions when in fact the cars did not.[89] The ability to have technically accountable and analyzable algorithms in this sector would aid in detecting such fraud.[90]

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 a car's logs show that the car performed as promised, but regulators will need ways to verify that the software and logs have not been altered.[91] 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.[92] Whether those updates are accurate, comport with the company's public claims of performance, and whether they adhere to safety regulations need to be tracked.[93] Furthermore, as self-driving or autonomous cars continue to be put on the road and evolve, regulating their software will be even more important.[94] For example, if there is a standard for the way a self-driving car brakes or avoids another car or avoids a person, what happens when the automaker pushes an update to the fleet? How can regulators be sure that the updated software complies with the standard? The automaker's change affects not only the user but also others on the road. The automaker may, in good faith, assert that the update is within the standards already approved, but society, and more specifically the regulating agency, 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.

C. Algorithms, Private Sector Concerns

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

89. *See, e.g.*, Hotten, *supra* note 10. For a more technical overview, see Contag et al., *supra* note 10; Indictment, *supra* note 10.

90. *See infra* Section V.A.3 (explaining how to build accountable and analyzable algorithms).

91. *See infra* Section V.A.3 (explaining audit logs and verification of such logs).

92. *See* Davies, *supra* note 12.

93. Issues with cars' software updates and security have been revealed in at least two cases. *See* Morris, *supra* note 9 (reporting that a security group took over a Tesla, opened its doors, opened its sunroof, and enabled keyless driving so the car could be driven away or stolen); Peterson, *supra* note 9 (describing researchers able to take over the braking system and more from 12 miles away).

94. *See, e.g.*, Bryant Walker Smith, *Automated Driving and Product Liability*, 2017 MICH. ST. L. REV. 1, 45–49 (examining product liability and compliance issues raised by software in self-driving cars).

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.[95] 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.

One 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. A 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."[96] The study rejected the hypothesis "that no difference exists" in the delivery of such ads because searches for "black-identifying first names" 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 [searches] for white-identifying first names."[97] According to Sweeney, this finding intersects with discrimination problems, because when one competes for "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.[98] Another study by Datta et al. on searches, 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."[99] 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.[100] These studies have identified some outcomes that may not meet the legal[101] or normative definition of discrimination, but raise questions

95. 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 33, 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 1376. 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.

96. Sweeney, *supra* note 13, at 52.

97. *Id.* at 51.

98. *Id.* at 44.

99. Amit Datta, Michael Carl Tschantz & Anupam Datta, *Automated Experiments on Ad Privacy Settings*, PROC. ON PRIVACY ENHANCING TECHS., Apr. 2015, at 92, 92.

100. *Id.* at 102.

101. 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 1–4 (2014). These statutes apply to online practices but how they apply for each sector and which practices within each sector are prohibited is not settled. *See id.* at 8–10. Sweeney's

about "the pervasive structural nature of . . . discrimination in society at large."[102]

The studies cannot, however, find one party to blame, in part because of the many factors at play.[103] As Sweeney states, "Why is this discrimination occurring? Is [the ad buyer, the ad seller,] or society to blame? We do not yet know."[104] The Datta study also admitted that it could not determine whether the cause of the outcomes were from the advertising network (Google), "the advertiser, or complex interactions among them and others."[105] As such, Sweeney turns to technical solutions to address the issues and argues that "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."[106] The Datta study offers a tool to allow the ad network and the advertiser "to audit [each] other" to detect undesired ad behaviors.[107] The study suggests that in the future there may be "machine learning algorithms that automatically avoid discriminating against users in unacceptable ways and automatically provide transparency to users."[108] Of course, it remains to be seen whether these techniques will be convincing to users, will require their own audits, or will be capable of the

---

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 Sweeney'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, "what would meet the statutory requirement that the advertisement 'indicates any preference, limitation, or discrimination' concerning a protected class" is unclear. *Id.* at 9 (quoting 42 U.S.C. § 3604(c) (1988)). Thus, for example, as online advertising improves its ability to target advertising, whether the Act will apply for ad purchases that "will reach members of a protected class far more or less often than other demographic groups," it becomes important to know "whether and when the Act covers" such practices. *Id.*

102. Datta et al., *supra* note 99, 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* Consent Decree, United States v. Spokeo, Inc., No. 2:12-cv-05001 (C.D. Cal. June 12, 2012); Consent Order, United States v. Instant Checkmate, Inc., No. 3:14-cv-00675-H-JMA (S.D. Cal. Apr. 1, 2014); *see also Spokeo to Pay $800,000 to Settle FTC Charges Company Allegedly Marketed Information to Employers and Recruiters in Violation of FCRA*, FED. TRADE COMM'N (June 12, 2012), http://www.ftc.gov/news-events/press-releases/2012/06/spokeo-pay-800000-settle-ftc-charges-company-allegedly-marketed [https://perma.cc/UXP7-TVYL]; *Two Data Brokers Settle FTC Charges That They Sold Consumer Data without Complying with Protections Required under the Fair Credit Reporting Act*, FED. TRADE COMM'N (Apr. 9, 2014), https://www.ftc.gov/news-events/press-releases/2014/04/two-data-brokers-settle-ftc-charges-they-sold-consumer-data [https://perma.cc/ZU8F-4FTL].

103. *See, e.g.*, Datta et al., *supra* note 99, at 105 ("[B]laming one party may ignore context and correlations that make avoiding such discrimination difficult.").

104. Sweeney, *supra* note 13, at 52.

105. Datta et al., *supra* note 99, at 105.

106. Sweeney, *supra* note 13, at 53.

107. Datta et al., *supra* note 99, at 106.

108. *Id.*

promised auditing. 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 technical accountability. In that case, as a response to the tools proposed to be used to police the suspect sector, that sector could make the same critiques about opacity, fairness, and due process as those who currently question the use of algorithms make.

Academics are not the only ones to think about software and society. 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 computer scientists have looked at personalization of commerce and search features to see whether a single actor's implementation of an algorithm poses problems.

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.[109] Several companies "were consistently adjusting prices and displaying different product offers based on a range of characteristics that could be discovered about the user."[110] For example, Staples, Inc., the office supply company, charged different prices for the same item depending on where Staples thought the consumer was.[111] Although the practice of altering prices based on the geography of the shopper or whether a good is online or in-store is common, the practice can reinforce inequality if it allows retailers to charge lower prices to those who live in ZIP codes with higher weighted average income and charge higher prices to those in ZIP codes with lower weighted average income.[112] Even if one accepts the argument that a retailer accounts for different costs at local, physical stores, costs associated with physical retail stores should not be an issue if the orders are fulfilled and shipped from a central warehouse. Although the outcomes of price discrimination would seem to indicate that inequality could be reinforced, price discrimination is not illegal. 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

109. Jennifer Valentino-Devries, Jeremy Singer-Vine & Ashkan Soltani, *Websites Vary Prices, Deals Based on Users' Information*, WALL ST. J. (Dec. 24, 2012), http://www.wsj.com/news/articles/SB10001424127887323777204578189391813881534 (last visited Dec. 19, 2017).

110. *Id.* ("The Journal identified several companies, including Staples, Discover Financial Services, Rosetta Stone Inc. and Home Depot Inc., that [engaged in such activities].").

111. *Id.*

112. *Id.*

class status, price discrimination becomes prohibited discrimination.[113] As such regulation is triggered, it becomes necessary to understand the system.

Another investigation by Nicholas Diakopoulos tried to test the algorithms behind the autocomplete feature for Google and Bing's search services to see how each one handled searches for sensitive topics such as "illicit sex" and "violence."[114] 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 a large number of "110 sex-related words."[115] According to the author, this point raises the specter of censorship, because "we look to algorithms to enforce morality."[116]

This position is puzzling because whether society should look to algorithms or to a company's manual choice over a blacklist for morality enforcement is answered, "No," by most who discuss the issue, including us.[117] In addition, whether society truly "look[s] to algorithms to enforce morality" is unclear.[118] One may believe algorithms should be constructed to provide moral guidance or enforce a given morality. Alternatively, one may claim that moral choices are vested with a system's users and that the system itself should be neutral, allowing all types of use.[119] Either position demands certain outcomes from computer systems such as search engines, and so defers to algorithms. That is the mistake. In this context, the platforms at issue exercise discretion in organizing and displaying information, and that organization flows from tools — algorithms, blacklists, human moderators, and more —

---

113. *See* SWIRE, *supra* note 101, at 7–8 (discussing Fair Housing Act prohibition on "steering"). Steering is the practice of "deliberately guiding loan applicants or potential purchasers toward or away from certain types of loans or geographic areas because of race." FED. RESERVE BD., FEDERAL FAIR LENDING REGULATIONS AND STATUTES: FAIR HOUSING ACT 3 (2016), https://www.federalreserve.gov/boarddocs/supmanual/cch/fair_lend_fhact.pdf [https://perma.cc/UB2T-ZGS8].

114. Nicholas Diakopoulos, *Sex, Violence, and Autocomplete Algorithms*, SLATE (Aug. 2, 2013, 11:43 AM), http://www.slate.com/articles/technology/future_tense/2013/08/words_banned_from_bing_and_google_s_autocomplete_algorithms.html (last visited Dec. 19, 2017).

115. *Id.*

116. *Id.*

117. *See, e.g.*, Bogost, *supra* note 3; Deven R. Desai, *Exploration and Exploitation: An Essay on (Machine) Learning, Algorithms, and Information Provision*, 47 LOY. U. CHI. L.J. 541, 578 (2015).

118. Diakopolous, *supra* note 114.

119. The debates around hate speech and filter bubbles capture this problem, as one has to choose what to show or not show. *See* Desai, *supra* note 117, at 561–62 (discussing difficulties of determining who gets to decide what information users see). The same applies to the move to have platforms such as Facebook regulate fake news or hate speech as shown by Germany's recent law that requires firms to remove and block such content or face fines up to fifty million Euros. *See Germany Approves Plans to Fine Social Media Firms up to €50m*, THE GUARDIAN (June 30, 2017, 7:14 AM), https://www.theguardian.com/media/2017/jun/30/germany-approves-plans-to-fine-social-media-firms-up-to-50m [https://perma.cc/SK9D-V43W].

used in combination. In that sense, looking to algorithms or vesting agency with the range of tools platforms use to enforce morality is a type of "worship" that reverses the skepticism of the Enlightenment.[120] 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.[121] That is another path to censorship and control.[122] Nonetheless, in its best light, the argument seems to be that people might defer in a default way to such enforcement by algorithm; and so we must cope with that fact. And a problem is that those algorithms will not be perfect because "filtering algorithms will always have some error margin where they let through things we might still find objectionable."[123] We would add the system can also block content that someone or some groups do not find objectionable. Either way, the logic is that because people sometimes look to algorithms to enforce morality, we must police those algorithms; and that "with some vigilance, we can hold such algorithms accountable and better understand the underlying human (and corporate) criteria that drive such algorithms' moralizing."[124] This position merely substitutes one type of deference for another without acknowledgement. Even if such deference is inevitable — as the study seems to believe — the critique does not explain exactly what sort of accountability and understanding is possible.

