Towards Verified Artificial Intelligence

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

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

† Stanford University
dorsa@cs.stanford.edu

July 21, 2020

Abstract

Verified artificial intelligence (AI) is the goal of designing AI-based systems that have strong, ideally provable, assurances of correctness 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) is a term used for computational systems that attempt to mimic aspects of human intelligence, including functions we intuitively associate with human minds such as 'learning' and 'problem solving' (e.g., see [17]). Russell and Norvig [66] describe AI as the study of general principles of rational agents and components for constructing these agents. We interpret the term AI broadly to include closely-related areas such as machine learning (ML) [53]. 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, finance, 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 [64, 21, 4]. 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 have strong, ideally provable, assurances of correctness with respect to mathematically-specified requirements. 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 [86, 16]. 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 [29], to model checking [14, 62, 15] and theorem proving (see, e.g. [58, 43, 37]) 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 [50], Boolean reasoning and manipulation routines based on Binary Decision Diagrams (BDDs) [9], and satisfiability modulo theories (SMT) solvers [6].

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 ideas to overcome these challenges. We emphasize that our discussion is focused on the role of formal methods and does not cover the broader set of techniques

that could be used to improve assurance in AI-based systems. Additionally, we seek to identify challenges applicable to a broad range of AI/ML systems, and not limited to specific technologies such as deep neural networks (DNNs) or reinforcement learning (RL) systems. Our view of the challenges is largely shaped by problems arising from the use of AI and ML in autonomous and semi-autonomous systems, though we believe the ideas presented here apply more broadly.

We begin in Sec. 2 with some brief background on formal verification and an illustrative example. We then outline challenges for Verified AI in Sec. 3 below, and describe ideas to address each of these challenges in Sec. 4.[1]

2 Background and Illustrative Example

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 take a broad view of

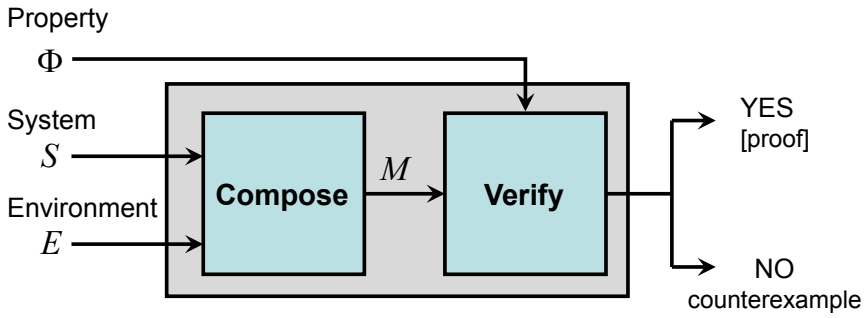

Figure 1: Formal verification procedure.

formal methods: any technique that uses some aspect of formal specification, or verification, or synthesis, is included. 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.

In order to apply formal verification to AI-based systems, at a minimum, one must be able to represent the three inputs S, E and Φ in formalisms for which (ideally) there exist efficient decision procedures to answer the YES/NO question as described above. However, as we describe in Sec. 3, even constructing good representations of the three inputs is not straightforward, let alone dealing with the complexity of the underlying decision problems and associated design issues.

We will illustrate the ideas in this paper with examples from the domain of (semi-)autonomous driving. Fig 2 shows an illustrative example of an AI-based system: a closed-loop cyber-physical system comprising

[1]The first version of this paper was published in July 2016 in response to the call for white papers for the CMU Exploratory Workshop on Safety and Control for AI held in June 2016, and a second version in October 2017. This is the latest version reflecting the evolution of the authors' view of the challenges and approaches for Verified AI.

a semi-autonomous vehicle with machine learning components along with its environment. Specifically, assume that the semi-autonomous "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 subsystem under control including other parts of the autonomy stack), and a sensor (camera) along with a perception component implemented using a deep neural network. The AEBS, when combined with the vehicle's environment, forms a closed loop cyber-physical system. The controller regulates the acceleration and braking of the plant using the velocity of the ego vehicle and the distance between it and an obstacle. The environment of the ego vehicle comprises both agents and objects outside the vehicle (other vehicles,

