DOI:10.1145/3448248

The pursuit of responsible AI raises the ante on both the trustworthy computing and formal methods communities.

BY JEANNETTE M. WING

Trustworthy AI

FOR CERTAIN TASKS, AI systems have achieved good enough performance to be deployed in our streets and our homes. Object recognition helps modern cars see. Speech recognition helps personalized voice assistants, such as Siri and Alexa, converse. For other tasks, AI systems have even exceeded human performance. AlphaGo was the first computer program to beat the best Go player in the world.

The promise of AI is huge. They will drive our cars. They will help doctors diagnose disease more accurately.[54] They will help judges make more consistent court decisions. They will help employers hire more suitable job candidates.

However, we know these AI systems can be brittle and unfair. Adding graffiti to a stop sign fools the classifier into saying it is not a stop sign.[22] Adding noise to an image of a benign skin lesion fools the classifier into saying it is malignant.[23] Risk assessment tools used in U.S. courts have shown to be biased against blacks.[4] Corporate recruiting tools have been shown to be biased against women.[17]

How then can we deliver on the promise of the benefits of AI but address these scenarios that have life-critical consequences for people and society? In short, how can we achieve *trustworthy AI?*

The ultimate purpose of this article is to rally the computing community to support a broad-based, long-term research program on trustworthy AI, drawing on the expertise and sensibilities from multiple research communities and stakeholders. This article focuses on addressing three key research communities because: trustworthy AI adds new desired properties above and beyond those for *trustworthy computing*; AI systems require new *formal methods* techniques, and in particular, the role of data raises brand new research questions; and *AI* systems can likely benefit from the scrutiny of formal methods for ensuring trustworthiness. By bringing together researchers in trustworthy computing, formal methods, and AI, we aim to foster a new research community across academia, industry, and government in trustworthy AI.

From Trustworthy Computing to Trustworthy AI

The landmark *Trust in Cyberspace* 1999 National Academies report lay the foundations of trustworthy computing and what continues to be an active research area.[41]

Around the same time, the National Science Foundation started a series of programs on trust. Starting with

key insights

- **The set of trustworthiness properties for AI systems, in contrast to traditional computing systems, needs to be extended beyond reliability, security, privacy, and usability to include properties such as probabilistic accuracy under uncertainty, fairness, robustness, accountability, and explainability.**
- **To help ensure their trustworthiness, AI systems can benefit from the scrutiny of formal methods.**
- **AI systems raise the bar on formal methods for two key reasons: the inherent probabilistic nature of machine-learned models, and the critical role of data in training, testing, and deploying a machine-learned model.**

IMAGE BY ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK

Trusted Computing (initiated in 2001), then Cyber Trust (2004), then Trustworthy Computing (2007), and now Secure and Trustworthy Cyberspace (2011), the Computer and Information Science and Engineering Directorate has grown the academic research community in trustworthy computing. Although it started within the computer science community, support for research in trustworthy computing now spans multiple directorates at NSF and engages many other funding organizations, including, through the Networking and Information Technology Research and Development (NITRD) Program, 20 federal agencies.

Industry has also been a leader and active participant in trustworthy computing. With Bill Gates's January 2002 "Trustworthy Computing" memo,[26] Microsoft signaled to its employees, customers, shareholders, and the rest of the information technology sector the importance of trustworthy software and hardware products. It referred to an internal Microsoft white paper, which identified four pillars to trustworthiness: security, privacy, reliability, and business integrity.

After two decades of investment and advances in research and development, *trustworthy* has come to mean a set of (overlapping) properties:

▸ Reliability: Does the system do the right thing?

▸ Safety: Does the system do no harm?

▸ Security: How vulnerable is the system to attack?

▸ Privacy: Does the system protect a person's identity and data?

▸ Availability: Is the system up when I need to access it?

▸ Usability: Can a human use it easily?

The *computing* systems for which we want such properties to hold are hardware and software systems, including their interaction with humans and the physical world. Academia and industry have made huge strides in trustworthy computing in the past decades. However, as technology advances and as adversaries get more sophisticated, trustworthy computing remains a holy grail.

AI systems raise the bar in terms of the set of properties of interest. In addition to the properties associated with trustworthy computing (as noted), we also want (overlapping) properties such as:

- Accuracy: How well does the AI system do on new (unseen) data compared to data on which it was trained and tested?
- Robustness: How sensitive is the system's outcome to a change in the input?
- Fairness: Are the system outcomes unbiased?
- Accountability: Who or what is responsible for the system's outcome?
- Transparency: Is it clear to an external observer how the system's outcome was produced?
- Interpretability/Explainability: Can the system's outcome be justified with an explanation that a human can understand and/or that is meaningful to the end user?
- Ethical: Was the data collected in an ethical manner? Will the system's outcome be used in an ethical manner?
- ...and others, yet to be identified

The machine learning community considers accuracy as a gold standard, but trustworthy AI requires us to explore trade-offs among these properties. For example, perhaps we are willing to give up on some accuracy in order to deploy a fairer model. Also, some of the above properties may have different interpretations, leading to different formalizations. For example, there are many reasonable notions of fairness,[40] including demographic parity, equal odds, and individual fairness,[20] some of which are incompatible with each other.[12,33]

Traditional software and hardware systems are complex due to their size and the number of interactions among their components. For the most part, we can define their behavior in terms of discrete logic and as deterministic state machines.

