Towards Verified Artificial Intelligence

Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry

University of California, Berkeley
{sseshia,dsadigh,sastry}@eecs.berkeley.edu

July 1, 2016

Abstract

Verified artificial intelligence (AI) is the goal of designing AI-based systems that are provably correct with respect to mathematically-specified requirements. This paper considers Verified AI from a formal methods perspective. We describe five challenges for achieving Verified AI, and five corresponding principles for addressing these challenges.

1 Introduction

Artificial Intelligence (AI), as described by Russell and Norvig [46], is the study of general principles of rational agents and components for constructing these agents. More broadly, it involves building intelligent entities that mimic 'cognitive' functions we intuitively associate with human minds, such as 'learning' and 'problem solving.' We interpret the term AI broadly to include closely-related areas such as machine learning [35]. Systems that heavily use AI, henceforth referred to as *AI-based systems*, have had a significant impact in society in domains that include healthcare, transportation, social networking, e-commerce, education, etc. This growing societal-scale impact has brought with it a set of risks and concerns including errors in AI software, cyber-attacks, and safety of AI-based systems [45, 16, 3]. Therefore, the question of verification and validation of AI-based systems has begun to demand the attention of the research community. We define "*Verified AI*" as the goal of designing AI-based systems that are proven to satisfy desired properties specified in a mathematical formalism. How can we achieve this goal?

A natural starting point is to consider *formal methods* — a field of computer science and engineering concerned with the rigorous mathematical specification, design, and verification of systems [53, 14]. 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. Verification techniques such as model checking [12, 43, 13] and theorem proving (see, e.g. [41, 28, 25]) are used routinely in the computer-aided design of integrated circuits and have been widely applied to find bugs in software, analyze embedded systems, and find security vulnerabilities. At the heart of these advances are computational proof engines such as Boolean satisfiability (SAT) solvers [34], Boolean reasoning and manipulation routines based on Binary Decision Diagrams (BDDs) [8], and satisfiability modulo theories (SMT) solvers [5].

In this paper, we consider the challenge of Verified AI from a formal methods perspective. That is, we review the manner in which formal methods have traditionally been applied, analyze the challenges this approach may face for AI-based systems, and propose techniques to overcome these challenges. To begin with, consider the typical formal verification process as shown in Figure 1, which begins with the following three inputs:

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

The verifier generates as output a YES/NO answer, indicating whether or not 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 violated. Some formal verification tools also include a proof or certificate of correctness with a YES answer. In this paper, we use the term "formal verification"
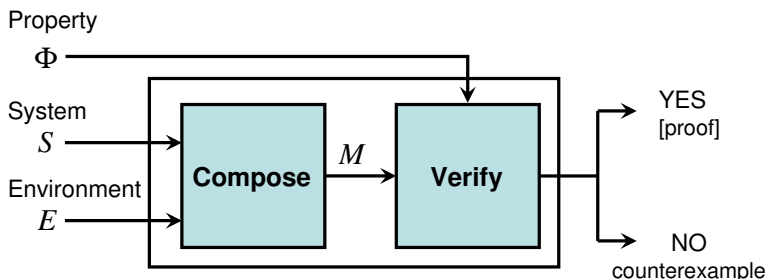

Figure 1: Formal verification procedure.

to apply to any verification technique that uses some aspect of formal methods. For instance, we include simulation-based hardware verification methods that, while based on formal specifications (assertions), employ best-effort heuristics to find violations of those specifications. (The term "semi-formal verification" is sometimes used for such methods.) Such simulation-based verification methods have also found practical use in industrial verification of cyber-physical systems, e.g., for automotive systems [27, 20, 54].

In order to apply formal verification to AI-based systems, at a minimum, one must be able to generate the three inputs S, E and Φ in formalisms for which (ideally) there exist decision procedures to answer the YES/NO question as described above. Additionally, these decision procedures must be efficient. Meeting these requirements, however, is not straightforward. Indeed, in our view, the challenges for Verified AI stem directly from these requirements. We outline these challenges in Section 2 below, and describe ideas to address each of these challenges in Section 3.

2 Challenges for Verified AI

We identify five major challenges to achieving formally-verified AI-based systems. In this section, we sketch out these challenges, illustrating them with examples from the domain of (semi-)autonomous driving.

