DOI:10.1145/3503914

Making AI more trustworthy with a formal methods-based approach to AI system verification and validation.

BY SANJIT A. SESHIA, DORSA SADIGH, AND S. SHANKAR SASTRY

Toward Verified Artificial Intelligence

ARTIFICIAL INTELLIGENCE (AI) is a term used for computational systems that attempt to mimic aspects of human intelligence, including functions we intuitively associate with intelligence, such as learning, problem solving, and thinking and acting rationally—for example, see Russell and Norvig.[26] We interpret the term AI broadly to include closely related areas such as machine learning (ML). Systems that heavily use AI, henceforth referred to as AI systems, have had a significant societal impact in domains that include healthcare, transportation, finance, social networking, e-commerce, and education.

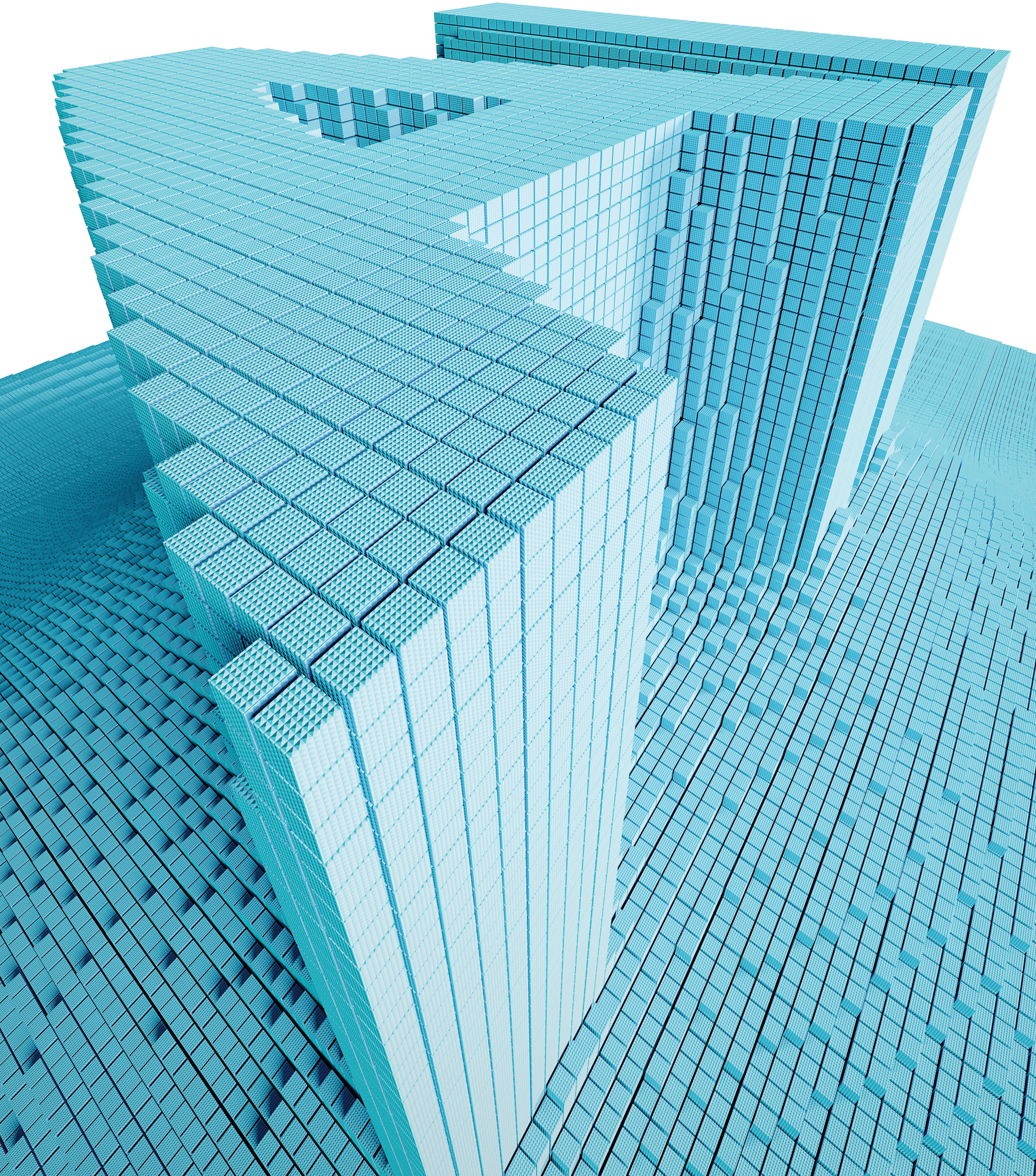
This growing societal-scale impact has brought with it a set of risks and concerns, including errors in AI software, cyber-attacks, and AI system safety.[4] Therefore, the question of verification and validation of AI systems, and, more broadly, of achieving trustworthy AI,[39] has begun to demand the attention of the research community. We define "verified AI" as the goal of designing AI systems that have strong, ideally provable, assurances of correctness with respect to mathematically specified requirements. How can we achieve this goal?

In this article, we consider the challenge of verified AI from the perspective of formal methods, a field of computer science and engineering concerned with the rigorous mathematical specification, design, and verification of systems.[38] At its core, formal methods is about proof: formulating specifications that form proof obligations; designing systems to meet those obligations; and verifying, via algorithmic proof search, that the systems indeed meet their specifications. A spectrum of formal methods, from specification-driven testing and simulation to model checking and theorem proving, are routinely used in the computer-aided design of integrated circuits (ICs) and have been widely applied to find bugs in software, analyze cyber-physical systems (CPS), and find security vulnerabilities. We review the way for-

» key insights

- **Formal methods promises to be an important enabler for trustworthy AI, but to fully realize this promise, the state of the art in formal methods must be advanced along multiple dimensions.**
- **The five main challenges include: developing languages and algorithms for environment and data modeling, abstractions and representations for complex ML components and systems, new specification formalisms and properties for AI systems and data, scalable computational engines for automated reasoning, and algorithmic techniques for trustworthy-by-construction design.**
- **This article proposes three principles to address each challenge, and surveys the progress toward and opportunities for achieving verified AI.**

ILLUSTRATION BY PETER CROWTHER ASSOCIATES



mal methods has traditionally been applied, identify the unique challenges arising in AI systems, and present ideas and recent advances towards overcoming these challenges.