Today's AI systems, especially those using deep neural networks, add a dimension of complexity to traditional computing systems. This complexity is due to their inherent probabilistic nature. Through probabilities, AI systems model the uncertainty of human behavior and the uncertainty of the physical world. More recent advances in machine learning, which rely on big data, add to their probabilistic nature, as data from the real world are just points in a probability space. Thus, trustworthy AI necessarily directs our attention from the primarily deterministic nature of traditional computing systems to the probabilistic nature of AI systems.

Verify, to Trust

How can we design, implement, and deploy AI systems to be trustworthy?

One approach for building end-user trust in computing systems is formal verification, where properties are proven once and for all over a large domain, for example, for all inputs to a program or for all behaviors of a concurrent or distributed system. Alternatively, the verification process identifies a counterexample, for example, an input value where the program produces the wrong output or a behavior that fails to satisfy the desired property, and thus provides valuable feedback on how to improve the system. Formal verification has the advantage of obviating the need to test individual input values or behaviors one-by-one, which for large (or infinite) state spaces is impossible to achieve completely. Early success stories in formal methods, for example, in verifying cache coherence protocols[48] and in detecting device driver bugs,[5] led to their scalability and practicality today. These approaches are now used in the hardware and software industry, for example, Intel,[29] IBM,[6] Microsoft,[5] and Amazon.[15,44] Due to advances in formal methods languages, algorithms, and tools, and to the increased scale and complexity of hardware and software, we have seen in the past few years a new surge of interest and excitement in formal verification, especially for ensuring the correctness of critical components of system infrastructure.[7,10,11,15,27,30,34,49]

Formal verification is a way to provide provable guarantees and thus increase one's trust that the system will behave as desired.

From traditional formal methods to formal methods for AI. In traditional formal methods, we want to show that a model M *satisfies* ($\models$) a property P.

$$M \models P$$

M is the object to be verified—be it a program or an abstract model of a complex system, for example, a concurrent, distributed, or reactive system. P is the correctness property, expressed in some discrete logic. For example, M might be a concurrent program that uses locks for synchronization and P might be "deadlock free." A proof that M is deadlock free means any user of M is assured that M will never reach a deadlocked state. To prove that M satisfies P, we use formal mathematical logics, which are the basis of today's scalable and practical verification tools such as model checkers, theorem provers, and satisfiability modulo theories (SMT) solvers.

Especially when M is a concurrent, distributed, or reactive system, in traditional formal methods, we often add explicitly a specification of a system's environment E in the formulation of the verification task:

$$E, M \models P$$

For example, if M is a parallel process, E might be another process with which M interacts (and then we might write $E \,||\, M \models P$, where $||$ stands for parallel composition). Or, if M is device driver code, E might be a model of the operating system. Or, if M is a control system, E might be a model of its environment that closes the control loop. The specification of E is written to make explicit the assumptions about the environment in which the system is to be verified.

For verifying AI systems, M could be interpreted to be a complex system, for example, a self-driving car, within which is a component that is a machine-learned model, for example, a computer vision system. Here, we would want to prove P, for example, safety or robustness, with respect to M (the car) in the context of E (traffic, roads, pedestrians, buildings, and so on). We can view proving P as proving a "system-level" property. Seshia et al. elaborate on the formal specification challenges with this perspective,[51] where a deep neural network might be a black-box component of the system M.

But what can we assert about the machine learned model, for example, a DNN, that is a critical component of the system? Is there a robustness or fairness property we can verify of the machine-learned model itself? Are there white-box verification techniques that can take advantage of the structure of the machine learned model? Answering these questions raises new verification challenges.

Verifying a machine-learned model M. For verifying an ML model, we reinterpret M and P: M stands for a machine-learned model. P stands for a trustworthy property, for example, safety, robustness, privacy, or fairness.

Verifying AI systems ups the ante over traditional formal methods. There are two key differences: the inherent probabilistic nature of the machine-learned model and the role of data.

The inherent probabilistic nature of M and P, and thus the need for probabilistic reasoning (⊨). The ML model, M, itself is semantically and structurally different from a typical computer program. As mentioned, it is inherently probabilistic, taking inputs from the real world, that are perhaps mathematically modeled as a stochastic process, and producing outputs that are associated with probabilities. Internally, the model itself operates over probabilities; for example, labels on edges in a deep neural network are probabilities and nodes compute functions over these probabilities. Structurally, because a machine generated the ML model, M itself is not necessarily something human readable or comprehensible; crudely, a DNN is a complex structure of if-then-else statements that would unlikely ever be written by a human. This "intermediate code" representation opens up new lines of research in program analysis.