2.1 Environment Modeling

In the traditional success stories for formal verification, such as verifying cache coherence protocols or device drivers, the *interface* between the system S and its environment E is well defined. Moreover, while the environment itself may not be known, it is usually acceptable to model it as a *non-deterministic* process subject to constraints specified in a suitable logic or automata-based formalism. Typically such an environment model is "over-approximate", meaning that it may include more environment behaviors than are possible.

We see systems based on AI or machine learning (ML) as being quite different. Consider an autonomous vehicle operating in rush-hour traffic in an urban environment. It may be impossible even to precisely define the interface between the system and environment (i.e., to identify the variables/features of the environment that must be modeled), let alone to model all possible behaviors of the environment. Even if the interface is

known, non-deterministic or over-approximate modeling is likely to produce too many spurious bug reports, rendering the verification process useless in practice.

Similarly, for systems involving joint human-machine control, such as semi-autonomous vehicles, human agents are a key part of the environment and/or system. Researchers have attempted modeling humans as non-deterministic or stochastic processes with the goal of verifying the correctness of the overall system [44, 47]. Given the variability and uncertainty in human behavior, a data-driven approach based on machine learning is usually necessary. Such an approach, in turn, is sensitive to the quality of data. For example, the technique of inverse reinforcement learning [38] can be used for learning the reward function of human agents [1, 55]. However, accuracy of the learned reward function depends on the expressivity of the hand-coded features by the designer and the amount and variety of the data collected. In order to achieve Verified AI for such human-in-the-loop systems, we need to address the limitations of the current human modeling techniques and provide guarantees about their *prediction accuracy and convergence*. When learned models are used, one must represent any *uncertainty in the learned parameters* as a first-class entity in the model, and take that into account in verification and control.

The first challenge, then, is to come up with a method of environment modeling that allows one to provide provable guarantees on the system's behavior even when there is considerable uncertainty about the environment.

2.2 Formal Specification

Formal verification critically relies on having a formal specification – a precise, mathematical statement of what the system is supposed to do. However, the challenge of coming up with a high-quality formal specification is well known, even in application domains in which formal verification has found considerable success (see, e.g., [6]).

This challenge is only exacerbated in AI-based systems. Consider a module in an autonomous vehicles that performs object recognition, distinguishing humans from other objects. What is the specification for such a module? How might it differ from the specifications used in traditional applications of formal methods? What should the specification language be, and what tools can one use to construct a specification?

Thus, the second challenge is to find an effective method to specify desired and undesired properties of systems that use AI- or ML-based components.

2.3 Modeling Systems that Learn

In most traditional applications of formal verification, the system S is precisely known: it is a C program, or a circuit described in a hardware description language. The system modeling problem is primarily concerned with reducing the size of the S to a more tractable representation by abstracting away irrelevant details.

AI-based systems lead to a very different challenge for system modeling. A major challenge is the use of *machine learning*, where the system evolves as it encounters new data and new situations. Modeling a deep neural network that has been trained on millions of data points can be challenging enough even if one "freezes" the training process: new abstraction techniques will be necessary. Additionally, the verification procedure must account for future changes in the learner as new data arrives. New techniques must be devised to formally model components based on machine learning.

2.4 Generating Training Data

Formal methods has proved effective for the systematic generation of test data in various settings including simulation-based verification of circuits (e.g., [29]) and finding security exploits in commodity software (e.g., [4]). In these cases, even though the end result is not a proof of correctness of the system S, the

generated tests raise the level of assurance in the system's correctness. Can the testing of AI-based systems leverage formal methods in a similar manner?

Recent efforts have shown that various machine learning algorithms can fail under small adversarial perturbations [39, 21, 37, 24]. Learning algorithms promise to generalize from data, but such simple perturbations that fool the algorithms create concerns regarding their use in safety-critical applications such as autonomous driving. Such small perturbations might be even unrecognizable to humans, but drive the algorithm to misclassify the perturbed data.

The fourth challenge we identify is to devise techniques based on formal methods to systematically generate training and testing data for ML-based components.

2.5   Scalability of Verification Engines

A constant question asked of formal verification is whether it can scale up to handle industrial designs. Much progress has been made in this regard, especially in the area of hardware verification, where formal methods are a standard part of the design flow.

However, in systems that use AI or ML, the scalability challenge is even greater. In addition to the scale of systems as measured by traditional metrics (dimension of state space, number of components, etc.), the *types* of components can be much more complex. For instance, in (semi-)autonomous driving, autonomous vehicles and their controllers need to 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.