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 varies depending on a range of factors.[125] As the authors explain, a given application of machine learning is better understood as consisting of 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" *working together as a system,* and so

120. *See* Bogost, *supra* note 3.

121. *Cf.* Desai, *supra* note 117, at 571–73 (explaining the difficulty for online information providers to show a given user a "good" song or to show a 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).

122. *See id.* at 561–62 (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).

123. Diakopoulos, *supra* note 114.

124. *Id.*

125. *See* Singh et al., *supra* note 23, at 3–4. Singh et al. also discuss 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.

each component offers different possibilities for control and responsibility.[126]

Focusing on only one part of such a system, the data, shows the ways that the idea of control becomes nuanced. Using "data in the wild" — that is deploying a system that may have been accurate and useful when built — requires ongoing monitoring and evaluation to ensure the model remains accurate given that the real world changes.[127] These changes can create what is called "concept drift" where the "once accurate model [becomes] obsolete."[128] 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."[129] 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 data by Internet trolls it became "foul-mouthed and racist"[130] — 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."[131] 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."[132] 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.[133] As the authors put it, "[w]ith 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."[134]

To summarize, there are broad descriptive claims of a range of differing problems appearing in the public and private sectors and flowing

126. *Id.* at 3.
127. *Id.* at 11–12.
128. *Id.* at 11.
129. *Id.*
130. *Id.*
131. Xinyu Xing et al., *Take This Personally: Pollution Attacks on Personalized Services*, 22 USENIX SEC. SYMP. 671, 671 (2013).
132. *Id.* at 672.
133. *Id.* at 671.
134. *Id.* at 684.

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 a viable solution and yet have different visions of what the term entails and how it would work in practice. In contrast, we argue that whether transparency is 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.

III. ALGORITHMS: A PRIMER

The word "algorithm" conjures thoughts of dark wizardry because algorithms are not well understood outside the technical community, not because they are a dark art.[135] Some algorithms are simple, and some are complex.[136] Regardless, an algorithm is a step-by-step process and "each of the steps must be absolutely precise, requiring no human intuition or guesswork."[137] Thus, we can call the steps for brushing teeth an algorithm. However, most of the time, including the issues addressed in this work and in most of the works 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 when a human follows a set of instructions and when a computer does.[138] Humans "might be able to tolerate it when an algorithm is imprecisely described, but a computer cannot."[139]

The idea of an algorithm as a recipe shows the problem. Recipes seem to be quite precise, but they are not.[140] As Brian Kernighan explains, "*Joy of Cooking* says that to poach an egg, 'Put in a small bowl:

---

135. *See* Bogost, *supra* note 3 ("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 et al., *supra* note 22, at 640 n.14 ("The term 'algorithm' is assigned disparate technical meanings in the literatures of computer science and other fields.").

136. 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 as Google's search are a "confluence of physical, virtual, computational, and non-computational stuffs — electricity, data centers, servers, air conditioners, security guards, financial markets." Bogost, *supra* note 3.

137. JOHN MACCORMICK, NINE ALGORITHMS THAT CHANGED THE FUTURE 3 (Vickie Kearn ed., 2012); *accord* THOMAS H. CORMEN, ALGORITHMS UNLOCKED 1 (Jim DeWolf ed., 2013).

138. CORMEN, *supra* note 137, at 1.

139. *Id.* As Cormen puts it, "[w]e 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.* at 2.

140. *Cf.* PEDRO DOMINGOS, THE MASTER ALGORITHM 3 (2015) ("[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.").

1 egg' but fails to specify that the egg should be broken and the shell removed first."[141] Humans can handle this ambiguity because humans can fill in details or otherwise guess at what to do when presented with a partially specified process, while a computer is a machine that can only follow precise instructions contained in its software.[142] As Cormen puts it, "given an input to a problem, [a computer algorithm] should always produce a correct solution to the problem."[143] In other words, a recipe is not an algorithm because a computer would not break the egg and so not produce the correct result.

Correctness, however, is not so simple; an algorithm's correctness can only be established relative to a specification of its behavior.[144] This point raises two issues. One can fail to choose as one's specification the correct solution to one's problem or one might fail to implement the solution faithfully. This distinction is analogous to building a house: a house may be built incorrectly because the builder failed to follow the blueprints (analogous to software failing to meet its specification) or the blueprints themselves may not describe the correct way to build the house (analogous to software faithfully implementing an incorrect specification). How GPS systems provide directions illustrates the problem for software.

If one thinks of a GPS navigation system giving a route, there may be several criteria for what the correct route should be.[145] 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. A driver may want the fastest route, the shortest route by distance (which may not be the fastest), or the fastest or shortest route that also avoids highways and

---

141. BRIAN W. KERNIGHAN, D IS FOR DIGITAL 53 (2012).

142. An irony is that the machine learning and neural network technologies driving 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 a cat and between one type of cat and another. Computer software was limited in such tasks in part because specifying such fine distinctions precisely for each instance (e.g. for each picture of a cat) in such a way that software could process well proved too complex a feat of engineering. Recent advances in machine learning use statistical methods and allow software systems to take less precise "objectives" that allow the technologies to discover the precise (if complex) rule by examining large data sets. Surprisingly, these less precise approaches have been the key to building systems that can accomplish the distinction task almost as well as a human can.

143. CORMEN, *supra* note 137, at 2. 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.* (stating that we want an algorithm to "use computational resources efficiently" while reaching the correct solution).

144. *See, e.g.*, Douglas D. Dunlop & Victor R. Basili, *A Comparative Analysis of Functional Correctness*, 14 ACM COMPUTING SURVS. 229, 229 (1982) (defining functional correctness as "a methodology for verifying that a program is correct with respect to an abstract specification function").

145. CORMEN, *supra* note 137, at 2.

toll roads.[146] Regardless, all correct routes will connect the origin to the destination.

Yet, after choosing from the above options, the user runs into a new problem. Suppose the user chooses the "fastest route" criterion as her definition of the correct route. The algorithm must have a way to determine whether one route is faster than another, which means without considering the current traffic conditions, the algorithm and its outputs will be incorrect.[147] Suppose that, instead of real-time traffic data, the user provides 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 actually the fastest route. The algorithm is nonetheless correct *given the input it had*.[148] 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."[149] Thus, the algorithm itself is correct regarding the specification, which in this case is to produce the fastest route given geography and traffic data. However, given incorrect input data, the correct 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, software can be incorrect either because the approach is incorrect (e.g., the programmer thought her approach would produce the fastest route, but it does not always do so) or because a programmer introduced a bug when converting it to computer code. Therefore, we must ask both whether a solution has been specified correctly and whether that specification has been correctly reduced to practice. In sum, *correctness is not as precise as policy critics would like*.

As such, to ask that an algorithm not discriminate or yield some other result prohibited by law, requires that a specification be provided so that the request is workable for computer scientists.[150] It also requires

146. *Id.*
147. *Id.*
148. *Id.*
149. *Id.*
150. The need for a specification further shows that algorithms are not recipes. *See supra* notes 140–142 and accompanying text. As Kroll et al. put it, "In technical approaches, it is

that the data used be accurate. And yet the policy process is rarely prepared to provide a complete specification for any rule, let alone thorny questions such as what constitutes discrimination (or even to answer the simpler question of whether a specific behavior is or is not acceptable or legal).[151] 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."[152] Thus, some argue that the "administrative process, like the judicial and legislative processes, [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."[153] 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. The same is true for software correctness, but the particulars of correctness in a given application will emerge from the messy governance processes around that application.

The idea that all computer systems are designed by deciding on a precise set of steps, which are completely specified, is wrong. Many systems spring more simply informal or notional specifications, where developers work from a poorly specified goal rather than a clear set of requirements. But this is not only a problem of insufficient precision — it can be quite desirable to build systems this way. Instead of a single formula that can set out or describe a specific result (e.g., $E=MC^2$), large modern computer systems often rely on high-level goals that generally describe how to solve a problem in cases where the solution is unknown in advance or benefits from exploration and testing in the real world. 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

---

traditional to have a detailed, well-defined specification of the behavior of a system for all types of situations." Kroll et al., *supra* note 22, at 695. The demand for specification allows an algorithm to work. In contrast, the law is more like a recipe where one may interpret instructions. *Id.* ("In lawmaking and the application of public policy, it is normal, and even encouraged, for rules to be left open to interpretation, with details filled by human judgment emerging from disputes in specific cases that are resolved after the fact.").

151. 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, when the state sets a clear rule, we can ask whether that rule is faithfully reflected in software. *See infra* Section VI.A. 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.

152. MASHAW, *supra* note 46, at 16.

153. *Id.* at 14.

unexpected, unplanned behaviors.[154] And so instead, developers experiment with likely approaches, measure their progress, and slowly optimize systems to find the algorithms which maximize revenue, user engagement, or whatever metrics are most relevant to a given problem.

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 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 discovered. 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 is derived either by the use of machine learning or simply by capturing sequences of movements of the actuators when controlled by a human directly. This model then defines the control of the arm, in place of a precisely specified algorithm. This lack of direct coding is especially necessary if the arm is to move and interact within a dynamic environment.[155] None of which is to say that the underlying physics do not matter and are not 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. The arm's movements are a combination of formulaic physics and 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.[156] The arm starts with a model that has unknown parameters.

---

154. As Singh et al. offer, "large-scale computing environments, for example IoT-enabled smart cities . . . entail 'systems of systems'. Such environments 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." Singh et al., *supra* note 23, at 15 (citation omitted). These overlapping factors create "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." *Id.* In that sense, large-scale computing environments face problems similar to what Mashaw described for the administrative state — rules that are more like ideals and complications flowing from imprecision and human factors in understanding and applying the rules. MASHAW, *supra* note 46, at 14–16.

155. *Cf.* Kroll et al., *supra* note 22, at 650.

156. There are several different approaches to machine learning. For a short overview of the approaches, see Singh et al., *supra* note 23, at 4–9. For a thorough treatment, see DOMINGOS, *supra* note 140 (explaining the current, major approaches to machine learning).