The properties P themselves may be formulated over continuous, not (just) discrete domains, and/or using expressions from probability and statistics. Robustness properties for deep neural networks are characterized as predicates over continuous variables.[18] Fairness properties are characterized in terms of expectations with respect to a loss function over reals (for example, see Dwork et al.[20]). Differential privacy is defined in terms of a difference in probabilities with respect to a (small) real value.[21] Note that just as with properties such as usability for trustworthy computing, some desired properties of trustworthy AI systems, for example, transparency or ethics, have yet to be formalized or may not be formalizable. For such properties, a framework that considers legal, policy, behavioral and social rules and norms could provide the context within which a formalizable question can be answered. In short, verification of AI systems will be limited to what can be formalized.

These inherently probabilistic models M and associated desired trust properties P call for scalable and/or new verification techniques that work over reals, non-linear functions, probability distributions, stochastic processes, and so on. One stepping-stone to verifying AI systems is probabilistic logics and hybrid logics (for example, Alur et al.,[3] Kwiatkowska et al.[35] and Platzer[46]), used by the cyber-physical systems community. Another approach is to integrate temporal logic specifications directly in reinforcement learning algorithms.[24] Even more challenging is that these verification techniques need to operate over machine-generated code, in particular code that itself might not be produced deterministically.[a]

Formal verification is a way to provide provable guarantees and thus increase one's trust that the system will behave as desired.

The role of data. Perhaps the more significant key difference between traditional formal verification and verification for AI systems is the role of data—data used in training, testing, and deploying ML models. Today's ML models are built and used with respect to a set, D, of data. For verifying an ML model, we propose to make explicit the assumptions about this data, and formulate the verification problem as:

$$D, M \models P$$

Data is divided into *available data* and *unseen data*, where *available data* is data-at-hand, used for training and testing M; and *unseen data* is data over which M needs (or is expected) to operate without having seen it before. The whole idea behind building M is so that based on the data on which it was trained and tested, M would be able to make predictions on data it has never seen before, typically to some degree of accuracy.

Making the role of data explicit raises novel specification and verification challenges, roughly broken into these categories, with related research questions:

Collection and partitioning of available data:

▸ How much data suffices to build

[a] The ways in which machine learning models, some with millions of parameters, are constructed today, perhaps through weeks of training on clusters of CPUs, TPUs, and GPUs, raise a meta-issue of trust: scientific reproducibility.

a model M for a given property P? The success of deep learning has taught us that with respect to accuracy, the more data, the better the model, but what about other properties? Does adding more data to train or test M make it more robust, fairer, and so on, or does it not have an effect with respect to the property P? What new kind of data needs to be collected if a desired property does not hold?

▸ How do we partition an available (given) dataset into a training set and a test set? What guarantees can we make of this partition with respect to a desired property P, in building a model M? Would we split the data differently if we were training the model with respect to multiple properties at the same time? Would we split the data differently if we were willing to trade one property over another?

Specifying unseen data: Including D in the formal methods framework $D, M \models P$ gives us the opportunity to state explicitly assumptions about the unseen data.

▸ How do we specify the data and/or characterize properties of the data? For example, we could specify D as a stochastic process that generates inputs over which the ML model needs to be verified. Or, we could specify D as a data distribution. For a common statistical model, for example, a normal distribution, we could specify D in terms of its parameters, for example, mean and variance. Probabilistic programming languages, for example, Stan,[8] might be a starting point for specifying statistical models. But what of large real-world datasets that do not fit common statistical models, or which have thousands of parameters?

▸ In specifying unseen data, by definition, we will need to make certain assumptions about the unseen data. Would these assumptions not then be the same as those we would make to build the model M in the first place? More to the point: *How can we trust the specification of D?* This seemingly logical deadlock is analogous to the problem in traditional verification, where given an M, we need to assume the specifications of the elements E and P are "correct" in the verification task $E, M \models P$. Then in the verification process, we may need to modify E and/or P (or even M). To break the circular reasoning at hand, one approach is to use a different validation approach for checking the specification of D; such approaches could borrow from a repertoire of statistical tools. Another approach would be to assume an initial specification is small or simple enough that it can be checked by (say, manual) inspection; then we use this specification to bootstrap an iterative refinement process. (We draw inspiration from the counterexample guided abstraction and refinement method[14] of formal methods.) This refinement process may necessitate modifying D, M, and/or P.

The formal methods community has recently been exploring robustness properties of AI systems, in particular, image processing systems used in autonomous vehicles.