3   Principles for Verified AI

For each of the challenges described in the preceding section, we suggest a corresponding set of principles to follow in the design/verification process to address that challenge. These five principles are:

1. *Introspect* on the system to model the environment;
2. Formally specify *end-to-end behavior* of the AI-based system;
3. Develop *abstractions* for and *explanations* from ML components;
4. Create a new class of *randomized formal methods* for systematically generating training/test data, and
5. Develop computational engines for *run-time, quantitative, and learning-based verification*.

We do not attempt to be exhaustive here. Our goal is to bring up ideas that complement other suggestions that have been made in the literature (e.g., [3]). Taken together with other ideas, we believe that the principles we suggest can point a way towards the goal of Verified AI.

3.1   Introspective Environment Modeling

Recall from Sec. 2.1 the challenge of modeling the environment E of an AI-based system S. We believe that a promising strategy to meet this challenge is to develop design and verification methods that are *introspective*, i.e., they identify assumptions A that system S makes about the environment E that are sufficient to guarantee the satisfaction of the specification Φ. The assumptions A must be such that, at run time, S can *efficiently monitor A* so as to ensure that they always hold. Moreover, if there is a human operator involved, one might want A to be translatable into an explanation that is *human understandable*, so that S can "explain" to the human why it may not be able to satisfy the specification Φ.

Ideally, the assumptions *A* must be the weakest set of such assumptions that *S* makes about its environment. However, given the other requirements for *A* to be efficiently monitorable and human understandable, one may need to settle for a stronger assumption.

As an example, consider an autonomous vehicle that is trying to maintain a minimum distance from any other object while being in motion — this forms the specification Φ. Note that Φ defines an interface and a set of sensors that the vehicle *S* must use to check for itself that Φ is satisfied. On top of this minimal interface, suppose that *S* tracks other features of the environment *E* such as the state of traffic lights, the number of vehicles in its vicinity, their state such as their velocity, whether they are human-driven, an estimate of those human drivers' intent and driving style, etc. It will then need to generate assumptions *A* to monitor over this expanded interface (as well as its internal state) so as to ensure that when *A* is satsified, so is Φ.

Extracting good assumptions may be easier during the design process, e.g., while synthesizing a controller for *S*. Preliminary work by the authors has shown that such extraction of monitorable assumptions is feasible in simple cases [31, 32, 23], although much more research is required to make this practical.

3.2 End-to-End Specifications and Specification Mining

Writing formal specifications for AI/ML components is hard, even impossible if the task involves a version of the Turing test. How can we address this challenge described in Sec. 2.2?

As researchers often say: when the problem is too hard, perhaps we should change the problem! We believe that formally specifying the behavior of an AI/ML component may be unnecessary. Instead, one should focus on *precisely specifying the end-to-end behavior* of the *entire AI-based system*. By "end-to-end" we mean the specification on the entire system, not just on the AI/ML component; this is sometimes referred to as a "system-level" specification. We believe that this latter task, in many if not most cases, can be done more easily.

Consider again our autonomous vehicle scenario from the previous section. It should be straightforward to specify the property Φ corresponding to maintaining a minimum distance from any object during motion. This property says nothing about any component that uses machine learning.

Of course, in order to test the ML-based component, it is useful to have a formal specification on its interface. However, we believe this specification does not need to be exact: a "likely specification" could suffice. We suggest the use of techniques for inferring specifications from behaviors and other artifacts — so-called *specification mining* techiques (e.g., [19, 30, 27]), for this purpose.

3.3 Abstractions and Explanations for Machine Learning

Let us now consider the challenge, described in Sec. 2.3, of modeling systems *S* that learn from experience. We believe a combination of *automated abstraction* and *explanation-based learning* will be needed to model such systems for purposes of formal verification.

First, effective techniques need to be developed to abstract ML components into a formalism for which efficient verification techniques exist or can be developed. Since the guarantees many ML algorithms give are probabilistic, this will require the development of *probabilistic logics* and similar formalisms that can capture these guarantees (e.g., [48]). Additionally, if the output of a learning algorithm is accompanied by a measure of *uncertainty* about its correctness, then that uncertainty must be propagated to the rest of the system and represented in the model of the overall system. For example, the formalism of *convex Markov decision processes* (convex MDPs) [40, 42, 47] provide a way of representing uncertainty in the values of learned transition probabilities. Algorithms for verification and control may then need to be extended to handle these new abstractions (see, e.g., [42]).