This article seeks to address more than just specific types of AI components, such as deep neural networks (DNNs), or specific methods, such as reinforcement learning (RL). It attempts to cover the broad range of AI systems and their design processes. Additionally, recognizing that formal methods provide but one approach to trustworthy AI, our perspective is meant to complement those from other areas. Our views are largely shaped by problems arising from the use of AI in autonomous and semiautonomous systems, where safety and correctness concerns are more acute, though we believe the ideas presented here apply more broadly. This article is written for formal methods researchers and practitioners as well as for the broader computer science community. For the former, we present our viewpoint on where the real problems lie and how formal methods can have the greatest impact. For the latter, we sketch out our vision of how formal methods can be a key enabler for trustworthy AI.

We begin with a brief background of formal verification, an illustrative example, and a summary of the article's key ideas. We then outline five challenges to verified AI, discussing recent progress and presenting principles to address them.[a]

Overview

Figure 1 shows the typical processes for formal verification, formal synthesis, and formally guided runtime resilience. Consider the formal verification process, which begins with three inputs:

1. A model of the system to be verified, S.
2. A model of the environment, E.
3. The property to be verified, Φ.

Figure 1. Formal methods for verification, synthesis, and runtime resilience.

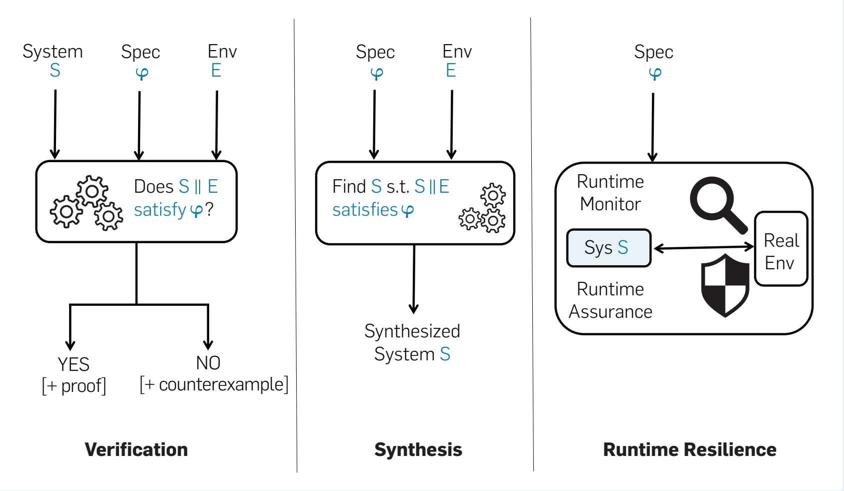
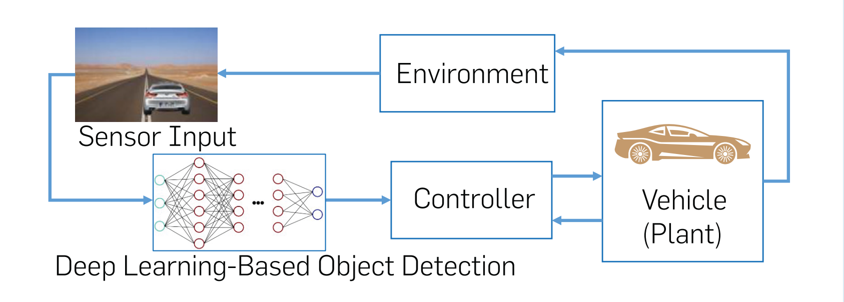


Figure 2. Example of a closed-loop cyber-physical system with machine-learning components (introduced in Dreossi et al.[5]).



The verifier generates a "yes/no" answer as output, indicating whether S satisfies the property Φ in environment E. Typically, a "no" output is accompanied by a counterexample, also called an *error trace*, which is an execution of the system that indicates how Φ is falsified. Some verification tools also include a proof or certificate of correctness with a "yes" answer. We take a broad view of formal methods to include any technique that uses some aspect of formal specification, verification, or synthesis. For instance, we include simulation-based hardware verification methods or model-based testing methods for software since they use formal specifications or models to guide the process of simulation or testing.

To apply formal verification to AI systems, one must be able to represent, at a minimum, the three inputs S, E, and Φ in formalisms for which (ideally) there exist efficient decision procedures to answer the "yes/no" question described previously. However, as will be shown, even constructing good representations of the three inputs is not straightforward, let alone dealing with the complexity of underlying design and verification problems.

We will illustrate the ideas in this article with examples from the domain of semiautonomous driving. Figure 2 shows an illustrative example of an AI system: a closed-loop CPS comprising a semiautonomous vehicle, with ML components, along with its environment. Specifically, assume the semiautonomous "ego" vehicle has an automated emergency braking system (AEBS) that attempts to detect and classify objects in front of it and actuate the brakes when needed to avert a collision. Figure 2 shows the AEBS as a system composed of a controller (automatic braking), a plant (vehicle sub-system under control, including other parts of the autonomy stack),

[a] The first version of this article was published on arXiv in July 2016 in response to a call for white papers for the 2016 CMU Exploratory Workshop on Safety and Control for AI. Two revisions have been completed since. This latest version reflects the evolution of the authors' perspective on verified AI. Since 2016, literature on the topic has grown substantially; however, per *Communications* guidelines we are limited to 40 references, so a comprehensive survey of the topic is out of scope.

and a sensor (camera), along with a perception component implemented using a DNN. The AEBS, when combined with the vehicle's environment, forms a closed-loop CPS. The environment of the ego vehicle comprises both agents and objects outside the vehicle (other vehicles, pedestrians, among others) as well as inside the vehicle—for instance, a driver. A safety requirement for this closed-loop system can be informally characterized as the property of maintaining a safe distance between the moving ego vehicle and any other agent or object on the road. However, there are many nuances to the specification, modeling, and verification of such a system.