▸ How does the specification of unseen data relate to the specification of the data on which M was trained and tested?

In traditional verification, we aim to prove property, P, a universally quantified statement: for example, *for all* input values of integer variable *x*, the program will return a positive integer; or *for all* execution sequences *x*, the system will not deadlock.

So, the first question for proving P of an ML model, M, is: in P, what do we quantify over? For an ML model that is to be deployed in the real world, one reasonable answer is to quantify over data distributions. But a ML model is meant to work only for certain distributions that are formed by real world phenomena, and *not* for arbitrary distributions. We do not want to prove a property *for all* data distributions. This insight on the difference in what we quantify over and what the data represents for proving a trust property for M leads to this novel specification question:

▸ How can we specify the class of distributions over which P should hold for a given M? Consider robustness and fairness as two examples:

▹ For robustness, in the adversarial machine learning setting, we might want to show that M is robust to all norm-bounded perturbations D. More interestingly, we might want to show M is robust to all "semantic" or "structural" perturbations for the task at hand. For example, for some vision tasks, we want to consider rotating or darkening an image, not just changing any old pixel.

▹ For fairness, we might want to show the ML model is fair on a given dataset and all unseen datasets that are "similar" (for some formal notion of "similar"). Training a recruiting tool to decide whom to interview on one population of applicants should ideally be fair on any future population. How can we specify these related distributions?

Toward building a fair classifier that is also robust, Mandal et al. show how to adapt an online learning algorithm that finds a classifier that is fair over a *class* of input distributions.[37]

Verification task: Once we have a specification of D and P, given an M, we are then left with verifying that M satisfies P, given any assumptions we have made explicit about available and unseen data in D, using whatever logical framework (⊨) we have at hand.

- How do we check the available data for desired properties? For example, if we want to detect whether a dataset is fair or not, what should we be checking about the dataset?
- If we detect the property does not hold, how do we fix the model, amend the property, or decide what new data to collect for retraining the model? In traditional verification, producing a counterexample, for example, an execution path that does not satisfy P, helps engineers debug their systems and/or designs. What is the equivalent of a "counterexample" in the verification of an ML model and how do we use it?
- How do we exploit the explicit specification of unseen data to aid in the verification task? Just as making explicit the specification of the environment, E, in the verification task E, M ⊨ P, how can we leverage having an explicit specification of D?
- How can we extend standard verification techniques to operate over data distributions, perhaps taking advantage of the ways in which we formally specify unseen data?

These two key differences—the inherent probabilistic nature of M and the role of data D—provide research opportunities for the formal methods community to advance specification and verification techniques for AI systems.

Related work. The formal methods community has recently been exploring robustness properties of AI systems,[18] in particular, image processing systems used in autonomous vehicles. The state-of-the-art VerifAI system[19] explores the verification of robustness of autonomous vehicles, relying on simulation to identify execution traces where a cyber-physical system (for example, a self-driving car) whose control relies on an embedded ML model could go awry. Tools such as ReluVal[56] and Neurify[57] look at robustness of DNNs, especially as applied to safety of autonomous vehicles, including self-driving cars and aircraft collision avoidance systems. These tools rely on interval analysis as a way to cut down on state exploration, while still providing strong guarantees. A case study using Verisig to verify the safety of a DNN-based controller for the F1/10 racing car platform provides a benchmark for comparing different DNN configurations and sizes of input data and identifies a current gap between simulation and verification.[32]

FairSquare[2] uses probabilistic verification to verify fairness of ML models. LightDP[60] transforms a probabilistic program into a non-probabilistic one, and then does type inference to automate verification of privacy budgets for differential privacy.

These pieces of work are in the spirit of trustworthy AI, but each focuses on only one trust property. Scaling their underlying verification techniques to industry-scale systems is still a challenge.

Additional formal methods opportunities. Today's AI systems are developed to perform a particular task in mind, for example, face recognition or playing Go. How do we take into consideration the task that the deployed ML model is to perform in the specification and verification problem? For example, consider showing the robustness of a ML model, M, that does image analysis: For the task of identifying cars on the road, we would want M to be robust to the image of any car that has a dent in its side; but for the task of quality control in an automobile manufacturing line, we would not.

Previously, we focused on the verification task in formal methods. But the machinery of formal methods has also successfully been used recently for program synthesis.[28] Rather than post-facto verification of a model M, can we develop a "correct-by-construction" approach in building M in the first place? For example, could we add the desired trustworthy property, P, as a constraint as we train and test M, with the intention of guaranteeing that P holds (perhaps for a given dataset or for a class of distributions) at deployment time? A variant of this approach is to guide the ML algorithm design process by checking at each step that the algorithm never satisfies an undesirable behavior.[53] Similarly, safe reinforcement learning addresses learning policies in decision processes where safety is added as a factor in optimization or an external constraint in exploration.[25]

The laundry list of properties enumerated at the outset of this article for trustworthy AI is unwieldy, but each is critical toward building trust. A task ahead for the research community is to formulate commonalities across these properties, which can then be specified in a common logical framework, akin to using temporal logic[38,47] for specifying safety ("nothing bad happens") and liveness ("something good eventually happens") properties[36] for reasoning about correctness properties of concurrent and distributed systems.