The task of modeling a learning system can be made easier if the learner accompanies its predictions with *explanations* of how those predictions result from the data and background knowledge. In fact, this idea is not new – it has long been investigated by the ML community under terms such as *explanation-based generalization* [36]. When such explanations are made compatible with the modeling languages used in formal methods, the task of system modeling for verification will be considerably easier.

The literature in formal methods on *explaining failures or counterexamples* may also be relevant. For example, assume that a misclassification by a ML component causes a failure of an end-to-end specification. If we can apply techniques from the formal methods literature to localize that failure (e.g., [30]), then we could identify whether the ML-component was responsible. *Counterfactual reasoning* has been used in the formal methods literature for explaining failures, and we believe such an approach will also be useful in the context of AI-based systems.

3.4 Randomized Formal Methods for Systematic Testing and Exploration

Consider the challenge, described in Sec. 2.4, of generating training data for a ML component in an AI-based system. More concretely, suppose we wish to systematically test a classifier $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that given a set of data points (images, audio, etc.) $\mathbf{x} \in \mathbb{R}^n$ assigns a real-valued label to them $f(\mathbf{x}) \in \mathbb{R}$. One testing problem is to find a perturbation $\mathbf{r} \in \mathbb{R}^n$ such that the algorithm flips the label it assigns to (many) examples upon perturbation, i.e., $f(\mathbf{x}) \neq f(\mathbf{x} + \mathbf{r})$.

Such perturbations cannot be done arbitrarily. One challenge is to define the space of "legal" perturbations so that the resulting examples are still legal inputs that look "realistic". Additionally, one might need to impose constraints on the distribution of the generated examples in order to obtain guarantees about convergence of the learning algorithm to the true concept. How do we meet all these requirements?

We believe that the answer may lie in a new class of *randomized formal methods* – randomized algorithms for generating test inputs subject to formal constraints and distribution requirements. Specifically, a recently defined class of techniques, termed *control improvisation* [22], holds much promise. An improviser is a generator of random strings (examples) $\mathbf{x}$ that satisfy three constraints: (i) a *hard constraint* that defines the space of legal $\mathbf{x}$; (ii) a *soft constraint* defining how the generated $\mathbf{x}$ must be similar to real-world examples, and (iii) a *randomness requirement* defining a constraint on the output distribution. The theory of control improvisation is still in its infancy, and we are just starting to understand the computational complexity and to devise efficient algorithms. Much more remains to be done.

Another challenge for the systematic generation of training data for ML components is to model those components and their requirements in a suitable manner. To deal with this, we think techniques for *black-box test generation* for cyber-physical systems (e.g. [17]) can be useful in generating training and test data for ML components. These methods would treat ML components as black-box components and generate training data using an end-to-end formal specification.

Additionally, there has been considerable recent work on *safe learning-based control* (e.g., [10, 11, 2]). In this approach, a safety envelope is pre-computed and a learning algorithm is used to tune a controller within that envelope. However, how does one generate "interesting" trajectories within the safety envelope to train the learning system well? We believe the theory of control improvisation could be useful here.

3.5 Engines for Run-Time, Quantitative, and Learning-Based Verification

Once effective methods for formalizing S, E, and Φ are devised for AI-based systems, then one must design engines for deciding whether S satisfies the specification Φ in the environment E. At this point, the scalability challenge described in Sec. 2.5 will loom large. We will need the analogs of SAT, BDDs, and SMT for AI-based systems.

Although it is too early to know exactly what formalisms will be best suited for AI-based systems, and therefore what computational engines are needed, we believe a few directions are clear.

First, the complexity and heterogeneity of AI-based systems means that, in general, many decision problems underlying formal verification are likely to be 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 either (i) find tractable but realistic problem classes, or (ii) settle for incomplete or unsound formal verification methods (i.e., semi decision procedures). Techniques for simulation-based verification or run-time verification can prove very fruitful in this regard, as has been recently demonstrated for industrial automotive systems (e.g. [18, 20, 15, 54]).

Further, we believe that it will be important to move from verification problems with purely "Boolean" YES/NO answers to problems with quantitative solutions. Such a move is motivated by the use of probabilistic modeling as well as a need for defining cost-based specifications with the goal of minimizing cost. Some initial steps have been taken with the use of logics with quantitative semantics, such as signal temporal logic [33], and automata-theoretic formalisms such as weighted automata [9], but much more remains to be done. Similarly, work on SMT solving must be extended to more effectively handle cost constraints — in other words, combining SMT solving with optimization methods (e.g., [52, 7]).