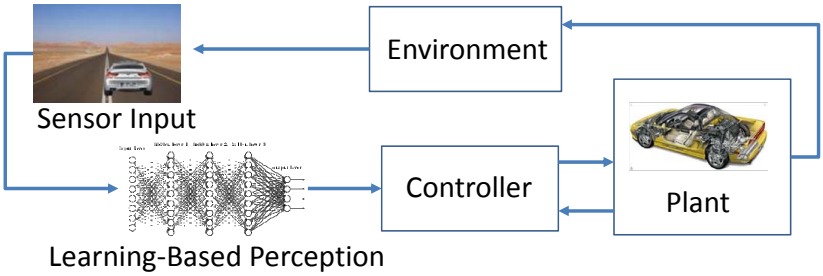

Figure 2: Example of closed-loop cyber-physical system with machine learning components (introduced in [22]).

pedestrians, road objects, etc.) as well as inside the vehicle (e.g., 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, as we will see in Sec. 3, there are many nuances to the specification, modeling, and verification of a system such as this one.

3 Challenges for Verified AI

We identify five major challenges to achieving formally-verified AI-based systems, described in more detail below.

3.1 Environment Modeling

The *environments* in which AI/ML-based systems operate can be very complex, with considerable uncertainty even about how many and which agents are in the environment (both human and robotic), let alone about their intentions and behaviors. As an example, consider the difficulty in modeling urban traffic environments in which an autonomous car must operate. Indeed, AI/ML is often introduced into these systems precisely to deal with such complexity and uncertainty! From a formal methods perspective, this makes it very hard to create realistic environment models with respect to which one can perform verification or synthesis.

We see the main challenges for environment modeling as being threefold:

- *Unknown Variables:* 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. The environment can only influence the system through this interface. However, for AI-based systems, such as an autonomous vehicle example of Sec. 2, it may be impossible to precisely define all the variables (features) of the environment. Even in restricted scenarios where the environment variables

(agents) are known, there is a striking lack of information, especially at design time, about their behaviors. Additionally, modeling sensors such as LiDAR that represent the interface to the environment is in itself a major technical challenge.

- *Modeling with the Right Fidelity:* In traditional uses of formal verification, it is usually acceptable to model the environment as a *non-deterministic* process subject to constraints specified in a suitable logic or automata-based formalism. Typically such an environment model is termed as being "over-approximate", meaning that it may include (many) more environment behaviors than are possible. Over-approximate environment modeling permits one to perform sound verification without a detailed environment model, which can be inefficient to reason with and hard to obtain. However, for AI-based autonomy, purely non-deterministic modeling is likely to produce highly over-approximate models, which in turn yields too many spurious bug reports, rendering the verification process useless in practice. Moreover, many AI-based systems make distributional assumptions on the environment, thus requiring the need for probabilistic modeling; however, it can be difficult to exactly ascertain the underlying distributions. One can address this by learning a probabilistic model from data, but in this case it is important to remember that the model parameters (e.g., transition probabilities) are only estimates, not precise representations of environment behavior. Thus, verification algorithms cannot consider the resulting probabilistic model to be "perfect"; we need to represent uncertainty in the model itself.
- *Modeling Human Behavior:* For many AI-based systems, 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 [63, 67]. However, such approaches must deal with the variability and uncertainty in human behavior. One could take a data-driven approach based on machine learning (e.g., [55]), but such an approach is sensitive to the expressivity of the features used by the ML model and the quality of data. In order to achieve Verified AI for such human-in-the-loop systems, we need to address the limitations of 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 systematic 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.

3.2 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., [7]). This challenge is only exacerbated in AI-based systems. We identify three major problems.

Specification for Hard-to-Formalize Tasks: Consider the perception module in the AEBS controller of Fig. 2 which must detect and classify objects, distinguishing vehicles and pedestrians from other objects. Correctness for this module in the classic formal methods sense requires a formal definition of each type of road user, which is extremely difficult, if not impossible. Similar problems arise for other tasks involving perception and communication, such as natural language processing. How then, do we specify correctness properties for such a module? What should the specification language be and what tools can one use to construct a specification?

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?

Data vs. Formal Requirements: The view of "data as specification" is common in machine learning. Labeled "ground truth" data is often the only specification of correct behavior. On the other hand, a specification in formal methods is a mathematical property that defines the set of correct behaviors. How can we bridge this gap?