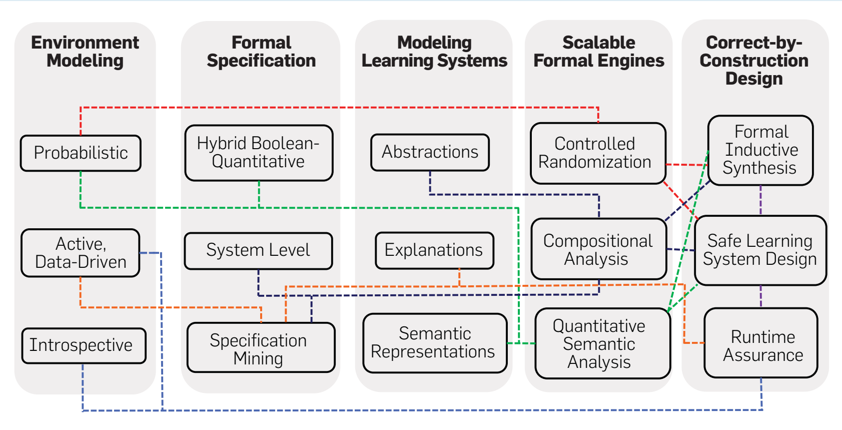
First, consider modeling the environment of a semiautonomous vehicle, where there can be considerable uncertainty even about how many and which agents are in the environment (human and nonhuman), let alone about their attributes and behaviors. Second, the perceptual tasks which use AI/ML can be hard, if not impossible, to formally specify. Third, components such as DNNs can be complex, high-dimensional objects that operate on complex, high-dimensional input spaces. Thus, even generating the three inputs S, E, Φ to the formal verification process in a form that makes verification tractable is challenging.

Assuming one solves that problem, then one is presented with the daunting task of verifying a complex AI-based CPS like the one in Figure 2, where a compositional (modular) approach is essential for scalability and yet difficult to implement—for example due to the difficulty of compositional specification. Finally, correct-by-construction design methods hold promise for achieving verified AI, but they are in their infancy and crucially rely on advances in specification and verification. Figure 3 summarizes the five challenge areas for verified AI. For each area, we have distilled the current promising approaches into three principles for overcoming that challenge, depicted as nodes. Edges between nodes show which principles for verified AI depend on each other, with a common thread of dependencies denoted by a single color; we will recapitulate these dependencies in the "Conclusion." The rest of the article elaborates on these challenges and corresponding principles.

Environment Modeling

The environments in which AI/ML-based systems operate are generally complex. For example, consider modeling the variety of urban traffic environments where an autonomous car must operate. Indeed, AI/ML is often introduced into these systems precisely to deal with such complexity and uncertainty. Current ML design flows usually specify environments implicitly, with data. Many AI systems are designed to discover and make sense of their environment during their operation, as opposed to traditional systems designed for an environment specified a priori. All formal verification and synthesis, however, is with respect to an environment model. Hence, assumptions about and properties of input data must be explicated into environment models. We distill this dichotomy into three challenges of environment modeling for AI systems and develop corresponding principles to address them.

Figure 3. Summary of the five challenge areas for verified AI, the 15 corresponding principles proposed to address them, and the connections and dependencies between them (the same color indicates a common dependency thread).



Modeling uncertainty. In traditional uses of formal verification, it is commonplace to model the environment as a nondeterministic process or "disturbance" that is subject to constraints. Such "overapproximated" environment modeling permits one to conservatively capture uncertainty about the environment without an overly detailed model that can be inefficient to reason about. However, for AI-based autonomy, purely nondeterministic modeling is likely to produce too many spurious bug reports, rendering the verification process useless in practice. As an example, consider modeling the behavior of environment vehicles for an autonomous car, where the range of possible behaviors is so broad that an accident might always occur if purely nondeterministic modeling is employed. Moreover, many AI/ML systems implicitly or explicitly make distributional assumptions on data from or behaviors of the environment, creating a need for probabilistic modeling. Since it can be difficult to exactly ascertain the underlying distributions, the resulting probabilistic model cannot be assumed to be "perfect," and uncertainty in the modeling process must be represented in the model itself.

Probabilistic formal modeling. To tackle this challenge, we suggest using formalisms that combine probabilistic and nondeterministic modeling. Where probability distributions can be reliably specified or estimated, one can use probabilistic modeling. Elsewhere, nondeterministic modeling can be used to overapproximate environment behaviors. While formalisms such as Markov Decision Processes (MDPs) already provide a way to blend probability and nondeterminism, we believe that richer formalisms, such as the paradigm of probabilistic programming (for example, Fremont et al.[10] and Milch et al.[20]), can provide an expressive and programmatic way to model environments. We expect that in many cases, such probabilistic programs will need to be learned/synthesized (in part) from data. In this case, any uncertainty in learned parameters must be propagated to the rest of the system and represented in the probabilistic model. For example, convex MDPs[25] provide a way of representing uncertainty in the values of learned transition probabilities, with algorithms for verification and control extended to account for this uncertainty.

Unknown variables. In the traditional domains for formal verification, such as verifying device drivers, the interface between the system S and its environment E is well defined, and E can only interact with S through this interface. For AI-based autonomy, such as the example in the "Overview," the interface is imperfect, specified by sensors and perception components that only partially and noisily capture the environment, and it does not capture all the interactions between S and E. Not all the environment's variables (features) are known, let alone sensed. Even in restricted scenarios where environment variables are known, there is a striking lack of information, especially at design time, about their evolution. Additionally, modeling sensors such as LiDAR, which represent the interface to the environment, is a major technical challenge.