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 that it cannot yet handle. 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. Key to the learning process is the mathematical definition of how (quantitatively) good a particular set of parameters is. Machine learning is the process of finding the choice of parameters that optimize this objective. To achieve this, machine learning methods usually have a way of updating a set of parameters to improve performance with respect to the objective. Overall, models are developed by guessing parameters at random and testing the model's performance according to the objective. In a well-defined problem, the random choices made initially will not matter and the learning process will "converge" to the same answer when run repeatedly. However, it can also be beneficial to incorporate randomness into a system's rules to address the limits of a static system.[157]

Thus, not all algorithms are fixed and complete designs specified when a programmer chooses a precise set of steps. 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.[158] In effect, the ultimate algorithm that controls the arm's movement is *discovered from data* by the learning algorithm (using the objective to determine how effective any proffered solution is and using the learning algorithm to decide which possible solution to test next). But while the control algorithm is not developed by a human, the learning algorithm, the data, and any necessary guiding hints are. That is, the discovery process is controlled by methods, data, an objective, and often many implementation choices, each of which is controlled by a programmer. And the resulting discovered rule is still an algorithm, a precise rule which can be applied repeatedly to achieve the same result.

Even the environment of a program, such as the specific hardware configuration it is running on (e.g., the processor, the type of disk, the type of network interface) or how many other programs are running on that same hardware, matters for examining how that program operates.

---

157. *See* Kroll et al., *supra* note 22, at 655 (noting if one "hard-coded" the movement of a robot like the Roomba vacuum, "an unusual furniture configuration" might cause the device to become trapped in a corner but "randomized motion allows it to escape these patterns and work more effectively").

158. *See* Singh et al., *supra* note 23, 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 we have a program that measures how long it takes for some function to run. Let us say we wish to use this program to test how long an algorithm that sorts a large list of numbers takes to run. To test the performance of this sorting function, the program implementing the sorting algorithm starts a timer before running the algorithm to sort a large list of numbers and then checks how much time has elapsed when the sort is complete. Now, suppose this timing program is run on two computers, one on which it is the only program running and one on which there are one thousand other programs running, all of which are demanding many resources and reading/writing the disk heavily. We would expect the program to take much longer to complete its task in the second scenario, even though we would expect it to complete the task successfully and correctly in both. In short, the same 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 to build tools that will, in all cases, interpret from a description of a disclosed program or even its full source code whether it has specific behaviors such as granting or denying loans to a specific class of consumers or managing emissions in a required way. That is not to say that determining such things is impossible, merely that it cannot be guaranteed. 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 an evaluation program (whether a kind of supervisory software or a process applied by humans) that tests whether other programs discriminate or yield some other prohibited result is not workable. To support this argument, we turn in Part IV to a core part of mathematical theory having to do with decidability and computer science known as the halting problem.

IV. 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 all disclosed

algorithms for compliance with desired norms, such a tool will not be possible.[159] This complicates the classic view of policymakers that disclosure of a system's internals will, of necessity, 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 technologists can understand and that are workable.

A. Undecidability and the Halting Problem

There are certain questions for which there is no algorithm. That is, "there are problems for which it is *provably* impossible to create an algorithm that *always gives a correct answer*."[160] These problems are called "undecidable." It is possible to prove that such problems exist because computer science at its core is a kind of "mathematical reasoning."[161] In that sense, the deepest ideas in computer science are not about programming software or hardware design, but abstract concepts and issues that exist beyond software and hardware.[162] As such, understanding what can and cannot be computed gives insight as to why demanding software code to test for hidden agendas or prohibited outcomes will not work in at least some cases, perhaps important ones.

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.[163] 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."[164] When Turing discovered the proof in 1937, he was "interested in whether or not a given computer program would eventually produce an answer."[165] Specifically, he was interested in determining what was

159. 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. The core of the response was an almost mantra-like invocation: the idea that one cannot have algorithmic transparency of this sort because it will not provide transparency for any non-trivial semantic determination. As we will discuss, deep and important ideas from computer science limit what it is possible to guarantee that disclosures alone will reveal. This standard response was part of what stimulated one author's interest in this project.

160. CORMEN, *supra* note 137, at 210 (emphasis added).

161. MACCORMICK, *supra* note 137, at 176.

162. *Id.* at 4.

163. *Id.* at 195. Turing began this avenue of his work asking whether a "given computer program would eventually produce an answer." *Id.* A related questions asks whether a given computer program will "ever terminate — or, alternatively, will it go on computing forever, without producing an answer?" *Id.* Another way to ask this question is "whether a given computer program will eventually terminate, or 'halt,' [and] 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.'" *Id.*

164. *Id.*

165. *Id.*

computable, or possible to generate with a computer. Because Turing's work applies to what can be known about algorithms in general, it matters to those who wish to regulate them.

Turing offered an idealized machine that can stand in for any "digital computer" as a way to answer whether there are numbers that are "nameable, definable, and *not* computable."[166] Turing's machine, called *U*, as in Universal, represents a simple yet powerful model of computing machines.[167] In short, "[a] Turing machine can do everything that a real computer can do."[168]

By connecting the idea of "writing down by machine" and creating the 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."[169] By extension, whatever applies to Turing machines and the algorithms they can run also applies to the real-world software and hardware 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*.[170] That is, Turing machines define the limits of all real computers. Because there are abstract limits on what a Turing machine can do, these limits cannot be circumvented by building a bigger or more powerful computer or writing better software.[171] The limit we care about here is called the halting problem.

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

---

166. JAMES GLEICK, THE INFORMATION: A HISTORY, A THEORY, A FLOOD 207, 210 (2011).

167. Turing, *supra* note 1, at 241–43; *accord* MICHAEL SIPSER, INTRODUCTION TO THE THEORY OF COMPUTATION 202 (3d ed. 2013) ("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.").

168. SIPSER, *supra* note 167, at 165. 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. *Id.* In particular, Turing machines are a more similar theoretical model to modern electronic computers than even equivalent earlier logical models. *See id.*; GLEICK, *supra* note 166, at 211 ("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.").

169. SIPSER, *supra* note 167, at 184; *see also* GLEICK, *supra* note 166, at 208 (Turing "defined calculation as a mechanical procedure, an algorithm." (emphasis omitted)).

170. *See* SIPSER, *supra* note 167, at 165 ("[E]ven a Turing machine cannot solve certain problems. In a very real sense, these problems are beyond the theoretical limits of computation.").

171. *See id.*

program running on a certain input and determines whether the input program "halts" or "run[s] to completion"; Turing proved that no such method exists.[172] As Turing explained, there is no general tool or evaluator we can use to test a set of instructions (the "Standard Description"[173]) on a Machine and see whether a specific outcome will occur.[174] 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.[175]

We can now apply Turing's halting problem to the problems proposed to be cured by the idea of algorithmic transparency. To do so, we can consider the sort of misbehavior we are concerned with, such as unlawful discrimination, as a kind of crash, and ask whether it will occur.

B. The Halting Problem Applied to Algorithmic Transparency

By thinking of prohibited outcomes such as unlawful discrimination as crashes, we can define such outcomes as problems any programmer wishes to avoid.[176] That is, "[v]ery occasionally, even high-quality,

---

172. CORMEN, *supra* note 137, at 210. See SIPSER, *supra* note 167, at 216–17, for an outline of a proof of this result.

173. Turing, *supra* note 1, at 240 (defining "S.D [sic]" as "standard description").

174. *See id.* at 248. In Turing's more general language, "[w]e can show further that there can be no machine *E* which, when supplied with the S.D [sic] of an arbitrary machine *M*, will determine whether *M* ever prints a given symbol (0 say)." *Id.* (emphasis omitted). See SIPSER, *supra* note 167, at 201–09, for a detailed description of the theorem of undecidability and step-by-step ideas that lead to the proof.

175. CORMEN, *supra* note 137, at 210; *accord* SIPSER, *supra* note 167, at 216–17.

176. *See* MACCORMICK, *supra* note 137, at 175.

well-written software can do something it was not intended to do."[177] That unintended outcome can be thought of as a crash, whether it actually terminates the program or not. Therefore, if in fact we can detect when a program is discriminating, any program can be turned into one that crashes when it discriminates simply by running a routine that detects discrimination and then crashes if that routine finds discrimination. 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 that could find just these sorts of bugs. But that hope is a false dream. It is possible to write bug-free programs using advanced technical tools that prevent programmers from writing bugs in the first place.[178] And 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.[179] In all of these techniques, verification is with respect to a well-defined specification. For problems that are not amenable to clear, complete specification of a "correct" answer, such as the placement of advertisements, the ranking of web pages and social media posts, or the determination of whether a news story is a hoax, these techniques are difficult or impossible to apply.

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. However, *one precise thing that can be shown is that we cannot show certain things about software programs*. In the specific case of software, detecting a

177. *Id.*

178. *See generally* BENJAMIN C. PIERCE ET AL., PROGRAMMING LANGUAGE FOUNDATIONS (2017) (ebook) (providing an overview of one such technology); BENJAMIN C. PIERCE, TYPES AND PROGRAMMING LANGUAGES (2002). This is only one of the common approaches in software verification, the area of computer science concerned with the development of bug-free software. For an overview of the area and how it fits into the practice of software engineering, see CARLO GHEZZI, MEHDI JAZAYERI & DINO MANDRIOLI, FUNDAMENTALS OF SOFTWARE ENGINEERING 269–73 (2d ed. 2002). Another approach 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. This technique is known as "model checking." A detailed overview of this technique can be found in EDMUND M. CLARKE, JR., ORNA GRUMBERG & DORON A. PELED, MODEL CHECKING (1999).

179. *See* MACCORMICK, *supra* note 137, at 175 ("[W]ill the automated software-checking tools ever . . . [be able to] detect all potential problems in all computer programs? This would certainly be nice, since it would eliminate the possibility of software crashes . . . . The remarkable thing . . . is that this software nirvana will never be attained."); SIPSER, *supra* note 167, at 201 ("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.

potential crash is an undecidable problem,[180] meaning that "it is provably impossible for any software-checking tool to detect all possible crashes in all programs."[181] In other words, an analysis that will always correctly identify the misbehavior we wish to limit simply by reviewing an algorithm's source code and inputs does not exist. Yet when advocates of transparency demand disclosure, so that someone can "look under the hood" of an algorithm, they assume that such disclosures will of necessity always make plain the misbehavior of interest. As our analysis shows, there is no standard of review that is guaranteed to accomplish this for all disclosed programs.