Compositional reasoning enables us to do verification on large and complex systems. How does verifying a component of an AI system for a property "lift" to showing that property holds for the system? Conversely, how does one decompose an AI system into pieces, verify each with respect to a given property, and assert the property holds of the whole? Which properties are global (elude compositionality) and which are local? Decades of research in formal methods for compositional specification and verification give us a vocabulary and framework as a good starting point.

Statistics has a rich history in model checking[b] and model evaluation, using tools such as sensitivity analysis, prediction scoring, predictive checking, residual analysis, and model criticism. With the goal of validating an ML model satisfies a desired property, these statistical approaches can complement formal verification approaches, just as testing and simulation complement verification of computational systems. Even more relevantly, as mentioned in "The role of data" noted earlier, they can help with the evaluation of any sta-

[b] Not to be confused with formal method's notion of model checking, where a finite state machine (computational model of a system) is checked against a given property specification.[13,50]

tistical model used to specify unseen data, D, in the D, M ⊨ P problem. An opportunity for the formal methods community is to combine these statistical techniques with traditional verification techniques (for early work on such a combination, see Younes et al.[59]).

Building a Trustworthy AI Community

Just as for trustworthy computing, formal methods is only one approach toward ensuring increased trust in AI systems. The community needs to explore many approaches, especially in combination, to achieve trustworthy AI. Other approaches include testing, simulation, run-time monitoring, threat modeling, vulnerability analysis, and the equivalent of design and code reviews for code *and data*. Moreover, besides technical challenges, there are societal, policy, legal, and ethical challenges.

On October 30–November 1, 2019, Columbia University's Data Science Institute hosted an inaugural Symposium on Trustworthy AI[1] sponsored by Capital One, a DSI industry affiliate. It brought together researchers from formal methods, security and privacy, fairness, and machine learning. Speakers from industry brought a reality check to the kinds of questions and approaches the academic community are pursuing. The participants identified research challenge areas, including:

- Specification and verification techniques;
- "Correctness-by-construction" techniques;
- New threat models and system-level adversarial attacks;
- Processes for auditing AI systems that consider properties such as explainability, transparency, and responsibility;
- Ways to detect bias and de-bias data, machine learning algorithms, and their outputs;
- Systems infrastructure for experimenting for trustworthiness properties;
- Understanding the human element, for example, where the machine is influencing human behavior; and
- Understanding the societal element, including social welfare, social norms, morality, ethics, and law.

Technology companies, many of which push the frontiers of machine learning and AI, have not been sitting still. They realize the importance of trustworthy AI for their customers, their business, and social good. Of predominant concern is fairness. IBM's AI Fairness 360 provides an open source toolkit to check for unwanted bias in datasets and machine learning models.[55] Google's TensorFlow kit provides "fairness indicators" for evaluating binary and multi-class classifiers for fairness.[31] Microsoft's Fairlearn is an open source package for machine learning developers to assess their systems' fairness and to mitigate observed unfairness.[39] At its F8 conference in 2018, Facebook announced its Fairness Flow tool intended "to measure for potential biases for or against particular groups of people."[52] In the spirit of industry and government collaborations, Amazon and the National Science Foundation have partnered since 2019 to fund a "Fairness in AI" program.[43]

In 2016, DARPA focused on explainability by launching the Explainable AI (XAI) Program.[16] The goal of this program was to develop new machine learning systems that could "explain their rationale, characterize their strengths and weaknesses, and convey an understanding of how they will behave in the future." With explainability would come increased trust by an end user to believe and adopt the outcome of the system.

Through the Secure and Trustworthy Cyberspace Program, NSF funds a Center on Trustworthy Machine Learning[9] led by Penn State University and involving researchers from Stanford, UC Berkeley, UC San Diego, University of Virginia, and University of Wisconsin. Their primary focus is on addressing adversarial machine learning, complementary to the formal methods approach outlined previously. (In the interests of full disclosure, the author is on this Center's Advisory Board.) In October 2019, the National Science Foundation announced a new program to fund National AI Institutes.[42] One of the six themes it named was "Trustworthy AI," emphasizing properties such as reliability, explainability, privacy, and fairness.

The NITRD report on AI and cybersecurity calls explicitly for research in the specification and verification of AI systems and for trustworthy AI decision-making.[45] Finally, in December 2020, the White House signed an executive order on trustworthy AI to provide guidance to U.S. federal agencies in adopting AI for their services and to foster public trust in AI.[58]

Just as for trustworthy computing, government, academia, and industry are coming together to drive a new research agenda in trustworthy AI. We are upping the ante on a holy grail!