Introspective environment modeling. We suggest to address this problem by developing design and verification methods that are introspective—that is, they introspect on the system S itself to algorithmically identify assumptions A about the environment E that are sufficient to guarantee the satisfaction of the specification Φ.[31] Ideally, A must be the weakest of such assumptions and must also be efficient enough to generate at design time and monitor at runtime over available sensors and other sources of information about the environment so that mitigating actions can be taken when they are violated. Moreover, if a human operator is involved, one might want A to be translatable into an explanation that is understandable, so that S can "explain" to the human why it may not be able to satisfy the specification Φ. Dealing with these multiple requirements, as well as the need for good sensor models, makes introspective environment modeling a highly nontrivial problem to solve.[31] Preliminary work has shown that such extraction of monitorable assumptions is feasible in simple cases,[17] although more work is required to make this practical.

We need techniques to model ML components along with their context so that semantically meaningful properties can be verified.

Modeling human behavior. For many AI systems, such as semiautonomous vehicles, human agents are a key part of the environment and/or system. Hand-crafted models of humans do not adequately capture the variability and uncertainty of human behavior. On the other hand, data-driven approaches for modeling human behavior can be sensitive to the expressivity of the features used by the ML model and the quality of data. To achieve high assurance for human-AI systems, we must address the limitations of current human modeling techniques and provide guarantees about their prediction accuracy and convergence.

Active data-driven modeling. We believe human modeling requires an active data-driven approach, with the model structure and features expressed in mathematical formalisms amenable to the use of formal methods. A critical aspect of human modeling is to capture human intent. We propose a three-pronged approach: define model templates/features based on expert knowledge, use offline learning to complete the model for design-time use, and learn and update environment models at runtime by monitoring and interacting with the environment. For instance, it has been shown that data gathered from driving simulators via human subject experiments can be used to generate models of human driver behavior that are useful for verification and control of autonomous vehicles.[27,28] In addition, adversarial training and attack techniques from computer security[13] can be used in active learning of human models and can further be devised to target specific human actions that lead to unsafe behaviors. These techniques can help develop verification algorithms for human-AI systems.

Formal Specification

Formal verification critically relies on having a formal specification—a precise, mathematical statement of what the system is supposed to do. Coming up with a high-quality formal specification is challenging even for domains in which formal methods have found considerable success, but there are unique aspects of this challenge for AI systems as we elaborate below.

Hard-to-formalize tasks. Consider the perception module in Figure 2's AEBS controller, which must detect and classify objects to distinguish vehicles and pedestrians from other entities. Accuracy for this module, in the classic formal methods sense, requires a formal definition of each type of road user and object, which is extremely difficult, if not impossible. This problem exists for any implementation of this perception module, not just approaches based on deep learning. Similar problems arise for other tasks involving perception and communication, such as natural language processing. How, then, do we specify accuracy properties for such a module? What should the specification language be and what tools can one use to construct a specification?

End-to-end/system-level specifications. To address the above challenge, we change the problem slightly. Rather than directly attempting a specification of a hard-to-formalize task, focus first on precisely specifying the end-to-end behavior of the AI system. From this "system-level" specification, one can derive constraints on the input-output interface of the hard-to-formalize component. These constraints serve as a component-level specification that is relevant in the context of the overall AI system. For our AEBS example (Figure 2), this involves specifying the property Φ corresponding to maintaining a minimum distance from any object during motion, from which we derive constraints on the input space of the DNN capturing a semantically meaningful input space for adversarial analysis (see Dreossi et al.[5]).

Quantitative vs. Boolean specifications. Traditionally, formal specifications tend to be Boolean, mapping a given system behavior to "true" or "false." However, in AI and ML, specifications are often given as objective functions specifying costs or rewards. Moreover, there can be multiple objectives, some of which must be satisfied together and others that may need to be traded off against each other in certain environments. What are the best ways to unify Boolean and quantitative approaches to specification? Are there formalisms that can capture commonly discussed properties of AI components, such as robustness and fairness, in a unified manner?

Hybrid quantitative-Boolean specifications. Boolean and quantitative specifications both have their advantages: Boolean specifications are easier to compose, however objective functions lend themselves to optimization-based techniques for verification and synthesis, and to defining finer granularities of property satisfaction. One way to bridge this gap is to move to quantitative specification languages, such as using logics with both Boolean and quantitative semantics (for example, metric temporal logic[22]) or combining automata with reward functions for RL.[15] Another approach is to combine Boolean and quantitative specifications into a common specification structure, such as a rulebook,[1] where specifications can be organized in a hierarchy, compared, and aggregated. Wing[39] has identified several categories of properties for AI systems, including robustness, fairness, privacy, accountability, and transparency. Novel formalisms, which bridge ideas from formal methods and ML, are being developed to model the variants of these properties including, for instance, notions of semantic robustness.[32]

Data vs. formal requirements. The view of "data as specification" is common in machine learning. Labeled "ground-truth" data over a finite input set is often the only specification of correct behavior. This is very different from formal methods, where a specification, typically given in logic or as automata, defines a set of correct behaviors over all possible inputs. This gap can be problematic, especially when the data is limited, biased, or from non-experts. We need techniques to formalize properties of data, including data available at design time and data that has yet to be encountered.

Specification mining. To bridge this gap between data and formal specification, we suggest using algorithms to infer specifications from data and other observations—so-called specification mining techniques. Such methods could be used for ML components in general, including for perception components, since in many cases it is not required to have an exact specification or one that is human-readable. Specification mining methods could also be used to infer human intent and other properties from demonstrations[37] or more complex forms of interaction between multiple agents, both human and AI.