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

3.3 Modeling Learning Systems

In most traditional applications of formal verification, the system S is precisely known: it is a program or 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-based systems lead to a very different challenge for system modeling, primarily stemming from the use of *machine learning*:

- *Very high-dimensional input space:* ML components used for perception usually operate over very high-dimensional input spaces. For the illustrative example of Sec. 2 from [22], each input RGB image is of dimension 1000×600 pixels, contains $256^{1000\times600\times3}$ elements, and in general the input is a stream of such high-dimensional vectors. Although formal methods has been used for high-dimensional input spaces (e.g., 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.
- *Very 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 the example of Fig. 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 reinforcement learning, 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 else 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 robustness of a DNN such as the one in Fig. 2 requires us to capture a model of the surrounding system. We need techniques to model ML components along with their *context* so that semantically meaningful properties can be verified.

3.4 Efficient and Scalable Design and Verification of Models and Data

The effectiveness of formal methods in the domains of hardware and software has been driven by advances in underlying "computational engines" — e.g., SAT, SMT, numerical simulation, and model checking. Given the scale of AI/ML systems, the complexity of their environments, and the new types of specifications involved, several advances are needed in creating computational engines for efficient and scalable training, testing, design, and verification of AI-based systems. We identify here the key challenges that must be overcome in order to achieve these advances.

Data Generation: Data is the fundamental starting point for machine learning. Any quest to improve the quality of a machine learning system must improve the quality of the data it learns from. Can formal methods help to systematically select, design and augment the data used for machine learning?

We believe the answer is yes, but that more needs to be done. Formal methods has proved effective for the systematic generation of counterexamples and test data that satisfy constraints including for simulation-based verification of circuits (e.g., [44]) and finding security exploits in commodity software (e.g., [5]). However, the requirements for AI/ML systems are different. The types of constraints can be much more complex, e.g., encoding requirements on "realism" of data captured using sensors from a complex environment such as a traffic situation. We need to generate not just single data items, but an ensemble that satisfies distributional constraints. Additionally, data generation must be selective, e.g., meeting objectives on data set size and diversity for effective training and generalization. All of these additional requirements necessitate the development of a new suite of formal techniques.

Quantitative Verification: Several safety-critical applications of AI-based systems are in robotics and cyber-physical systems. In such systems, the scalability challenge for verification can be very high. 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. Yet, most of the existing verification methods are targeted towards answering Boolean verification questions. To address this gap, new scalable engines for quantitative verification must be developed.

Compositional Reasoning: In order for formal methods to scale to large AI/ML systems, compositional (modular) reasoning is essential. In compositional verification, a large system (e.g., program) is split up into its components (e.g., 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. A theory of assume-guarantee contracts does not yet exist for AI-based systems.

Moreover, AI/ML systems pose a particularly vexing challenge for compositional reasoning. Compositional verification requires *compositional specification* — i.e., the components must be formally-specifiable. However, as noted in Sec. 3.2, 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 [75]. Additionally, more work needs to be done for extending the theory and application of compositional reasoning to probabilistic systems and specifications.

3.5 Correct-by-Construction Intelligent Systems

In an ideal world, verification should be integrated with the design process so that the system is "correct-by-construction." Such an approach could either interleave verification steps with compilation/synthesis steps, such as in the register-transfer-level (RTL) design flow common in integrated circuits, or devise synthesis algorithms so as to ensure that the implementation satisfies the specification, such as in reactive synthesis from temporal logic [60]. Can we devise a suitable correct-by-construction design flow for AI-based systems?

Specification-Driven Design of ML Components: Can we design, from scratch, a machine learning component (model) that provably satisfies a formal specification? (This assumes, of course, that we solve the formal specification challenge described above in Sec. 3.2.) The clean-slate design of an ML component has many aspects: (1) designing the data set, (2) synthesizing the structure of the model, (3) generating a

good set of features, (4) synthesizing hyper-parameters and other aspects of ML algorithm selection, and (5) automated techniques for debugging ML models or the specification when synthesis fails. More progress is needed on all these fronts.