Acknowledgments

During 2002–2003, I was fortunate to spend a sabbatical at Microsoft Research and witnessed firsthand how trustworthy computing permeated the company. It was also the year when the SLAM project[5] showed how the use of formal methods could systematically detect bugs in device driver code, which at the time was responsible for a significant fraction of "blue screens of death." Whereas formal methods had already been shown to be useful and scalable for the hardware industry, the SLAM work was the first industry-scale project that showed the effectiveness of formal methods for software systems. I also had the privilege to serve on the Microsoft Trustworthy Computing Academic Advisory Board from 2003–2007 and 2010–2012.

When I joined NSF in 2007 as the Assistant Director for the Computer and Information Science and Engineering Directorate, I promoted trustworthy computing across the directorate and with other federal agencies via NITRD. I would like to acknowledge my predecessor and successor CISE ADs, and all the NSF and NITRD program managers who cultivated the community in trustworthy computing. It is especially gratifying to see how the Trustworthy Computing program has grown to the Secure and Trustworthy Cyberspace program, which continues to this day.

ACM sponsors the annual FAT* conference, which originally promoted fairness, accountability, and transparency in machine learning. The Microsoft Research FATE group added "E" for ethics. FAT* has since grown to recognize other properties, including ethics, as well as the desirability of these properties for AI systems more generally, not just machine learning. My list of trustworthy AI properties is inspired by this community.

I would like to acknowledge S. Agrawal, R. Geambasu, D. Hsu, and S. Jana for their insights into what makes verifying AI systems different from verifying traditional computing systems. Thanks to A. Chaintreau for instigating my journey on trustworthy AI. Special thanks to R. Geambasu and T. Zheng who provided comments on an earlier draft of this article. Thanks also to the anonymous reviewers for their pointers to relevant related work.

Final thanks to Capital One, JP Morgan, the National Science Foundation, and the Sloan Foundation for their support and encouragement to promote trustworthy AI.