Modeling Learning Systems

In most traditional applications of formal verification, the system S is fixed and known at design time—for example, it is a program, or it is a circuit described in a programming language or hardware-description language. The system-modeling problem is primarily concerned with reducing the size of S to a more tractable one by abstracting away irrelevant details. AI systems lead to a very different challenge for system modeling, primarily stemming from the use of machine learning:

- **High-dimensional input space.** ML components used for perception usually operate over very high-dimensional input spaces. For the illustrative example of in Dreossi et al.,[5] each input RGB image is 1000 x 600 pixels, contains $256^{1000\times600\times3}$ elements, and in general the input is a stream of such high-dimensional vectors. Although researchers have used formal methods for high-dimensional input spaces (for example, in digital circuits), the nature of the input spaces for ML-based perception is different—not entirely Boolean, but hybrid, including both discrete and continuous variables.
- **High-dimensional parameter/state space.** ML components such as deep neural networks have anywhere from thousands to millions of model parameters and primitive components. For example, state-of-the-art DNNs used by the authors in instantiations of Figure 2 have up to 60 million parameters and tens of layers. This gives rise to a huge search space for verification that requires careful abstraction.
- **Online adaptation and evolution.** Some learning systems, such as a robot using RL, evolve as they encounter new data and situations. For such systems, design-time verification must either account for future changes in the behavior of the system or be performed incrementally and online as the learning system evolves.
- **Modeling systems in context.** For many AI/ML components, their specification is only defined by the context.

For example, verifying the safety of Figure 2's DNN-based system requires a model of its environment. We need techniques to model ML components along with their context so that semantically meaningful properties can be verified.

In recent years, much work has focused on improving the efficiency with which tools can verify robustness and input-output properties of DNNs (see Liu et al.[18] for a recent survey). However, this is not enough. Advances are needed in three areas:

Automated abstraction and efficient representations. Techniques for automatically generating abstractions of systems have been the linchpins of formal methods, playing crucial roles in extending the reach of formal methods to large hardware and software systems. To address the challenges of very high-dimensional hybrid-state spaces and input spaces for ML-based systems, we need to develop effective techniques to abstract ML models into simpler models that are more amenable to formal analysis. Some promising directions include using abstract interpretation to analyze DNNs (for example, Gehr et al.[12]), developing abstractions for falsifying cyber-physical systems with ML components,[5] and devising novel representations for verification (for instance, star sets and other examples[36]).

Explanations and causality. The task of modeling a learning system can be simplified if the learner accompanies its predictions with explanations of how those predictions result from the data and background knowledge. While this idea is not new—it has been investigated by the ML community under terms such as explanation-based generalization[21]—there has been a recent renewal of interest in using logic to explain the output of learning systems (for example, Jha et al.[16]). Explanation generation can aid in debugging both designs and specifications at design time and in synthesizing robust AI systems for runtime assurance. ML that incorporates causal and counterfactual reasoning[24] can also help to generate explanations for use with formal methods.

Semantic feature spaces. The adversarial analysis[13] and formal verification of ML models is more meaningful when the generated adversarial inputs and counterexamples make semantic sense in the context in which the ML models are used. For example, techniques that analyze a DNN object detector against small changes in the color of cars or time of day are arguably more useful that those that add noise to a small number of arbitrarily chosen pixels. Most current methods (for example, Goodfellow et al.[13] or Liu et al.[18]) fall short on this count. We need semantic adversarial analysis[7] in which ML models are analyzed within the context of the systems they are part of. A key step is to represent the semantic feature space modeling the environment in which the ML system operates, as opposed to the concrete feature space which defines the input space for the ML model. This follows the intuition that the semantically meaningful part of the concrete feature space (for instance, traffic scene images) form a much lower dimensional latent space compared to the full concrete feature space. Figure 2's semantic feature space is the lower-dimensional space representing the 3D world around the autonomous vehicle, whereas the concrete feature space is the high-dimensional pixel space. Since the semantic feature space is lower dimensional, it can be easier to search over (see, for example, Dreossi et al.[5] or Huang et al.[14]). However, one needs a "renderer" that maps a point in the semantic feature space to one in the concrete feature space. The renderer's properties, such as differentiability, can make it easier to apply formal methods to perform goal-directed search of the semantic feature space.

Computational Engines for Design and Verification

The effectiveness of formal methods for hardware and software systems has been driven by advances in underlying "computational engines"—for instance, Boolean satisfiability solving (SAT), satisfiability modulo theories (SMT), and model checking. Given the scale of AI/ML systems, the complexity of their environments, and the new types of specifications involved, a new class of computational engines is needed for efficient and scalable training, testing, design, and verification. We identify the key challenges that must be overcome to achieve these advances.

Dataset design. Data is the fundamental starting point for machine learning. Any quest to improve the quality of an ML system must improve the quality of the data it learns from. How can formal methods help to systematically select, design, and augment the data used for ML?

Data generation for ML shares similarities with the problem of test generation for hardware and software. Formal methods approaches have proved effective for systematic, constraint-based test generation. However, the requirements for AI systems are different. The types of constraints can be much more complex—for example, encoding requirements on the "realism" of data captured using sensors from a complex environment, such as a traffic situation. We need to generate not just data items with specific characteristics (such as tests that uncover bugs), but an ensemble that satisfies distributional constraints. Data generation must also meet objectives on dataset size and diversity for effective training and generalization. These requirements necessitate the development of a new suite of formal techniques.

Controlled randomization in formal methods. This problem of dataset design has many facets. First, one must define the space of "legal" inputs so that the examples are well formed according to the application semantics. Secondly, constraints capturing a measure of similarity with real-world data are needed. Third, constraints are typically required on the distribution of the generated examples to obtain guarantees about convergence of the learning algorithm to the true concept.