Finally, we believe that verification methods themselves are benefiting from the use of inductive learning [49, 50]. An emerging topic in formal methods, termed *formal inductive synthesis* [26], shows much promise for tackling hard problems in verification and synthesis by leveraging expert human insight in combination with induction and deduction.

4 Conclusion

Taking a formal methods perspective, we have analyzed the challenge of formally verifying systems that use artificial intelligence or machine learning. We identified five main challenges: environment modeling, formal specification, system modeling, generating training/test data, and scalability. For each of these five challenges, we have identified corresponding principles for design and verification that hold promise for addressing that challenge. We are currently engaged in developing the theory behind these principles, and applying them to the design of human cyber-physical systems [51], with a special focus on semi-autonomous driving, and expect to report on the results in the years to come.

Acknowledgments

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] Anayo Akametalu. Reachability-based safe learning with gaussian processes. Master's thesis, EECS Department, University of California, Berkeley, Aug 2015.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[4] Thanassis Avgerinos, Sang Kil Cha, Alexandre Rebert, Edward J. Schwartz, Maverick Woo, and David Brumley. Automatic exploit generation. *Commun. ACM*, 57(2):74–84, 2014.

[5] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Armin Biere, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 4, chapter 8. IOS Press, 2009.

[6] I. Beer, S. Ben-David, C. Eisner, and Y. Rodeh. Efficient detection of vacuity in ACTL formulas. *Formal Methods in System Design*, 18(2):141–162, 2001.

[7] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. νz-an optimizing SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 194–199. Springer, 2015.

[8] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[9] Krishnendu Chatterjee, Laurent Doyen, and Thomas A Henzinger. Quantitative languages. *ACM Transactions on Computational Logic (TOCL)*, 11(4):23, 2010.

[10] M. Chen, J. F. Fisac, S. Sastry, and C. J. Tomlin. Safe sequential path planning of multi-vehicle systems via double-obstacle hamilton-jacobi-isaacs variational inequality. In *14th European Control Conference (ECC)*, 2015.

[11] M. Chen, Q. Hu, C. Mackin, J. F. Fisac, and C. J. Tomlin. Safe platooning of unmanned aerial vehicles via reachability. In *54th IEEE Conference on Decision and Control (CDC)*, 2015.

[12] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.

[13] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 2000.

[14] Edmund M Clarke and Jeannette M Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996.

[15] Jyotirmoy Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A. Seshia. Robust online monitoring of signal temporal logic. In *Proceedings of the International Conference on Runtime Verification (RV)*, volume 9333 of *Lecture Notes in Computer Science*, pages 55–70. Springer, September 2015.

[16] Thomas G Dietterich and Eric J Horvitz. Rise of concerns about AI: reflections and directions. *Communications of the ACM*, 58(10):38–40, 2015.

[17] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, pages 167–170, 2010.

[18] Alexandre Donzé, Xiaoqing Jin, Jyotirmoy V. Deshmukh, and Sanjit A. Seshia. Automotive systems requirement mining using Breach. In *American Control Conference (ACC)*, page 4097, 2015.

[19] Michael Ernst. *Dynamically Discovering Likely Program Invariants*. PhD thesis, University of Washington, Seattle, 2000.

[20] Georgios E. Fainekos. Automotive control design bug-finding with the S-TaLiRo tool. In *American Control Conference (ACC)*, page 4096, 2015.

[21] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*, 2015.

[22] Daniel J. Fremont, Alexandre Donzé, Sanjit A. Seshia, and David Wessel. Control improvisation. In *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*, pages 463–474, 2015.

[23] Shromona Ghosh, Dorsa Sadigh, Pierluigi Nuzzo, Vasumathi Raman, Alexandre Donzé, Alberto L. Sangiovanni-Vincentelli, S. Shankar Sastry, and Sanjit A. Seshia. Diagnosis and repair for synthesis from signal temporal logic specifications. In *Proceedings of the 9th International Conference on Hybrid Systems: Computation and Control (HSCC)*, April 2016.

[24] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[25] M. J. C. Gordon and T. F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.

[26] S. Jha and S. A. Seshia. A Theory of Formal Synthesis via Inductive Learning. *ArXiv e-prints*, May 2015.