References

1. Agrawal, S. and Wing, J.M. Trustworthy AI Symposium. Columbia University, (Oct. 30–Nov. 1, 2019); https://datascience.columbia.edu/trustworthy-ai-symposium.
2. Albarghouthi, A., D'Antoni, L., Drews, S. and Nori, A. FairSquare: Probabilistic verification of program fairness. In *Proceedings of ACM OOPSLA '17.*
3. Alur, R., Henzinger, T.A. and Ho, P.H. Automatic symbolic verification of embedded systems. *IEEE Trans. Software Eng. 22* (1996), 181–201.
4. Angwin, J., Larson, J., Mattu, S. and Kirchner, L. Machine bias. *ProPublica* (May 23, 2016).
5. Ball, T., Cook, B., Levin, V. and Rajamani, S. SLAM and Static driver verifier: Technology Transfer of formal methods inside Microsoft. Technical Report MSR-TR-2004-08. Microsoft Research, Jan. 2004.
6. Baumgartner, J. Integrating formal verification into mainstream verification: The IBM experience. *Formal Methods in Computer-Aided Design*, Haifa, Israel, 2006.
7. Bhargavan, K. et al. Everest: Towards a verified, drop-in replacement of HTTPS. In *Proceedings of the 2nd Summit on Advances in Programming Languages*, May 2017.
8. Carpenter, B et al. Stan: A probabilistic programming language. *J. Statistical Software 76*, 1 (2017); DOI 10.18637/jss.v076.i01
9. Center for Trustworthy Machine Learning; https://ctml.psu.edu
10. Chen, H., Chajed, T., Konradi, A., Wang, S., Ileri, A., Chlipala, A., Kaashoek, M.F. and N. Zeldovich, N. Verifying a high-performance crash-safe file system using a tree specification. In *Proceedings of the 26th ACM Symposium on Operating Systems Principles*, 2017.
11. Chen, H., Ziegler, D., Chajed, T., Chlipala, A., Kaashoek, M.F. and Zeldovich, N. Using crash Hoare logic for certifying the FSCQ file system. In *Proceedings of the 25th ACM Symp. Operating Systems Principles*, 2015.
12. Chouldechova, A. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. In *Proceedings of FATML*, 2016.
13. Clarke, E.M. and Emerson, E.A. Characterizing correctness properties of parallel programs using fixpoints. Automata, Languages and Programming, *Lecture Notes in Computer Science 85* (1980), 169–181
14. Clarke, E., Grumberg, O., Jha, S., Lu, Y. and Veith, H. Counterexample-guided abstraction refinement. Computer Aided Verification. E.A. Emerson, A.P. Sistla, eds. *Lecture Notes in Computer Science 1855* (2000). Springer, Berlin, Heidelberg.
15. Cook, B. Formal reasoning about the security of Amazon Web Services," *Proceedings of the International Conference on Computer Aided Verification*, Volume 10981, 2018.
16. DARPA. Explainable AI (XAI) Program. Matt Turek, Defense Advanced Research Projects Agency, 2016; https://www.darpa.mil/program/explainable-artificial-intelligence.
17. Dastin, J. Amazon scraps secret AI recruiting tool that showed bias against women. Reuters, Oct. 9, 2018.
18. Dreossi, T., Ghosh, S., Sangiovanni-Vincentelli, A.L. and Seshia, S.A. A formalization of robustness for deep neural networks. In *Proceedings of the AAAI Spring Symp. Workshop on Verification of Neural Networks*, Mar. 2019.
19. Dreossi, T., Ghosh, S., Sangiovanni-Vincentelli, A.L. and Seshia, S.A. VERIFAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *Proceedings of Intern. Conf. Computer-Aided Design*, 2019.
20. Dwork, C., Hardt, M., Pitassi, T., Reingold, O. and Zemel. R. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 2012; https://doi.org/10.1145/2090236.2090255
21. Dwork, C., McSherry, F., Nissim, K. and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Con. Theory of Cryptography*. S. Halevi and T. Rabin, Eds. Springer-Verlag, Berlin, Heidelberg, 2006, 265–284; DOI:10.1007/11681878 14
22. Eykholt, K. et al. Robust physical-world attacks on deep learning visual classification. In *Proceedings of CVPR 2017.*
23. Finlayson, S.G., Bowers, J.D., Ito, J., Zittrain, J.L., Beam, A.L., Kohane, I.S. Adversarial attacks on medical machine learning. *Science 363*, 6433 (2019), 1287–1289; DOI: 10.1126/science.aaw4399
24. Gao, Q., Hajinezhad, D., Zhang, Y., Kantaros, Y. and Zavlanos, M.M. Reduced variance deep reinforcement learning with temporal logic specifications. In *Proceedings of ACM/IEEE Intern. Conf. Cyber-Physical Systems*, 2019, 237–248.
25. Garcia, J. and Fernandez, F. A comprehensive survey on safe reinforcement learning. *J. Machine Learning Research 16* (2015), 1437–1480.
26. Gates, B. Trustworthy computing. Microsoft memo (Jan. 15, 2002); wired.com
27. Gu, R., Shao, Z., Chen, H., Wu, X.N., Kim, J., Sjberg, V. and Costanzo, D. Certikos: An extensible architecture for building certified concurrent OS kernels. *Proceedings of 12th USENIX Symp. Operating Systems Design and Implementation*, 2016.
28. Gulwani, S., Polozov, O. and Singh, R. Program Synthesis. *Foundations and Trends® in Programming Languages.* Now Publishers Inc., 2017.
29. Harrison, J. Formal verification at Intel. In *Proceedings of the 18th Annual IEEE Symp. Logic in Computer Science.* IEEE, 2003.
30. Hawblitzel, C., Howell, J., Lorch, J.R., Narayan, A., Parno, B., Zhang, D. and Zill, B. Ironclad apps: End-to-end security via automated full-system verification. In *Proceedings of the 11th USENIX Symp. Operating Systems Design and Implementation*, 2014.
31. Hutchinson, B., Mitchell, M., Xu, C., Doshi, T. Fairness indicators: Thinking about fairness evaluation, 2020; https://www.tensorflow.org/tfx/fairness_indicators/guidance.
32. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J. and Lee, I. Case study: Verifying the safety of an autonomous racing car with a neural network controller. In *Proceedings of the 23rd ACM Intern. Conf. Hybrid Systems: Computation and Control*, 2020.