We believe these facets can be addressed by randomized formal methods—randomized algorithms for generating data that are subject to formal constraints and distribution requirements. A new class of techniques, termed *control improvisation*,[9] holds promise. An improviser is a generator of random strings (examples) x that satisfy three constraints:

- A hard constraint that defines the space of legal x
- A soft constraint defining how the generated x must be similar to real-world examples

- A randomness requirement defining a constraint on the output distribution.

Control improvisation theory is still in its infancy, and we are just starting to understand the computational complexity and to devise efficient algorithms. Improvisation, in turn, relies on recent progress on computational problems such as constrained random sampling and model counting (for instance, Meel et al.[19]) and generative approaches based on probabilistic programming (for instance, Fremont et al.[10]).

Quantitative verification. In addition to the scale of AI systems as measured by traditional metrics (dimension of state space, number of components, and so on), the types of components can be much more complex. For instance, autonomous and semiautonomous vehicles and their controllers must be modeled as hybrid systems, combining both discrete and continuous dynamics. Moreover, agents in the environment (humans, other vehicles) may need to be modeled as probabilistic processes. Finally, the requirements may involve not only traditional Boolean specifications on safety and liveness, but also quantitative requirements on system robustness and performance. Yet, most of the existing verification methods are targeted toward answering Boolean verification questions. To address this gap, new scalable engines for quantitative verification must be developed.

Quantitative semantic analysis. The complexity and heterogeneity of AI systems means that, in general, formal verification of specifications (Boolean or quantitative) is undecidable—for example, even deciding whether a state of a linear hybrid system is reachable is undecidable. To overcome this obstacle posed by computational complexity, one must augment the abstraction and modeling methods discussed earlier in this section with novel techniques for probabilistic and quantitative verification over the semantic feature space. For specification formalisms that have both Boolean and quantitative semantics, in formalisms such as metric temporal logic, the formulation of verification as optimization is crucial to unifying computational methods from formal methods with those from the optimization literature, such as in simulation-based temporal logic falsification (for example, Nghiem et al.[22]), although they must be applied to the semantic feature space for efficiency.[6] Such falsification techniques can also be used for the systematic, adversarial generation of training data for ML components.[6] Techniques for probabilistic verification should be extended beyond traditional formalisms, such as Markov chains or MDPs, to verify probabilistic programs over semantic feature spaces. Similarly, work on SMT solving must be extended to handle cost constraints more effectively—in other words, combining SMT solving with optimization methods (for example, Shoukry et al.[34]).

> We need to develop an understanding of what can be guaranteed at design time, how the design process can contribute to safe operation at runtime, and how design-time and runtime techniques can interoperate effectively.

Compositional reasoning for AI/ML. For formal methods to scale to large systems, compositional (modular) reasoning is essential. In compositional verification, a large system (for example, a program) is split into its components (for example, procedures), each component is verified against a specification, and then the component specifications together entail the system-level specification. A common approach for compositional verification is the use of *assume-guarantee contracts*. For example, a procedure assumes something about its starting state (pre-condition) and in turn guarantees something about its ending state (post-condition). Similar assume-guarantee paradigms have been developed for concurrent software and hardware systems. However, these paradigms do not cover AI systems, in large part due to the challenges in specifying AI systems as discussed in the "Formal Specification" section. Compositional verification requires compositional specification—that is, the components must be formally specifiable. However, as noted in "Formal Specification," it may be impossible to formally specify the correct behavior of a perception component. One of the challenges, then, is to develop techniques for compositional reasoning that do not rely on having complete compositional specifications. Additionally, the quantitative and probabilistic nature of AI systems requires extending the theory of compositional reasoning to quantitative systems and specifications.

Inferring component contracts. Compositional design and analysis of AI systems needs progress on multiple fronts. First, theories of probabilistic assume-guarantee design and verification need to be developed for the semantic spaces of such systems, building on some promising initial work (for example, Nuzzo et al.[23]). Second, new techniques for inductive synthesis[30] must be devised to generate assume-guarantee contracts algorithmically to reduce the specification burden and facilitate compositional reasoning. Third, to handle the case of components, such as perception, that do not have precise formal specifications, we suggest techniques that infer component-level constraints from system-level analysis (for example, Dreossi et al.[5]) and use such constraints to focus component-level analysis, including adversarial analysis, on searching the "relevant" part of the input space.

Correct-by-Construction Intelligent Systems

In an ideal world, verification would be integrated with the design process, so the system is "correct-by-construction." For example, verification can be interleaved with compilation/synthesis steps, such as in the register-transfer-level (RTL) design flow common in ICs, or it can be integrated into synthesis algorithms to ensure the implementation satisfies the specification. Can we devise a suitable correct-by-construction design flow for AI systems?

Specification-driven design of ML components. Given a formal specification, can we design a machine-learning component (model) that provably satisfies that specification? This clean-slate ML component design has many aspects: (1) design the dataset, (2) synthesize the structure of the model, (3) generate a representative set of features, (4) synthesize hyperparameters and other aspects of ML algorithm selection, and (5) automate techniques for debugging ML models or the specification when synthesis fails.

Formal synthesis of ML components. Solutions are emerging that address some of the aspects listed previously. Properties can be enforced on ML models using semantic loss functions (for example, Xu et al.[40]) or via certified robustness (for example, Cohen et al.[2]). These techniques can be combined with methods such as neural architecture search to produce correct-by-construction DNNs. Another approach is based on the emerging theory of formal inductive synthesis,[30] the synthesis from examples of programs that satisfy formal specifications. The most common approach to solving a formal inductive synthesis problem is to use an oracle-guided approach, in which a learner is paired with an oracle that answers queries. For the example in Figure 2, the oracle can be a falsifier that generates counterexamples showing how a failure of the learned component violates the system-level specification. Finally, the use of theorem proving to ensure correctness of algorithms used to train ML models (for example, Selsam et al.[29]) is also an important step toward correct-by-construction ML components.