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 — 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 does not know when to declare that A is stuck and will never halt, or whether it is making progress towards an answer, but slowly. B is not necessarily hopeless, but there is always some input pair (A, *x*) about which 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 is a very different proposition from the theorem saying that *no analysis can be done*. Rather, the point is just that any analysis will not work *in general*. That is, any sufficiently complicated analysis[182] 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."[183]

Computer science has met this reality 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

---

180. *See* MACCORMICK, *supra* note 137, at 195–96 ("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.").

181. *Id.* at 175–76.

182. Observe that "any sufficiently complicated analysis" might include a combination of analytic approaches, such that the switching to a different analysis if one's preferred standard cannot reach a conclusion does not sidestep this fundamental problem.

183. CORMEN, *supra* note 137, at 210. A related undecidable problem comes from Rice's theorem, which states that "determining *any property* of the languages recognized by Turing machines is undecidable." SIPSER, *supra* note 167, at 219. For Rice's paper, see H. G. Rice, *Classes of Recursively Enumerable Sets and Their Decision Problems*, 74 TRANSACTIONS AM. MATHEMATICAL SOC'Y 358 (1953).

particular, we can limit the failures of analysis methods in two ways. First, we can demand that the analysis be "complete," meaning that the analysis will detect all cases of a particular misbehavior. 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 undecidable analyses methods will, of necessity, miss some cases of misbehavior. No undecidable problem has an analysis that is both sound and complete; however, many sound and complete analysis methods may exist for problems which are computable.[184] 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.

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 such programs meet their specifications. That is, although we cannot guarantee that we can always analyze programs for correctness, we can guarantee that properly designed programs are correct. This supports arguments for capturing important values in software at the point of design and construction. 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.

V. PRACTICAL SOLUTIONS FROM COMPUTER SCIENCE

Returning to the idea of software misbehavior as a crash, that is, a result that we do not want, we can use computer science methods to mitigate such outcomes. On the one hand, one might ask for a guarantee

184. For example, it is undecidable in general to determine if a program has the security property of *memory safety*, which says that it will never make memory accesses beyond those which are intended, even when presented with adversarial inputs. However, programs written in restricted languages or modified in particular ways can be easily analyzed for memory safety. *See* Stephen McCamant & Greg Morrisett, *Evaluating SFI for a CISC Architecture*, 15 USENIX SEC. SYMP. 209 (2006) (describing a method for modifying a program to make it amenable to memory safety analysis). Google Chrome uses this technology to enable security analysis of programs efficiently enough to allow the execution of unvetted programs over the internet. *See* Bennet Yee et al., *Native client: A Sandbox for Portable, Untrusted x86 Native Code*, 30 IEEE SYMP. ON SEC. & PRIVACY 79 (2009).

that certain software was built to meet a specific standard and to have a way to verify that promise. As one of the authors has argued, one can keep that promise if one starts with the goal of building programs that can be analyzed for the particular property of interest.[185] 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. With a design that enables analysis, computer science can offer ways to give a complete guarantee that something is true about a piece of software under certain circumstances. The problem is that one cannot have such certainty for software that shows up from an unknown source, such as malware or software disclosed under a transparency regime. In addition, although one cannot discover 100% of bugs in existing software, one can write software that provably meets a specification and is therefore provably faithful to a stated goal. Further, one may be able to achieve high confidence outcomes for properties that are difficult or expensive to specify — such as the high availability or reliability guarantees software companies now offer in some areas.[186] Either of 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 satisfy critics who worry 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. Such critics may desire, and try, to audit software systems and the algorithms underlying them using social science methods after the systems are deployed to test whether these systems generate undesired outcomes.[187] 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. This section explains when and how certain testing methods can allow someone to test a given software and the limits — technical and practical — of those techniques.

---

185. *See* Kroll et al., *supra* note 22, at 652–53 (explaining that software design principles allow programs to satisfy guarantees convincingly, sidestepping the analytical difficulties created by the halting problem).

186. *Cf.* CORMEN, *supra* note 137, at 197 ("[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*.").

187. *See* Sandvig et al., *supra* note 20, at 5–6.

A. Testing and Evaluating Algorithms

The practical issues computer scientists face in evaluating software and algorithms show the limits of transparency[188] and that different iterations 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. That may seem to be the dream for some transparency advocates, but that approach still has 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.

Dynamic analysis is only as good as the number of input-output pairs that are available to be observed or tested.[189] 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.[190] And the inputs and outputs available for review or considered by such testing may or may not match the set of inputs and outputs that matter for policy analysis.

1. White-Box Testing

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

> Computer scientists evaluate programs using two testing methodologies: (1) static methods, which look at the code without running the program; and (2) dynamic methods, which run the program and assess the outputs for particular inputs or the state of the program as it is running. Dynamic methods can be divided into (a) observational methods in which an analyst can see how the program runs in the field with its natural inputs; and (b) testing methods, which are

188. This discussion is indebted to and draws on Kroll et al.'s paper. Kroll et al., *supra* note 22. In addition, we offer thanks to Ariel Feldman, for his generosity in exploring these issues.

189. *See id.* at 650.

190. *See id.*

> more powerful, where an analyst chooses inputs and submits them to the program.[191]

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 program analysis, both static and dynamic, that the halting problem implies: one cannot always predict whether a certain outcome will occur or whether a program has a certain type of bug.[192] Since nearly any interesting static analysis[193] is not computable, most static analysis used in practice is either unsound or incomplete, as described above.

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. Software is discrete — it can exhibit wildly different outputs or behaviors for different inputs, and can even carve out exceptions that apply only for specific inputs. So, testing what happens for an input x tells you essentially nothing about what happens for input y — it might be completely, radically different. For this reason, it is generally necessary and considered to be good practice when developing software to limit the scope of what inputs a program will allow, if only to make that program more easily testable (but also to make it more robust and secure).[194]

However, by combining simple properties determined by static analysis, it may be possible to establish ranges or classes of inputs for which output behavior can be generalized. In this way, transparency in the form of source code disclosure can prove useful to an analyst. However, transparency is often impossible or undesirable in practice. Without such additional knowledge, the analyst will have to test all possible inputs, which is rarely feasible for programs of interest.[195] In addition, even analysis using sophisticated automated systems, such as those we wish to interrogate, will run into the limits of software bug testing.[196] These problems increase with black-box testing.

---

191. *Id.* at 646–47 (citation omitted).

192. *See id.* at 650.

193. Roughly speaking, Rice's theorem defines non-trivial properties of programs as properties that are true of some programs but not others. For example, the property "this program outputs 0 regardless of its input" is a non-trivial property. It is easy to see that properties of interest to public policy will always be non-trivial.

194. A common aphorism in system design is to "be conservative in what you do, be liberal in what you accept from others." INFO. SCI. INST., TRANSMISSION CONTROL PROTOCOL 13 (Jon Postel ed., Jan. 1981). Sometimes called the Robustness Principle, this saying is often attributed to Jon Postel as "Postel's Law."

195. Still, this approach is sometimes done in the form of "model checking." *See* Kroll et al., *supra* note 22, at 664 n.102.

196. *See id.* at 650.

2. Black-Box Testing

As Kroll et al. explain, black-box testing poses large challenges as a basis for analyzing software systems. To start, there are a "finite number of inputs that can be tested or outputs that can be observed. This is important because decision policies tend to have many more possible inputs than a dynamic analysis can observe or test."[197] In other words, "Dynamic methods, including structured auditing of possible inputs, can explore only a small subset of those potential inputs. Therefore, no amount of dynamic testing can make an observer certain that he or she knows what the computer would do in some *other* situation that has yet to be tested.[198] Even if one combined static and dynamic testing methods, not every algorithm will be able to be fully analyzed.[199] As Kroll et al. 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 question.[200] To combat the problems and limits of transparency and black-box 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, one goal of those in favor of algorithmic transparency or regulatory agencies who need to govern industries that use algorithms is to review a software-driven action after the fact of the action. That goal runs into the problems of transparency and software testing, as well as the problem of determining whether the action originated from a specific piece of software. There are many options to meet those challenges. One option is full transparency. A computer is, after all, a precise machine, and with a full understanding of its construction and operating state, we can reproduce its actions. 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

197. *Id.*

198. *Id.* (footnote omitted).

199. Indeed, Rice's Theorem holds that most interesting program analysis is undecidable. Informally, the theorem states that computing any non-trivial property of a program is undecidable. *See* SIPSER, *supra* note 167, at 219. For a statement of the theorem, see *id.* at 241 (problem 5.28). For an outline of the proof, see *id.* at 243 (solution 5.28). For Rice's paper, see Rice, *supra* note 183.

200. Kroll et al., *supra* note 22, at 659.

system again). But such complete transparency is rarely possible or desirable.[201] Often such systems may be subject to trade secret or other concerns that run counter to full transparency. And here we run into the limits discussed in Part III, *supra*, for full transparency supporting reproducibility will in many cases require disclosing significant detail about a program's operating environment, such as the full contents of databases with which a program interacts and information about other programs running on the same computer.[202] Again, such detail is rarely desirable, or even feasible, to disclose. But we may not need such detail to achieve the outcomes full transparency advocates seek.

Insofar as testing an outcome after the fact is about technical accountability and 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[203] and zero-knowledge proofs[204] 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 these undisclosed elements are duly recorded, or are consistent among decision subjects as appropriate (e.g., all decisions are rendered from the same decision policy).[205] These techniques can allow for ex-post verification even when the entire system is not transparent, functioning as a kind of agent-less software escrow.[206] Skeptical decision subjects, oversight entities, and concerned citizens can later verify that the software which was meant to be used and committed to — or, in the escrow analogy, deposited into escrow — was actually used for a decision.[207] Further, they 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

---

201. Kroll et al. also note that transparency alone is often also *insufficient* to support governance inquiries, as systems can often exhibit behaviors or investigators may take interest in properties that are not meaningfully explained by any disclosures. *See id.* at 658–59.

202. *See supra* Part III.

203. *See* Kroll et al., *supra* note 22, at 665. As Kroll et al. explain, a cryptographic commitment "is the digital equivalent of a sealed document held by a third party or in a safe place." *Id.* Such commitments act like a "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." *Id.*

204. *Id.* at 668. A zero-knowledge proof works as part of a cryptographic commitment. *Id.* It "allows a decisionmaker . . . 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." *Id.* (emphasis omitted).

205. *Id.* at 662.

206. This escrow is a simulated one, rather than a full deposit of a given system's code such as Pasquale has called for. Pasquale, *supra* note 95, at 166. And we believe aids in achieving some of Pasquale's goals without the problems of a full escrow.

207. This approach thus helps address concerns Levine has raised about the tensions between a third-party verification system and trade secret protection. David Levine, *Trade Secrets in Our Public Infrastructure*, 59 FLA. L. REV. 135, 187 (2007).

decisions about them correspond to the version of the system reviewed by an analyst 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. Thus, oversight ex post can compel the release of protected materials under a suitable protective regime, analogous to a court ordering an escrow agent to turn over the escrowed information.