Theories of Compositional Design: Another challenge is to design the overall system comprising multiple learning and non-learning components. While theories of compositional design have been developed for digital circuits and embedded systems (e.g. [70, 80]), we do not as yet have such theories for AI-based systems. For example, if two ML models are used for perception on two different types of sensor data (e.g., LiDAR and visual images), and individually satisfy their specifications under certain assumptions, under what conditions can they be used together to improve the reliability of the overall system? And how can one design a planning component so as to overcome limitations of an ML-based perception component that it receives input from?

Bridging Design Time and Run Time for Resilient AI: Due to the complexity of AI-based systems and the environments in which they operate, even if all the challenges for specification and verification are solved, it is likely that one will not be able to prove unconditional safe and correct operation. There will always be situations in which we do not have a provable guarantee of correctness. Therefore, techniques for achieving fault tolerance and error resilience at run time must play a crucial role. In particular, there is not yet a systematic understanding of what can be achieved at design time, how the design process can contribute to safe and correct operation of the AI system at run time, and how the design-time and run-time techniques can interoperate effectively.

4 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. Use an *introspective, data-driven, and probabilistic* approach to model the environment;
2. Combine formal specifications of *end-to-end behavior* with *hybrid Boolean-quantitative formalisms* for learning systems and perception components and use *specification mining* to bridge the data-property gap;
3. For ML components, develop new *abstractions*, *explanations*, and *semantic analysis* techniques;
4. Create a new class of *compositional, randomized, and quantitative formal methods* for data generation, testing, and verification, and
5. Develop techniques for *formal inductive synthesis* of AI-based systems and design of *safe learning systems*, supported by techniques for *run-time assurance*.

We have successfully applied these principles over the past few years, and, based on this experience, believe that they provide a good starting point for applying formal methods to AI-based systems. Our formal methods perspective on the problem complements other perspectives that have been expressed (e.g., [4]). Experience over the past few years provides evidence that the principles we suggest can point a way towards the goal of Verified AI.

4.1 Environment Modeling: Introspection, Probabilities, and Data

Recall from Sec. 3.1, the three challenges for modeling the environment E of an AI-based system S: *unknown variables*, *model fidelity*, and *human modeling*. We propose to tackle these challenges with three corresponding principles.

Introspective Environment Modeling: We suggest to address the unknown variables problem by developing design and verification methods that are *introspective*, i.e., they algorithmically identify assumptions A that system S makes about the environment E that are sufficient to guarantee the satisfaction of the specification

Φ [76]. The assumptions A must be ideally the *weakest* such assumptions, and also must be *efficient to generate at design time and monitor at run time* over available sensors and other sources of information about the environment so that mitigating actions can be taken when they are violated. 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. Dealing with these multiple requirements, as well as the need for good sensor models, makes introspective environment modeling a highly non-trivial task that requires substantial progress [76]. Preliminary work by the authors has shown that such extraction of monitorable assumptions is feasible in very simple cases [48], although more research is required to make this practical.

Active Data-Driven Modeling: We believe human modeling requires an *active data-driven* approach. Relevant theories from cognitive science and psychology, such as that of bounded rationality [81, 65], must be leveraged, but it is important for those models to be expressed in formalisms compatible with formal methods. Additionally, while using a data-driven approach to infer a model, one must be careful to craft the right model structure and features. A critical aspect of human modeling is to capture *human intent*. We believe a three-pronged approach is required: first, define model templates/features based on expert knowledge; then, use offline learning to complete the model for design time use, and finally, *learn and update* environment models at run time by monitoring and interact with the environment. Initial work has shown how 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 [67, 69].

Probabilistic Formal Modeling: In order to tackle the model fidelity challenge, we suggest to use formalisms that combine probabilistic and non-deterministic modeling. Where probability distributions can be reliably specified or estimated, one can use probabilistic modeling. In other cases, non-deterministic modeling can be used to over-approximate environment behaviors. While formalisms such as Markov Decision Processes (MDPs) already provide a way to blend probability and non-determinism, we believe techniques that blend probability and logical or automata-theoretic formalisms, such as the paradigm of *probabilistic programming* [52, 32], 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, the formalism of *convex Markov decision processes* (convex MDPs) [56, 61, 67] 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., [61]).

4.2 End-to-End Specifications, Hybrid Specifications, and Specification Mining

Writing formal specifications for AI/ML components is hard, arguably even impossible if the component imitates a human perceptual task. Even so, we think the challenges described in Sec. 3.2 can be addressed by following three guiding principles.