Design of ML-based systems. A second challenge is to design an overall system comprising both learning and non-learning components. Several research questions arise. Can we compute safety envelopes within which ML components can be constrained to operate? Can we design a control or planning algorithm that can overcome the limitations of an ML-based perception component it receives input from? Can we devise theories of compositional design for AI systems? For example, if two ML models are used for perception on two different types of sensor data (for instance, LiDAR and visual images), and each satisfies its specifications under certain assumptions, under what conditions can they be used together to improve the reliability of the overall system?

Safe learning. A prominent example of progress on this challenge is the work on safe learning-based control (for example, Fisac et al.[8]). In this approach, a safety envelope is pre-computed, and a learning algorithm is used to tune a controller within that envelope. Techniques are needed for efficiently computing such safety envelopes based on, for example, reachability analysis.[35] Similarly, the field of safe RL has seen remarkable progress (see Garcia and Fernández[11] for a survey). However, these do not yet fully address the challenges posed by ML for perception and prediction—for instance, provably safe, end-to-end, deep RL has yet to be achieved.

Bridging design time and runtime for resilient AI. Many AI systems operate in environments that are not specifiable a priori, as discussed in the "Environment Modeling" section, and therefore, there will always be environments in which we do not have a provable guarantee of correctness. Therefore, techniques for achieving fault tolerance and error resilience at runtime play an especially crucial role for AI systems. We need to develop a systematic understanding of what can be guaranteed at design time, how the design process can contribute to safe and correct operation of the AI system at runtime, and how the design-time and runtime techniques can interoperate effectively.

Runtime assurance. The literature on fault-tolerant and dependable systems offers us with a foundation to develop techniques for runtime assurance—that is, runtime verification and mitigation techniques. For example, the Simplex method[33] provides one approach to combining a complex but error-prone module with a safe, formally verified backup module. Recent techniques for combining design-time and runtime assurance methods (for example, Desai et al.[3]) have shown how unverified components, including those based on AI and ML, can be wrapped within a runtime assurance framework to provide guarantees of safe operation. However, these are currently limited to specific classes of systems, or they require manual design of runtime monitors and mitigation strategies. More work is needed on methods such as introspective environment modeling[31] and synthesis of monitors and safe fallback strategies for AI.

The correct-by-construction design methods discussed here may introduce overhead that makes it more difficult to meet performance and real-time requirements. However, we believe that (perhaps nonintuitively) formal methods can even help to improve a system's performance or energy efficiency, in the following sense. Conventional performance tuning tends to be context-independent—for instance, tasks need to meet deadlines independent of the environment in which they operate. However, there may be environments where an ML model could trade off accuracy for higher efficiency if such environments are formally characterized at design time and monitored at runtime, and if system operation in them is formally verified to be safe. This trade-off can be a fruitful area for future research.

Conclusion

Taking a formal methods perspective, we have dissected the problem of designing high-assurance AI systems. As summarized in Figure 3, we identified five main challenges for applying formal methods to AI systems. For each of these five challenges, we have formulated three design and verification principles which hold promise for addressing that challenge. The edges in Figure 3 show the dependencies between these principles. For example, runtime assurance relies on introspective and data-driven environment modeling to extract monitorable assumptions and environment models. Similarly, to perform system-level analysis, we require compositional reasoning and abstraction, where some AI components may require specifications to be mined, while others are generated correct-by-construction via formal inductive synthesis.

Several researchers, including the authors, have been working on addressing these challenges since 2016, when the original version of this article was published; a few sample advances are described. We have developed open-source tools, VerifAI[6] and Scenic,[10] which implement techniques based on the principles described in this article and have been applied to industrial-scale systems in the autonomous driving and aerospace domains. These results are but a start and much more remains to be done. Verified AI promises to continue to be a fruitful area for research in the years to come.

Acknowledgments

Our work has been supported in part by the National Science Foundation (NSF), the Defense Advanced Research Projects Agency (DARPA), the Semiconductor Research Corporation (SRC), and several industry sponsors. We gratefully acknowledge the many people with whom our conversations and collaborations have helped shape this article.