For example, if one wants to review an action after the fact, one needs an audit log or audit trail, a time-stamped record that documents actions that affect an operation, procedure, or event.[208] If one wants to know that the audit log corresponds to what actually happened, one can either re-run the entire system and compare the inputs and outputs or one can use these cryptographic methods. Even a passive observer who is a member of the public (and not just privileged regulators with 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 analysis based on the creation of audit logs. First, if a system changes over time, it is necessary to determine which version was in effect for any decision of interest. If such systems are created with the above requirements for the creation of reviewable evidence in mind, whatever analysis is possible may aid in determining whether, when, and how things changed or could be used 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, especially highly dynamic systems, that are not designed for evaluation pose perhaps a larger problem, as it is not clear the extent to

208. The reliance on such trails is known and used in the law. For example, the Food and Drug Administration requires the use of such trails in electronic record keeping. *See* 21 C.F.R. § 11.10 (1997). In addition, Securities and Exchange Commission Rule 613 requires a "comprehensive consolidated audit trail" that "allows regulators to efficiently and accurately track all activity in . . . securities through the U.S. markets." SEC Rule 613, 17 C.F.R. Parts 242 (October 1, 2012), https://www.sec.gov/rules/final/2012/34-67457.pdf [https://perma.cc/8U78-ZB44].

which even weaker methods such as auditing provide meaningful information about their operation.

The systems that cause much concern — those that govern "search engine rankings, spam filters, intrusion detection systems, . . . website ads, . . . [and which] social media posts to display to users"[209] — that is, software that uses certain types of machine learning or is modified frequently by its operators to respond and adapt to dynamic inputs and user behavior, are not addressed well by the solutions presented in §§ V.A.1–3, *supra*. Many systems change often, either because of regular changes by designers or because they use automated processes such as online machine learning models which "can update their . . . predictions after each decision, incorporating each new observation as part of their training data."[210] 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. call for a design-based approach, which necessarily addresses issues looking forward. While Kroll et al. specifically consider ex-post review and oversight, it is primarily as a mechanism for updating policies for future decisions or redressing deficiencies in past decisions. Specifically, Kroll et al.'s call to use commitments requires that a single policy to be decided on and published 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 Section VI.3, *infra*.

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.

VI. 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

209. Kroll et al., *supra* note 22, at 659–60 (citations omitted).
210. *Id.* at 660.

and legal standpoint. The nature of the appropriate mechanism for technical accountability will depend on the nature of the software system at issue. Whether stringent measures guaranteeing technical accountability are needed may turn on whether the use of such systems is regulated. In some cases, robust technical accountability will be a requirement; in others, building technical accountability into software systems will be a best practice, at least under current policy.[211] We begin this Part by looking at public sector decision-making and explain why technical accountability is necessary as a matter of due process. We then turn to private sector, regulated industries which also require technical accountability. Next, we examine unregulated industries and offer best practices for technical 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, society generally needs and wants accountable public systems, and by extension society needs some ability to verify that certain systems operate in a certain way. Recall that a large barrier to technical accountability occurs when a system is built without an eye from the start to what evidence is necessary to review its actions and hold it to account. One proffered solution is that when the government chooses to use or purchase software, it must use software that meets the standards set by the government, and the standards should include full transparency.[212] But as we have argued, this approach simultaneously demands too much disclosure without assurance that these disclosures in fact cover what is needed.[213]

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 the evaluation of relevant guarantees[214] and thus open the door to political accountability.[215] The state may build such software in-house or buy it, but the requirement applies in both cases. The government, especially at the state or local level, may not be able to build the software it needs. Yet

---

211. *Cf.* Citron, *supra* note 6, at 1309.

212. *See id.* at 1308 (arguing that source code should be released to the public because opening source code "would reveal how a system works").

213. *See, e.g.*, Dwyer, *supra* note 84 (noting a New York legislator's call for software transparency).

214. *See* Citron, *supra* note 6, at 1310.

215. *Id.* at 1297 ("Encoded rules that change established policy cannot be understood by affected individuals or reviewed by more democratically accountable superiors.").

when using outside vendors to provide the software or software-based analysis, the government can and should, set requirements that promote technical accountability. Specifically, the contract can require that the creation of relevant evidence be a property of the system. That is, the evidence should explain both the goals of the system and the fact that it meets those goals, and should be evident to a curious and competent observer.[216]

Given that the government already vets software for security and privacy issues among other criteria, demanding the creation of certain types of evidence is not peculiar. Private sector companies build software with similar requirements for both internal use and for sale to customers. In addition, enterprise software providers often create custom software for clients. The government is a large, albeit somewhat special, customer. It is nonetheless a customer that can list its needs for the market to meet.[217] 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."[218] The FDA also specifies that audit trails be a part of "such procedures and controls."[219] Our argument is that technical accountability via 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 decisions made. 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, as described by Schwartz: making accurate and cost-efficient judgments, while giving "attention to the dignity of participants."[220]

A modification of the demands of dignity is, however, needed. Although the subject of such a process may not have the literal ability to

---

216. Citron has argued that agencies using software "should be required to test a system's software." *Id.* at 1310. We seek to add to that insight ways to meet that requirement, noting that evidence may be produced directly via software or it may come from process controls outside automated processing.

217. *See id.* ("Federal procurement regulations could require contracts to specify that decision systems pass testing suites before states can accept systems from vendors.").

218. 21 C.F.R. § 11.10.

219. *Id.* ("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."). The code also makes sure that the records are kept so that agency review is possible. *See id.* ("[A]udit 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.").

220. Schwartz, *supra* note 33, at 1348.

know or understand what reasons are behind a decision,[221] when a sensitive decision is being made — or as Mashaw, Schwartz, and Citron have stated in different ways, when the state is making decisions that raise due process concerns — the state must use software that furnishes relevant evidence to support evaluation and hence allow for technical accountability. 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.

Examining an application of this approach further illustrates its benefits. At the general level, software firms will have a requirement that they can build — furnish the evidence necessary to enable oversight entities (and ultimately end users, decision subjects, and passive observers) to review the correct operation of the system as a whole. Insofar as the government is the one claiming that a process uses certain methods, meets certain legal requirements, etc., the government can try to offer specifications that a vendor can implement. Government programs that Schwartz and Citron critiqued or that Kroll et al. offer as good cases for requiring technical accountability have set parameters about what factors they can and cannot consider and about the methods they can use to make a decision. When an agency is deciding who should receive certain welfare benefits, whether someone should be on a no-fly list, who should be identified as owing child support,[222] 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* must be able to show that in fact it used those criteria, whether or not software was part of the decision-making.

Of course, many decisions in this area involve the discretion of someone in an agency, but that does not change the need for having robust evidence throughout the decision process.[223] Insofar as the parameters allow for little discretion, when they are set out and applied via software with audit-logs and the sort of technical accountability Kroll et al. describe, society 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 under certain criteria and upon which a decision was made, the decision-maker can show that the

221. *See id.* at 1349.

222. Citron has identified these as examples of automated decision-making systems that "offend basic norms of due process by failing to provide both notice of the basis of their decisions and the means to review them." Citron, *supra* note 6, at 1256–57.

223. *See supra* notes 152–153, and accompanying text (noting difference between exact rules and demands of justice for administrative law).

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.

B. Private Systems

Although due process concerns explain why we would require the creation of robust trails of evidence for software-driven decision processes in government, 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 evidence of the correctness of their decision processes as matter of trust, but whether a given actor or area of industry will be required by law to do so varies depending on the actor or industry.

Whether the creation of such evidence is required for private sector use of software turns on a few issues. If the private sector industry in question is regulated, such as the auto or pharmaceutical industry, evidence of correctness can naturally become a requirement. If the sector in question is not regulated, it would seem that building software to produce the evidence necessary to facilitate technical accountability 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 demonstrating the correctness of their automated decision-making as a best practice, since it will enable trust in their products and services. In addition, we offer that such an approach can allow community feedback and encourage the identification of unexpected errors in ways that can help companies, much as bug testing challenges currently help discover problems in existing software. Last, we offer that whistleblower protection may be useful to aid in revealing whether a company is intentionally and overtly using prohibited discriminatory methods such as criteria based on race or gender in its software.

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 correctness would have assisted compliance and enforcement. An automaker could be required to provide the software to a government tester. The software could be required to be designed to provide evidence that supports analysis of its compliance with relevant standards. Using Kroll et al.'s ideas about zero-knowledge proofs, the automaker could keep proprietary methods secret, yet still be required to commit to using certain methods and to keep audit logs so that testers could verify that the system performed as promised in the field.

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 a station.[224] And while that new functionality is important and useful, updates can radically change the behavior, and therefore compliance, of a computer system.

The key point is that Tesla and other automakers pursuing the next generation of auto making are treating the car more like a PC than any car made to date. Tesla alone is poised to issue updates every few months, has issued updates to improve 0-to-60 performance and that add "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.[225] 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.[226] The so-called auto-pilot mode was delivered as a software upgrade as well.[227] Future Teslas, Volkswagens, and likely almost all other automakers' cars will rely more and more on software for controlling most of the ways in which their cars operate, and on network connections to the world outside the vehicle for updates and real-time data. This point calls out a problem for any networked, software-driven device — what one could put under the idea of the Internet of Things[228] — that requires

224. *See* Davies, *supra* note 12.

225. *Id.*

226. *See id.*

227. *See* Neal E. Boudette, *Tesla Makes Safety Upgrade in Autopilot, to Cars Already on the Road*, N.Y. TIMES, Sept. 24, 2016, at B3, https://www.nytimes.com/2016/09/24/business/tesla-upgrades-autopilot-in-cars-on-the-road.html (last accessed Dec. 19, 2017).

228. The Internet of Things refers to the way in which low-power sensors and networked devices together mean that almost any physical thing can gather information and be on the Internet. *See 2013: The Year of the Internet of Things*, MIT TECH. REV. (Jan. 4, 2013), https://www.technologyreview.com/s/509546/2013-the-year-of-the-internet-of-things/ [https://perma.cc/NZ59-EZC4]. Perhaps now familiar examples are public transit cars and

some approval before being deployed, since software-driven features of such devices can change radically when the software is updated.