33. Kleinberg, J., Mullainathan, S. Raghavan, M. Inherent trade-offs in the fair determination of risk scores. In *Proceedings of Innovations in Theoretical Computer Science*, 2017.
34. Koh, N., Li, Y., Li, Y., Xia, L., Beringer, L., Honore, W., Mansky, W., Pierce, B.C. and Zdancewic, S. From C to interaction trees: Specifying, verifying, and testing a networked server. In *Proceedings of the 8th ACM SIGPLAN Intern. Conf. Certified Programs and Proofs*, Jan. 2019.
35. Kwiatkowska, M., Norman, G. and Parker, D. PRISM: Probabilistic Symbolic Model Checker. In *Proceedings of the PAPM/PROBMIV'01 Tools Session.* Sept. 2001, 7–12. Available as Technical Report 760/2001, University of Dortmund.
36. Lamport, L. Proving the correctness of multiprocess programs. *IEEE Trans. Software Engineering* SE-3, 2 (Mar. 1977), 125–143; doi: 10.1109/TSE.1977.229904.
37. Mandal, D., Deng, S., Hsu, D., Jana, S. and Wing, J.M. Ensuring fairness beyond the training data. In *Proceedings of the 34th Conf. Neural Information Processing Systems*, 2020.
38. Manna, Z. and Pneuli, A. Verification of concurrent programs: Temporal proof Principles. *Workshop on Logic of Programs.* Springer-Verlag, 1981, 200–252.
39. Microsoft Azure blog. Fairness in machine learning models, 2020; https://docs.microsoft.com/en-us/azure/machine-learning/concept-fairness-ml
40. Narayanan, A. 21 Definitions of fairness and their politics. In *Proceedings of FAT* 2018.* Tutorial; https://www.youtube.com/watch?v=jIXIuYdnyyk.
41. National Research Council. *Trust in Cyberspace.* The National Academies Press, 1999; https://doi.org/10.17226/6161
42. National Science Foundation. National AI Institutes Call for Proposals, 2019; https://www.nsf.gov/pubs/2020/nsf20503/nsf20503.htm.
43. National Science Foundation. NSF Program on Fairness in Artificial Intelligence in Collaboration with Amazon (FAI), 2020; https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505651
44. Newcombe, C., Rath, T., Zhang, F., Munteanu, B., Brooker, M. and Deardeuff, M. How Amazon Web Services uses formal methods. *Commun. ACM 58*, 4 (Apr. 2015), 66–73.
45. Networking and Information Technology Research and Development Subcommittee, Machine Learning and Artificial Intelligence Subcommittee, and the Special Cyber Operations Research and Engineering Subcommittee of the National Science and Technology Council. Artificial Intelligence and Cybersecurity: Opportunities and Challenges. Public Report; https://www.nitrd.gov/pubs/AI-CS-Tech-Summary-2020.pdf.
46. Platzer, A. *Logical Foundations of Cyber-Physical Systems.* Springer, Cham, 2018.
47. Pnueli, P. The temporal logic of programs. In *Proceedings of the Symp. Foundations of Computer Science*, 1977, 46–57.
48. Pong. F. and Dubois, M. Verification techniques for cache coherence protocols. *ACM Computing Surveys 29*, 1 (Mar. 1997).
49. Protzenko, J. et al. Verified low-level programming embedded in F*. In *Proceedings of 22nd Intern. Conf. Functional Programming*, May 2017.
50. Queille, J.P. and Sifakis, J. Specification and verification of concurrent systems in CESAR. In *Proceedings of the Intern. Symp. Programming*, LNCS 137, 1982, 337–351.
51. Seshia, S.A. et al. Formal specification for deep neural networks. In *Proceedings of the Intern. Symp. Automated Technology for Verification and Analysis*, LNCS 11138, Sept. 2018.
52. Shankland, S. Facebook starts building AI with an ethical compass. CNET, 2018; https://www.cnet.com/news/facebook-starts-building-ai-with-an-ethical-compass/.
53. Thomas, P.S., da Silva, B.C., Barto, A.G., Giguere, S., Brun, Y. and Brunskill, E. Preventing undesirable behavior of intelligent machines. *Science 366*, 6468 (2019), 999–1004.
54. Tiwari, P. et al. Computer-extracted texture features to distinguish cerebral radionecrosis from recurrent brain tumors on multiparametric MRI: A feasibility study. *American J. Neuroradiology 37*, 12 (2016), 2231–2236.
55. Varshney, K. Introducing AI Fairness 360. IBM, 2018; https://www.ibm.com/blogs/research/2018/09/ai-fairness-360/.
56. Wang, S., Pei, K., Whitehouse, J., Yang, J. and Jana, S. Formal security analysis of neural networks using symbolic intervals. In *Proceedings of the 27th USENIX Security Symp.*, 2018.
57. Wang, S., Pei, K., Whitehouse, J., Yang, J. and Jana, S. Efficient formal safety analysis of neural networks. In *Proceedings of Neural Information Processing Systems*, 2018.
58. White House. Executive Order on Promoting the Use of Trustworthy Artificial Intelligence in the Federal Government, Dec. 3, 2020; https://www.whitehouse.gov/presidential-actions/executive-order-promoting-use-trustworthy-artificial-intelligence-federal-government/.
59. Younes, H.L.S. and Simmons, R.G. Probabilistic verification of discrete event systems using acceptance sampling. In *Proceedings of the 14th Intern. Conf. Computer Aided Verification*, (Copenhagen, Denmark, July 2002), E. Brinksma and K. Guldstrand Larsen, Eds. Lecture Notes in Computer Science 2404, 223–235.
60. Zheng, D. and Kifer, D. Light DP: Towards automating differential privacy proofs. In *Proceedings of the 44th ACM SIGPLAN Symp. Principles of Programming Languages*, 2017, 888–901.

Jeannette M. Wing (wing@columbia.edu) is Avanessians Director of the Data Science Institute and Professor of Computer Science at Columbia University, New York, NY, USA.

Watch the author discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/trustworthy-ai