References

1. Censi, A., Slutsky, K., Wongpiromsarn, T., Yershov, D., Pendleton, S., Fu, J., and Frazzoli, E. Liability, ethics, and culture-aware behavior specification using rulebooks. In *2019 Intern. Conf. on Robotics and Automation (ICRA)*, IEEE, 8536–8542.
2. Cohen, J.M., Rosenfeld, E., and Kolter, J.Z. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th Intern. Conf. on Machine Learning 97* (2019), 1310–1320.
3. Desai, A., Ghosh, S., Seshia, S.A., Shankar, N., and Tiwari, A. SOTER: A runtime assurance framework for programming safe robotics systems. In *IEEE/IFIP Intern. Conf. on Dependable Systems and Networks* (2019), 138–150.
4. Dietterich, T.G. and Horvitz, E.J. Rise of concerns about AI: Reflections and directions. *Communications of the ACM 58*, 10 (2015), 38–40.
5. Dreossi, T., Donze, A., and Seshia, S.A. Compositional falsification of cyber-physical systems with machine learning components. In *Proceedings of the NASA Formal Methods Conf. (Lecture Notes in Computer Science) 10227* (2017), 357–372.
6. Dreossi, T., Fremont, D.J., Ghosh, S., Kim, E., Ravanbakhsh, H., Vazquez-Chanlatte, M., and Seshia, S.A. VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *31st Intern. Conf. on Computer Aided Verification (Lecture Notes in Computer Science) 11561* (2017), 432–442.
7. Dreossi, T., Jha, S., and Seshia, S.A. Semantic adversarial deep learning. In *30th Intern. Conf. on Computer Aided Verification (Lecture Notes in Computer Science) 10981* (2018), 3–26.
8. Fisac, J.F. et al. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control 64*, 7 (2018), 2737–2752.
9. Fremont, D.J., Donzé, A., Seshia, S.A., and Wessel, D. Control improvisation. In *35th IARCS Annual Conf. on Foundations of Software Technology and Theoretical Computer Science* (2015), 463–474.
10. Fremont, D.J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A.L., and Seshia, S.A. Scenic: A language for scenario specification and scene generation. In *Proceedings of the 40th annual ACM SIGPLAN Conf. on Programming Language Design and Implementation* (2019).
11. Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *J. of Machine Learning Research 16*, 1 (2015), 1437–1480.
12. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. AI2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy* (2018), 3–18.
13. Goodfellow, I., McDaniel, P., and Papernot, N. Making machine learning robust against adversarial inputs. *Communications of the ACM 61*, 7 (2018), 56–66.
14. Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. In *Intern. Conf. on Computer Aided Verification*, Springer, (2017), 3–29.
15. Icarte, R.T., Klassen, T., Valenzano, R., and McIlraith, S. Using reward machines for high-level task specification and decomposition in reinforcement learning. *Intern. Conf. on Machine Learning* (2018), 2107–2116.
16. Jha, S., Sahai, T., Raman, V., Pinto, A., and Francis, M. Explaining AI decisions using efficient methods for learning sparse Boolean formulae. *J. of Automated Reasoning 63*, 4 (2019), 1055–1075.
17. Li, W., Sadigh, D., Sastry, S.S., and Seshia, S.A. Synthesis for human-in-the-loop control systems. In *Proceedings of the 20th Intern. Conf. on Tools and Algorithms for the Construction and Analysis of Systems* (2014), 470–484.
18. Liu, C. et al. Algorithms for verifying deep neural networks. *Foundations and Trends in Optimization 4*, 3–4 (2021), 244–404.
19. Meel, K.S. et al. Constrained sampling and counting: Universal hashing meets SAT solving. In *Beyond NP, Papers from the 2016 AAAI Workshop*, (Feb. 12, 2016).
20. Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D.L., and Kolobov, A. BLOG: Probabilistic models with unknown objects. *Statistical Relational Learning* (2007), 373.
21. Mitchell, T.M., Keller, R.M., and Kedar-Cabelli, S.T. Explanation-based generalization: A unifying view. *Machine Learning 1*, 1 (1986), 47–80.
22. Nghiem, T. et al. Monte-Carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proceedings of the 13th ACM Intern. Conf. on Hybrid Systems: Computation and Control* (2010), 211–220.
23. Nuzzo, P., Li, J., Sangiovanni-Vincentelli, A.L., Xi, Y., and Li, D. Stochastic assume-guarantee contracts for cyber-physical system design. *ACM Transactions on Embedded Computing Systems 18*, 1, (Jan. 2019).
24. Pearl, J. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM 62*, 3 (2019), 54–60.
25. Puggelli, A., Li, W., Sangiovanni-Vincentelli, A.L., and Seshia, S.A. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In *Proceedings of the 25th Intern. Conf. on Computer-Aided Verification* (2013).
26. Russell, S.J. and Norvig, P. *Artificial Intelligence: A Modern Approach.* Prentice Hall (2010).
27. Sadigh, D. et al. Data-driven probabilistic modeling and verification of human driver behavior. In *Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symposium* (2014).
28. Sadigh, D., Sastry, S., Seshia, S.A., and Dragan, A.D. Information gathering actions over human internal state. In *Proceedings of the IEEE/RSJ Intern. Conf. on Intelligent Robots and Systems* (2016).
29. Selsam, D., Liang, P., and Dill, D.L. Developing bug-free machine learning systems with formal mathematics. In *Proceedings of the 34th Intern. Conf. on Machine Learning 70* (2017), 3047–3056.
30. Seshia, S.A. Combining induction, deduction, and structure for verification and synthesis. In *Proceedings of the IEEE 103*, 11 (2015), 2036–2051.
31. Seshia, S.A. Introspective environment modeling. *19th Intern. Conf. on Runtime Verification* (2019), 15–26.
32. Seshia, S.A. et al. Formal specification for deep neural networks. In *Proceedings of the Intern. Symp. on Automated Technology for Verification and Analysis* (2018), 20–34.
33. Sha, L. Using simplicity to control complexity. *IEEE Software 18*, 4 (2001), 20–28.
34. Shoukry, Y., et al. SMC: Satisfiability modulo convex optimization. In *Proceedings of the 10th Intern. Conf. on Hybrid Systems: Computation and Control* (2017).
35. Tomlin, C., Mitchell, I., Bayen, A.M., and Oishi, M. Computational techniques for the verification of hybrid systems. In *Proceedings of the IEEE 91*, 7 (2003), 986–1001.
36. Tran, H-D., Xiang, W., and Johnson, T.T. Verification approaches for learning-enabled autonomous cyber-physical systems. *IEEE Design & Test* (Aug. 2020). https://doi.org/10.1109/MDAT.2020.3015712.
37. Vazquez-Chanlatte, M., Jha, S., Tiwari, A., Ho, M.K., and Seshia, S.A. Learning task Specifications from demonstrations. In *Advances in Neural Information Processing Systems 31* (2018), 5372–5382.
38. Wing, J.M. A specifier's introduction to formal methods. *IEEE Computer 23*, 9 (Sept. 1990), 8–24.
39. Wing, J.M. Trustworthy AI. *Communications of the ACM 64*, 10 (2021), 64–71.
40. Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Van den Broeck, G. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th Intern. Conf. on Machine Learning* (2018), 5498–5507.

Sanjit A. Seshia (sseshia@eecs.berkeley.edu) is a professor of Electrical Engineering and Computer Sciences at the University of California, Berkeley, CA, USA.

Dorsa Sadigh is an assistant professor of Computer Science and Electrical Engineering at Stanford University, CA, USA.

S. Shankar Sastry is the Thomas Siebel Professor of Computer Science and professor of Electrical Engineering and Computer Sciences, BioEngineering, and Mechanical Engineering at the University of California, Berkeley, CA, USA.

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