Insofar as any item makes a claim about performance or adds a feature in a regulated context, that feature must be tested before being put into the stream of commerce and should be built to provide the requisite evidence of how will operate and did operate once in the field. If a company wants to update systems that normally undergo review by a regulatory agency such as NHTSA or FDA, it may 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 (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 alters the agreed upon original testing parameters — companies can update rapidly as long as they can demonstrate that their update falls within those parameters. With guarantees from a company in place, an agency may be able to test and approve updates faster than it could 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 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 shifts in unregulated sectors. Like a car battery range indicator, the charge indicator on a cell phone is based on an algorithm,[229] 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, we want to know whether the indicator is accurate within certain parameters, and we 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, Kroll et al.'s approach of requiring public commitments to the software used coupled with proofs that the software referenced by

buses with sensors that let operators and riders know vehicles' statuses, the spread of personal health monitors such as Fitbits, and smart home technology. *See id.*

229. *See* Wen-Yeau Chang, *The State of Charge Estimating Methods for Battery: A Review*, 2013 ISRN APPLIED MATHEMATICS, at 1, https://www.hindawi.com/journals/isrn/2013/953792/ [https://perma.cc/YM2Y-ZLHG].

the commitment was in use in practice should work well, save for the fact that battery indicators likely change too often to justify the overhead of these techniques in practice. If a company wishes to update or enhance a feature that does not require government testing and approval before launch, and 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 the logs are as they were when the issue arose, and have not been doctored.

Thus, as software is developed and deployed, starting with the question of what evidence should be created to afford evaluation of the behavior of that software to facilitate technical accountability for its outputs aids a company in two ways. First, the approach allows a company to address allegations that the software did not function in the expected way, because there will be evidence with which to rebut such claims in public (or simply to deter spurious claims in the first place). 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 inquiries without having to stick their head in the technical sand.[230] They will be able to claim that trade secret, complexity, technical infeasibility, or some combination of these issues means the government and society must simply defer to and trust the company that they are being honest and obeying rules.

Although designing software that facilitates evaluation through the creation of evidence 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 evidentiary criteria for which we argue. As above, whether such systems should be allowed 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 systems produce the necessary evidence to enable the evaluation and analysis that supports technical accountability for public sector use of software mean that the

230. *Cf.* Deven R. Desai, *Beyond Location: Data Security in the 21st Century*, 56 COMM. ACM 34, 36 (2013).

government may not be allowed to use such dynamic systems for certain purposes.[231] Regardless of whether the government builds or outsources the creation of the software in question, unless evaluations supported by technical accountability are possible and the software adequately addresses compliance requirements, due process, participation, and justice concerns indicate that such systems cannot be used.

Some may argue 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. That view is, of course, an overstatement. It also misses the point of Mashaw's triumvirate for government decision-making. Decisions must strive to be accurate, to be 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."[232] 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 by perhaps detecting radicalization. New online machine learning techniques may be better than humans alone at detecting patterns of radicalization that indicate someone may be considering violent or terrorist acts. 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 billions of people who are online and who use social media, the vast number of posts, and the nuances of each language used in a post, having humans try to cull through and understand all of the posts 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 the one at work the next day or week. In addition, the raw

231. Even if one meets the criteria for technical accountability as we have described it, there are reasons that the government should not be allowed to use certain techniques when due process and justice are at stake. For example, on November 16, 2017, a group of fifty-four leading AI researchers (including the computer scientist author of this paper) released an open letter to the Acting Secretary of Homeland Security arguing that a proposed procurement for a machine learning system that seeks to make "determinations through automation" as to whether an individual seeking a visa for entry to the United States will be a "positively contributing member of society", will "contribute to the national interests", or "intends to commit" criminal or terrorist acts, is not viable because "no computational methods can provide reliable or objective assessments of the traits that ICE seeks to measure. [And in] all likelihood, the proposed system would be inaccurate and biased." Hal Abelson et al., *Technology Experts Letter to DHS Opposing the Extreme Vetting Initiative*, BRENNAN CENTER (Nov. 16, 2017), https://www.brennancenter.org/sites/default/files/Technology%20Experts%20Letter%20to%20DHS%20Opposing%20the%20Extreme%20Vetting%20Initiative%20-%2011.15.17.pdf [https://perma.cc/VTC7-EWFG].

232. Schwartz, *supra* note 33, at 1349.

dollar cost of such manual review would be quite high. If we further desire to cross-reference the paper trail with pubic online social media posts, facial recognition, and more, we see that the task is not a good one for humans, but is 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, and indeed no "ground truth" exists for the criterion of the list (an unmeasurable risk), testing whether the list is accurate becomes an open question. Furthermore, if someone is on the list, believes that she should not be, and wants to challenge the choice, the lack of interpretability from being denied the purchase of an airline ticket or the ability to board a flight undermines the dignity interest that completes Mashaw's triumvirate. Thus, 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 technical accountability, evaluation, and redress to subjects of incorrect decisions.

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. Similarly, systems could be designed to support the legitimate activities of law enforcement officers, making determinations more efficient and cost-effective within the context of existing, constrained bureaucratic processes.

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 which are 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 the claims, as they are varied and have differing amounts of accuracy and substance to their objections. For the purposes of this Article, 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 online machine learning techniques because those techniques evolve the rule in effect for any given decision. That is, whether a company using dynamic systems will be able to work within the technical accountability and evaluation criteria for which we

have called 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 desirable. 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.[233] 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 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. 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, many users may believe their posts ought to be seen by all their friends and followers, and so changes which rank posts based on how interesting they will be to readers can alter that expectation. 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 limit the number of demoted sites or posts, or the severity of any demotion, 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, society is demanding some sort of ability to assess them. In addition, many of the systems at issue use machine learning. The term "machine learning" suffers from a problem similar to the term algorithm: there is an almost mystical or monolithic view of what the term means.[234] Yet machine learning as a field employs many specific, non-mysterious algorithmic tools such as decision trees, rule learners, classification techniques, such as naïve Bayes or nearest-neighbor classification, neural networks, and support vector machines[235] because "[i]n practice, . . . each of these algorithms is good

---

233. *See, e.g.*, *Google Algorithm Change History*, MOZ, https://moz.com/google-algorithm-change [https://perma.cc/BJ3Y-P9U9] (displaying a database of major Google algorithm changes).

234. *See* Bogost, *supra* note 3.

235. *See* Singh et al., *supra* note 23, 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 140.

for some things but not others."[236] As a general matter, one group of computer scientists has noted within machine learning "[s]ome . . . algorithms are more amenable to meaningful inspection and management than others."[237] Decision trees, naïve Bayes classification, and rule learners were the most interpretable, kNN was in the middle, and neural networks and support vector machines were the least interpretable.[238] But interpretability is not the only challenge or goal when using these approaches.

A study looking at which method was best to use for medical predictions explains that different approaches offer trade-offs because some "are very accurate, but unfortunately also relatively unintelligible" and others are intelligible but less accurate.[239] Recent new techniques "are very accurate, but unfortunately also relatively unintelligible such as boosted trees, random forests, bagged trees, kernelized-SVMs, neural nets, deep neural nets, and ensembles of these methods."[240] That study showed that one method — high-performance generalized additive models with pairwise interactions ("GA2M") — allowed for high accuracy and intelligibility.[241] The study was looking, however, at only 46 features (in simpler terms, factors used to make the prediction),[242] and so the suggested methods may not work in other areas of information provision such as search, online ad delivery, and online news where hundreds if not thousands of features are used.[243] In addition, even for the smaller feature set, the explanation does not tell why the decision rule was chosen or that it is the correct one; it only shows that the rule has high utility.[244] As such, we offer generally that machine learning systems need not be inscrutable, but the context of their application such as the number of features used and what one is trying to do governs how intelligible and accurate a given machine learning technique will be.[245]

Even if a system was not built from the start to facilitate technical accountability, there are ways to mitigate this fact. Any company using

---

236. DOMINGOS, *supra* note 140, at xvii.

237. Singh et al., *supra* note 23, at 4.

238. *See id.*

239. Rich Caruana et al., *Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission*, 21 PROC. ACM SIGKDD INT'L CONF. ON KNOWLEDGE DISCOVERY & DATA MINING 1721, 1722 (2015).

240. *Id.*

241. *See id.* at 1730.

242. *Id.* at 1722.

243. *See* Burrell, *supra* note 85, at 9 ("With greater computational resources, and many terabytes of data to mine . . . the number of possible features to include in a classifier rapidly grows way beyond what can be easily grasped by a reasoning human.").

244. *See id.* (noting that giving up on answering why an algorithm gave a result in favor of evaluating the outcomes as fair or desired is an option when facing opacity problems).

245. Michael Veale has shown that public contractors and vendors are sensitive to the tradeoff between transparency and opacity. Veale, *supra* note 77.

a given machine learning approach 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 can facilitate technical accountability in practice because analysis of a system's behavior that must be created to support the requirements of the system's creators and operators will often also help oversight processes. Beyond this, much technical work looks to make machine learning interpretable,[246] or attempts to provide explanations of what an inscrutable model is doing and why.[247] Such approaches aim to find ways to make simple models perform as well as complex ones, and to provide additional information along with the predictions of a complex machine learning model that either describes or justifies its predictions. Another area of research aims to solve the problem of "attributing the prediction of a deep network to its input features,"[248] meaning to assess the extent to which portions of the input affected the output. Researchers argue that if one can "understand the input-output behavior of the deep network, [that] gives us the ability to improve it. *Such understandability is critical to all computer programs, including machine learning models*."[249]

However, the usefulness of such technologies can be limited. After all, explanations can easily justify wrong answers or lend credence to uncertainty. Rather, understanding the nature of the problem that a software system is trying to solve (and thereby what evidence is necessary to convince different stakeholders that the system is actually solving that problem) is a more promising approach. Further, it is this understanding that promotes consistency with social, political, and legal norms. Interpretation or explanation of the behavior of a machine learning system can help the developers of such systems interrogate whether the systems are correctly reflecting the world in certain cases, but not without an understanding of what that reflection should look like. Regardless, convincing those outside the creation of the system that it is operating correctly is necessary, and explanations do not provide the evidence required for this. And indeed, the utility of interpretability for

246. For a summary of the area, see Finale Doshi-Velez & Been Kim, *Towards a Rigorous Science of Interpretable Machine Learning*, ARXIV (Mar. 2, 2017), https://arxiv.org/pdf/1702.08608.pdf [https://perma.cc/M576-9FJE] (laying out the field of machine learning interpretability and a framework for future research).

247. *See, e.g.*, Marco Tulio Ribeiro, Sameer Singh & Carlos Guestrin, *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*, 22 PROC. ACM SIGKDD INT'L CONF. ON KNOWLEDGE DISCOVERY & DATA MINING 1135, 1135 (2016) (describing Local Interpretable Model Explanations, a technique that purports to explain any model, no matter how complex, by considering its behavior for inputs similar to a test input).