[27] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy Deshmukh, and Sanjit A. Seshia. Mining requirements from closed-loop control models. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control (HSCC)*, April 2013.

[28] Matt Kaufmann, Panagiotis Manolios, and J. Strother Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, 2000.

[29] Nathan Kitchen and Andreas Kuehlmann. Stimulus generation for constrained random simulation. In *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 258–265. IEEE Press, 2007.

[30] Wenchao Li. *Specification Mining: New Formalisms, Algorithms and Applications*. PhD thesis, EECS Department, University of California, Berkeley, Mar 2014.

[31] Wenchao Li, Lili Dworkin, and Sanjit A. Seshia. Mining assumptions for synthesis. In *Proceedings of the Ninth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, pages 43–50, July 2011.

[32] Wenchao Li, Dorsa Sadigh, S. Shankar Sastry, and Sanjit A. Seshia. Synthesis for human-in-the-loop control systems. In *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 470–484, April 2014.

[33] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, pages 152–166, 2004.

[34] Sharad Malik and Lintao Zhang. Boolean satisfiability: From theoretical hardness to practical success. *Communications of the ACM (CACM)*, 52(8):76–82, 2009.

[35] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[36] Tom M Mitchell, Richard M Keller, and Smadar T Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine learning*, 1(1):47–80, 1986.

[37] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv preprint arXiv:1511.04599*, 2015.

[38] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 663–670, 2000.

[39] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. IEEE, 2015.

[40] A. Nilim and L. El Ghaoui. Robust Control of Markov Decision Processes with Uncertain Transition Matrices. *Journal of Operations Research*, pages 780–798, 2005.

[41] S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer-Verlag, June 1992.

[42] Alberto Puggelli, Wenchao Li, Alberto Sangiovanni-Vincentelli, and Sanjit A. Seshia. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In *Proceedings of the 25th International Conference on Computer-Aided Verification (CAV)*, July 2013.

[43] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In *Symposium on Programming*, number 137 in LNCS, pages 337–351, 1982.

[44] John Rushby. Using model checking to help discover mode confusions and other automation surprises. *Reliability Engineering & System Safety*, 75(2):167–177, 2002.

[45] Stuart Russell, Tom Dietterich, Eric Horvitz, Bart Selman, Francesca Rossi, Demis Hassabis, Shane Legg, Mustafa Suleyman, Dileep George, and Scott Phoenix. Letter to the editor: Research priorities for robust and beneficial artificial intelligence: An open letter. *AI Magazine*, 36(4), 2015.

[46] Stuart Jonathan Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice hall, 2010.

[47] Dorsa Sadigh, Katherine Driggs-Campbell, Alberto Puggelli, Wenchao Li, Victor Shia, Ruzena Bajcsy, Alberto L. Sangiovanni-Vincentelli, S. Shankar Sastry, and Sanjit A. Seshia. Data-driven probabilistic modeling and verification of human driver behavior. In *Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symposium*, March 2014.

[48] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[49] Sanjit A. Seshia. Sciduction: Combining induction, deduction, and structure for verification and synthesis. In *Proceedings of the Design Automation Conference (DAC)*, pages 356–365, June 2012.

[50] Sanjit A. Seshia. Combining induction, deduction, and structure for verification and synthesis. *Proceedings of the IEEE*, 103(11):2036–2051, 2015.

[51] Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Formal methods for semi-autonomous driving. In *Proceedings of the Design Automation Conference (DAC)*, pages 148:1–148:5, June 2015.

[52] Yasser Shoukry, Pierluigi Nuzzo, Alberto Puggelli, Alberto L. Sangiovanni-Vincentelli, Sanjit A. Seshia, Mani Srivastava, and Paulo Tabuada. Imhotep-SMT: A satisfiability modulo theory solver for secure state estimation. In *In 13th International Workshop on Satisfiability Modulo Theories (SMT)*, July 2015.

[53] Jeannette M Wing. A specifier's introduction to formal methods. *IEEE Computer*, 23(9):8–24, September 1990.

[54] Tomoya Yamaguchi, Tomoyuki Kaga, Alexandre Donze, and Sanjit A. Seshia. Combining requirement mining, software model checking, and simulation-based verification for industrial automotive systems. Technical Report UCB/EECS-2016-124, EECS Department, University of California, Berkeley, June 2016.

[55] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.