248. Mukund Sundararajan, Ankur Taly & Qiqi Yan, *Axiomatic Attribution for Deep Networks*, 34 PROC. INT'L CONF. ON MACHINE LEARNING, at 1 (2017), http://proceedings.mlr.press/v70/sundararajan17a/sundararajan17a.pdf [https://perma.cc/D6SD-P7KA]. For an approach to examining the influence of inputs on outputs, see Anupam Datta, Shayak Sen & Yair Zick, *Algorithmic Transparency via Quantitative Input Influence: Theory and Experiments with Learning Systems*, 37 IEEE SYMP. ON SEC. & PRIVACY 598 (2016).

249. Sundararajan et al., *supra* note 248, at 1 (emphasis added).

facilitating understanding in legal and policy contexts has been questioned.[250]

In addition, companies are constantly updating and building their software systems, and that provides the chance to build the creation of relevant evidence of operation into the software as it evolves. Even when that is not possible, insofar as a company may need to explain what its system does, it can make use of these explanation approaches, supplemented by evidence appropriate to the context of the system in question and the problem that system is trying to solve, to provide limited insight into the system's operation.

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[251] rather than an antagonistic policing game of "Gotcha!," in which critics cry foul in public, a more constructive system of helping improve the software 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.[252] A system of testing and bounties could help ferret out such problems before they become problematic product issues and negative news stories.

Of course, even when a company offers evidence that a system is working well, skeptics may believe that the company designed its software to have discriminatory or illegal outcomes in the hope that problems would be missed in review of the evidence or that, on being discovered, it could deny that intent. For that possibility we propose that a whistleblower and public interest right of action law may be needed.

---

250. *See* Lilian Edwards & Michael Veale, *Slave to the Algorithm? Why a 'Right to an Explanation' Is Probably Not the Remedy You Are Looking For*, 16 DUKE L. & TECH. REV. 18 (2017) (arguing that explanations do not yield the sort of understanding required by legal and policy processes).

251. This is commonly done with software security bugs. These programs date to the mid-1990s. *See Netscape Announces 'Netscape Bugs Bounty' with Release of Netscape Navigator 2.0 Beta*, PR NEWSWIRE (Oct. 10, 1995), https://www.thefreelibrary.com/NETSCAPE+ANNOUNCES+'NETSCAPE+BUGS+BOUNTY'+WITH+RELEASE+OF+NETSCAPE...-a017551947 [https://perma.cc/3KFP-R8R4]; Casey Ellis, *Bug Bounty Model Celebrates 21st Birthday!*, BUGCROWD BLOG, (Oct. 20, 2016, 10:15:00 AM), https://blog.bugcrowd.com/bug-bounty-21st-birthday [https://perma.cc/QAW6-NFG4].

252. *See, e.g.*, Alastair Barr, *Google Mistakenly Tags Black People as 'Gorillas,' Showing Limits of Algorithms*, WALL ST. J. (July 1, 2015, 3:40 PM), http://blogs.wsj.com/digits/2015/07/01/google-mistakenly-tags-black-people-as-gorillas-showing-limits-of-algorithms/ (last visited Dec. 19, 2017).

C. Legislative Changes to Improve Accountability

Even with robust technical controls, some may still believe that companies using software-driven decision-making will have designed the software to cause negative, discriminatory, or unfair outcomes or to evade regulatory law while hoping that no one would know or that the perpetrators could deny that intent.[253] A perhaps more likely problem would be software that was not designed to discriminate or, for example, misreport emissions, but testing and evaluation during development or after software is released shows such outcomes. A firm may try to ignore or hide such knowledge, but an employee may know of the problem and wish to report it. Thus, we offer that, in addition to technical measures, policy-based controls are appropriate. In particular, we propose recent changes in trade secrecy law to protect whistleblowers, protection from employers who retaliate against whistleblowers, and a new public interest cause of action law would improve legal-political accountability.[254] Taken together, enhanced 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 or other illegal acts via software.

Because software and data can be treated as trade secrets, trade secret law clashes with the ability to know whether software is operating as it should and by extension trade secret law can interfere with legal-political accountability,[255] but that does not have to be so. Trade secret law began as, and is part of, commercial morality.[256] Trade secret law

253. *See* Barocas & Selbst, *supra* note 19, at 674.

254. The ideas for this section came to one of the authors as he noticed a parallel to problems he had written about concerning the private military industry. *See* Deven R. Desai, *Have Your Cake and Eat It Too*, 39 UNIV. S.F. L. REV. 825, 861–64 (2005). While this Article was being written, Professor Peter Menell wrote about the problems with trade secret law, the public interest, and the lack of whistleblower protection for whistleblowers who need to use trade secrets to alert authorities to wrong-doing. Peter S. Menell, *Tailoring a Public Policy Exception to Trade Secret Protection*, 105 CALIF. L. REV. 1 (2017). Professor Menell's argument was adopted as part of the Defend Trade Secrets Act of 2016. *Id.* at 62. As such, we are grateful for Professor Menell's work and draw on it in the final version of this Article. In addition, we are indebted to Professor Sonia Katyal whose email to one of us and forthcoming paper on civil rights and algorithms alerted us to Professor Menell's work. Sonia K. Katyal, *Algorithmic Civil Rights*, 103 IOWA L. REV. (forthcoming 2017). This section adds to Professor Menell's and now the law's whistleblower protection by arguing for the public interest cause of action.

255. *See* Menell, *supra* note 254, at 29 ("The routine use of blanket NDAs [key mechanisms in protecting trade secrets] by a broad swath of enterprises throughout the economy undermines society's interest in reporting illegal activity.").

256. *See, e.g.*, MELVIN F. JAGER, TRADE SECRETS LAW § 1:3 (2013) (noting commercial morality origins of trade secret law); Eastman Co. v. Reichenbach, 20 N.Y.S. 110, 116 (N.Y. Sup. Ct. 1892), *aff'd sub nom.* Eastman Kodak Co. v. Reighenbach, 29 N.Y.S. 1143 (N.Y. Gen. Term 1894). On the general history of these origins, see Menell, *supra* note 254, at 14–15.

has another guiding principle, protecting technological progress.[257] By protecting any information, "including a formula, pattern, compilation, program, device, method, technique, or process," as long as such information creates economic value by not being public and the holder of the information takes reasonable steps to keep it secret,[258] the Uniform Trade Secret Act ("UTSA") addresses the "contemporary business environment marked by high employee mobility and cybercrime."[259] The adoption of the UTSA in forty-seven states and the District of Columbia and the requirement that one take reasonable steps to protect a trade secret, means that companies have embraced the use of Non-Disclosure Agreements ("NDA") with employees and required that all trade secret material be returned when the employee leaves the company.[260] But as Professor Menell puts it, "uncritical protection of all secret business information can conflict with effective law enforcement."[261] As stated during the passage of the Sarbanes-Oxley Act of 2002 ("SOX"), "[w]ith an unprecedented portion of the American public investing in [publicly-traded] companies and depending upon their honesty, . . . [the lack of whistleblower protection for private-sector whistleblowers did] not serve the public good."[262] Similarly, with an unprecedented portion of decision-making with due process and vital verification interests at stake being processed through software, protection for employees who blow the whistle on software companies who knowingly violate the law is vital. In other words, the government needs private actors to aid in law enforcement, and there is a long history of private citizens aiding in law enforcement by providing support to public prosecution and through "private enforcement and private evidence gathering."[263]

Whistleblowing plays a large part of such efforts,[264] but if the pertinent information is a trade secret and the evidence is covered by an NDA and an agreement not to take trade secret documents, a potential whistleblower, and by extension the government, cannot investigate "illegal conduct, . . . [such as acts that] undermin[e] civil rights, public health, environmental protection, and compliance with government contracts."[265] Laws such as SOX and the Dodd-Frank Wall Street Reform and Consumer Protection Act (Dodd-Frank) provide whistleblower protection to aid in detecting illegal activity in the financial

257. *See* Menell, *supra* note 254, at 14–15.
258. UNIF. TRADE SECRETS ACT § 1(4) (NAT'L CONF. OF COMM'RS ON UNIF. STATE LAWS 1985).
259. Menell, *supra* note 254, at 18.
260. *See id.* at 16–17.
261. *Id.* at 18.
262. 148 CONG. REC. S7418-01, S7420 (daily ed. July 26, 2002) (statement of Sen. Leahy).
263. Menell, *supra* note 254, at 22.
264. *See id.* at 24.
265. *Id.* at 18.

sector.[266] The IRS also has a whistleblower program and offers incentives to whistleblowers.[267] The passage of the Environmental Protection Act led to follow up acts to protect whistleblowers who reported violations related to safe drinking water, toxic substances, and clean air.[268] In addition, the Defend Trade Secrets Act of 2016 ("DTSA") has a whistleblower provision regarding the use of trade secrets modeled on Professor Menell's approach.[269]

The approach addresses whistleblower protection well, but we offer that additional protection from employer retaliation and a new private right of action would improve the system. Under the DTSA, a potential whistleblower has immunity so long as she makes trade secret disclosures in confidence or under seal:

> An individual shall not be held criminally or civilly liable under any Federal or State trade secret law for the disclosure of a trade secret that —
>
> (A) is made —
>
> (i) in confidence to a Federal, State, or local government official, either directly or indirectly, or to an attorney; and
>
> (ii) solely for the purpose of reporting or investigating a suspected violation of law; or
>
> (B) is made in a complaint or other document filed in lawsuit or other proceeding, if such filing is made under seal.[270]

This part of the approach has several benefits. First, 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 for software companies. Because the DTSA is a federal law, the potential whistleblower no longer has to navigate whether they are governed by whatever law, or lack of law, addresses the action in a given state or country. Second, a potential whistleblower must make the submission "in confidence."[271] This provision addresses

---

266. Sarbanes-Oxley Act of 2002, Pub. L. No. 107-204 § 806, 116 Stat. 745, 802 (2002); Dodd-Frank Wall Street Reform and Consumer Protection Act of 2010, Pub. L. No. 111-203, §§ 748, 922, 124 Stat. 1376, 1739, 1841 (2010).

267. *See* 26 U.S.C. § 7623 (2012).

268. *See* Menell, *supra* note 254, at 20; Richard E. Condit, *Providing Environmental Whistleblowers with Twenty-First Century Protections*, 2 AM. U. LAB. & EMP. L.F. 31, 39 (2011) (detailing the numerous whistleblower provisions).

269. *See* Menell, *supra* note 254, at 62.

270. 18 U.S.C. § 1833(b)(1) (2016).

271. *Id.* § 1833(b)(1)(A)(i).

many, but not all, of the concerns about the trade secret being disclosed because the trade secret is not widely shared.[272] Instead, it is under seal or shared with attorneys, both of which maintain confidentiality of the information.[273] Disclosure to the press would not be covered unless the government has found that the information is not secret.[274] Third, the immunity applies for "a suspected violation of law."[275] As such whether the public or private sector develops software, someone who suspects a violation of law would be able to report that suspicion. Fourth, it provides civil and criminal immunity for the whistleblower, which means neither private contracts, such as NDAs, nor criminal sanctions that may apply could be used to go after, and so deter, the potential whistleblower. Fifth, another related protection is that if the whistleblower follows the immunity rules, she can use the information as part of her defense to a retaliation lawsuit. Although anti-retaliation protection of this type is important, unlike the False Claims Act ("FCA"), SOX, and Dodd-Frank, the DTSA appears to lack a provision to protect whistleblowers against such employment suits.[276] In addition, there is not yet a *qui tam* or bounty program — as the FCA, SOX, Dodd-Frank, and the IRS whistleblower program have — to encourage reporting of illegal conduct.[277]

Making retaliation for disclosure of trade secrets a felony under the DTSA and providing a public interest right of action similar to the FCA

---

272. *See* Menell, *supra* note 254, at 59–60 (discussing limits of the approach such as possible misconduct by "government agencies or officers").

273. *See id.* at 47–48.

274. *See id.* at 47.

275. 18 U.S.C. § 1833(b)(1)(A)(ii).

276. *See* Menell, *supra* note 254, at 26–29 (tracing the history of anti-retaliation as part of whistleblowing). As Menell notes, other acts protect against retaliation as well. *Id.* at 31 (citing Clean Water Act of 1972 § 507(a), 33 U.S.C. § 1367(a) (1972); American Recovery and Reinvestment Act of 2009, Pub. L. No. 111-5, § 1553(a), 123 Stat. 115, 297 (2009); Pipeline Safety Improvement Act of 2002, Pub. L. No. 107-355 § 6, 116 Stat. 2985, 2989 (2002)).

277. *See* Menell, *supra* note 254, at 24–29 (tracing the history of *qui tam* laws as part of whistleblowing). Bounties are one way to provide incentives to whistleblowers; the main point is that money helps motivate whistleblowers, in part, because whistleblowers want the income and in part because financial incentives help mitigate repercussions from whistleblowing. *See* Stavros Gadinis & Colby Mangels, *Collaborative Gatekeepers*, 73 WASH. & LEE L. REV. 797, 828–29 (2016) (explaining the problems of blacklisting and Dodd-Frank bounty system design as a way to mitigate that problem). Although the IRS does not use a bounty system in the *qui tam* style, the IRS does provide a monetary incentive. *See* Karie Davis-Nozemack & Sarah Webber, *Paying the IRS Whistleblower: A Critical Analysis of Collected Proceeds*, 32 VA. TAX REV. 77, 82 (2012). The point is that an incentive must be in place and somewhat certain to be paid. Thus, as Professors Davis-Nozemack and Webber explain, although the IRS's Whistleblower Program was changed to increase payments and thus increase tips, the way in which the Whistleblower Program interpreted "collected proceeds" — the pool of money from which a whistleblower would be paid — hindered and reduced payments. *See id.* at 78. They suggested changes to the way collected proceeds were defined and paid out as way to improve certainty for potential whistleblowers and increase long-term willingness to use the program. *See id.* at 130–32.

or similar state laws would add to the way the DTSA encourages private help in enforcing the law in the software context. By making retaliation a felony there would be greater protection for whistleblowers.[278] That protection will not 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 of getting a new job after blowing the whistle.[279] Issues of civil rights, due process, and the integrity of our democratic systems affect the United States government in at least two important ways. First, there can be a direct harm to the government's pocketbook because the government would have paid for illegal software, lost time and money, and have to remedy the failure. Second, there is a general harm to the government's perception with U.S. citizens and even people around the world because people need to trust that the government is using sound and fair systems. Protecting whistleblowers in these ways at a federal level shows a commitment by the government to take these issues seriously and to stop them.

On top of whistleblower protection, we think 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 in building a system to govern the use of software.[280] Our model derives from *qui tam* actions under the FCA, which dates back to 1863,[281] which allows "any

278. This is similar to the anti-retaliation protection afforded to witnesses, victims, and government informants. *See* 18 U.S.C. § 1513 (2008) (making it a felony to kill, attempt to kill, or cause bodily injury to a witness who testifies at an official proceeding).

279. *See* Gadinis & Mangels, *supra* note 277, at 829.

280. Some argue that such an approach causes a "gold rush" effect on litigation, but a recent empirical study indicates such claims are overstated. *See* David Freeman Engstrom, *Private Enforcement's Pathways: Lessons from Qui Tam Litigation*, 114 COLUM. L. REV. 1913, 1922 (2014).

281. *See* Julie Anne Ross, Comment, *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"); *see also* 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."). 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, 607–18 (2005) (examining private attorney general actions and describing pro-*qui tam* arguments — "private 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."). 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. *See id.* at 610–18 (finding "no definitive resolution to the policy debate over private attorneys general" and that legislatures resolve the matter differently based on context); *see also* John Beisner, Matthew Shors & Jessica Davidson Miller, *Class Action 'Cops': Public Servants or Private Entrepreneurs?*, 57 STAN. L. REV. 1441, 1457–60 (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

person to prosecute a civil fraud — in the name of the United States — against any person who allegedly makes a false claim to the United States government"[282] and California's Safe Drinking Water and Toxic Enforcement Act of 1986 ("Proposition 65" or "Act"), which addresses environmental concerns by allowing a public interest cause of action balanced against governmental interests in pursuing such matters.[283] 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 vendor software and regulated private sector cases.[284]

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.[285] The California approach sets the penalty at $2,500 per violation as some statutory amount is needed because of the lack of money at issue under California law for the behavior in question. We do not take a position on the correct amount for software. Rather we note that the amount must be large enough to provide an 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, but where important yet 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 this chapter 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

---

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).

282. Saikrishna Prakash, *The Chief Prosecutor*, 73 GEO. WASH. L. REV. 521, 531 (2005).

283. *See* CAL. HEALTH & SAFETY CODE § 25249.7 (2009).

284. 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 permits accountability; just this option to police the software.

285. *See id.* (using the same levers for environmental protection).

> regulated community as a whole; (G) Any other factor that justice may require.[286]

Given the concentration of software companies in California and Washington, either state could pass a state law and have a large effect on the software industry.[287] But 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 of a city [with] a population in excess of 750,000 [people]" could bring a suit on their own.[288] Government attorneys in smaller jurisdictions could bring suits provided they receive permission from their district attorney.[289]

Under this approach, non-governmental actors could bring an action in the public interest under certain conditions.[290] 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 sixty days as with the California law before bringing the suit.[291] The waiting period could be shorter or longer, but it would allow the state to decide whether to act. If the State picked up the case and was "diligently prosecuting" it, the private action would no longer be allowed.[292]

As another way to limit frivolous lawsuits, the person bringing the suit would have to provide a certificate of merit which would verify that:

---

286. *Id.* Given the importance of context for such a piece of legislation, it is beyond the scope of this Article to set out what is correct in all cases. But as an example, if one had a cause of action to allow a suit against a maker of voting machines that failed to follow specified software requirements, one could choose to assess the penalty based on the number of machines as we have noted. Yet given that voter trust in the system is vital to voting and enfranchisement, one might wish to base the penalty on the number of voters who used the machines in question.

287. For example, because California is a large market for autos, the California Air Resources Board has affected national practices for emissions. *See* Paul Rogers, *California's 'Clean Car' Rules Help Remake Auto Industry*, YALE ENV'T 360 (Feb. 8, 2012), http://e360.yale.edu/features/californias_clean_car_rules_help_remake_us_auto_industry [https://perma.cc/5MMF-JDAT]. *But see* Hiroko Tabuchi, *California Upholds Auto Emissions Standards, Setting Up Face-Off With Trump*, N.Y. TIMES (Mar. 24, 2017), https://www.nytimes.com/2017/03/24/business/energy-environment/california-upholds-emissions-standards-setting-up-face-off-with-trump.html?mcubz=3 (last visited Dec. 19, 2017) (noting that California requires a waiver from the federal government for California emissions rules and that the Trump administration opposes such strict emission standards). State executive branches can also be effective in changing policy. For example, state attorneys general have used their offices to experiment and expand different ways to protect privacy. *See generally* Danielle Keats Citron, *The Privacy Policymaking of State Attorneys General*, 92 NOTRE DAME L. REV. 747 (2016).

288. *See* HEALTH & SAFETY § 25249.7(c).

289. *See id.*

290. *See id.* § 25249.7(d).

291. *See id.* § 25249.7(d)(1).

292. *See id.* § 25249.7(d)(2).

> [T]he person executing the certificate has consulted with one or more persons with relevant and appropriate experience or expertise who has 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.[293]

Furthermore, "[f]actual 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."[294] Insofar as such material is covered by a trade secret, the DTSA now enables a potential whistleblower to provide the information. 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,"[295] which would help protect potential whistleblowers who might provide such material.

After a case is over, however, the alleged violator could move for, or the court of its own volition could conduct, an in camera review of the certificate of merit and the legal theories of the plaintiff to see whether the basis was real. [296] If the court deems the basis to be false, it could bring sanctions for a frivolous lawsuit.[297] California's provision on which we draw states, "[i]f 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, then *the action shall be deemed frivolous* within the meaning of Section 128.7 of the Code of Civil Procedure."[298] These two sections and their adaptation to our approach are vital, because one section allows the court to order the bringer of the frivolous lawsuit to pay attorney's fees and reasonable expenses,[299] 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.[300] 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

---

293. *Id.* § 25249.7(d)(1).
294. *Id.*
295. *Id.* § 25249.7(h)(1).
296. *See id.* § 25249.7(h)(2).
297. *See id.*
298. *Id.* (emphasis added).
299. *See* CAL. CODE CIV. PROC. § 128.6(a) (2005).
300. *Id.* § 128.7(d)–(h) ("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.").

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.

VII. 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 decisions are made — especially in the occasional cases where no answer is available — fuels distrust. In addition, just the specter of the lack of integrity of voting machine counts undermines faith in the political process. In the private sector, evidence of misuse of software by the auto industry, such as with Volkswagen and recently at Fiat-Chrysler, erodes 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 tests 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 at some level, 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 to thrive 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 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 allow for analyzability and technical accountability. Or rather, software can be built so that we can trust, but verify.