End-to-End/System-Level Specifications: In order to address the specification-for-perception challenge, let us change the problem slightly. We suggest to first focus on *precisely specifying the end-to-end behavior* of the *AI-based system*. By "end-to-end" we mean the specification on the entire closed-loop system (see Fig. 2) or a precisely-specifiable sub-system containing the AI/ML component, not on the component alone. Such a specification is also referred to as a "system-level" specification. For our AEBS example, this involves specifying the property Φ corresponding to maintaining a minimum distance from any object during motion. Starting with such a system-level (end-to-end) specification, we then derive from it constraints on the input-output interface of the perception component that guarantee that the system-level specification is satisfied. Such constraints serve as a partial specification under which the perception component can be analyzed (see [22]). Note that these constraints need not be human-readable.

Hybrid Quantitative-Boolean Specifications: Boolean and quantitative specifications both have their advantages. On the one hand, Boolean specifications are easier to compose. On the other hand, objective functions lend themselves to optimization based techniques for verification and synthesis, and to defining finer granularities of property satisfaction. One approach to bridge this gap is to move to quantitative specification languages, such as logics with both Boolean and quantitative semantics (e.g. STL [49]) or notions of weighted automata (e.g. [13]). Another approach is to combine both Boolean and quantitative specifications into a common specification structure, such as a *rulebook* [10], where specifications can be organized in a hierarchy, compared, and aggregated. Additionally, novel formalisms bridging ideas from formal methods and machine learning are being developed to model the different variants of properties such as robustness, fairness, and privacy, including notions of semantic robustness (see, e.g., [77, 24]).

Specification Mining: In order to bridge the gap between data and formal specifications, we suggest the use of techniques for inferring specifications from behaviors and other artifacts — so-called *specification mining* techiques (e.g., [26, 47]). 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 [85] or more complex forms of interaction between multiple agents, both human and robotic.

4.3 System Modeling: Abstractions, Explanations, and Semantic Feature Spaces

Let us now consider the challenges, described in Sec. 3.3, arising in modeling systems S that learn from experience. In our opinion, advances in three areas are needed in order to address these challenges:

Automated Abstraction: 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. In order 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 advances in this regard include the use of abstract interpretation to analyze deep neural networks (e.g. [35]), the use of abstractions for falsifying cyber-physical systems with ML components [22], and the development of probabilistic logics that capture guarantees provided by ML algorithms (e.g., [68]).

Explanation Generation: 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* [54]. Recently, there has been a renewal of interest in using logic to explain the output of learning systems (e.g. [84, 40]). Such approaches to generating explanations that are compatible with the modeling languages used in formal methods can make the task of system modeling for verification considerably easier. ML techniques that incorporate causal and counterfactual reasoning [59] can also ease the generation of explanations for use with formal methods.

Semantic Feature Spaces: The verification and adversarial analysis [36] of ML models is more meaningful when the generated adversarial inputs and counterexamples have semantic meaning in the context in which the ML models are used. There is thus a need for techniques that can analyze ML models in the context of the systems within which they are used, i.e., for *semantic adversarial analysis* [25]. 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 (e.g. images of traffic scenes) form a much lower dimensional latent space as compared to the full concrete feature space. For our illustrative example in Fig. 2, the 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 (e.g. [22, 38]). However, one typically needs to have a "renderer" that maps a point in the semantic feature space to one in the concrete feature space, and certain properties of this renderer, such as differentiability [46], make it easier to apply formal methods to perform goal-directed search of the semantic feature space.

4.4 Compositional and Quantitative Methods for Design and Verification of Models and Data

Consider the challenge, described in Sec. 3.4, of devising computational engines for scalable training, testing, and verification of AI-based systems. We see three promising directions to tackle this challenge.

Controlled Randomization in Formal Methods: Consider the problem of *data set design* – i.e., systematically generating training data for a ML component in an AI-based system. This synthetic data generation problem 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, one might want to impose constraints on "realism", e.g., a measure of similarity with real-world data. Third, 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. What can formal methods offer towards solving this problem?

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* [31], holds 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. Improvisation, in turn, relies on recent progress on computational problems such as constrained random sampling and model counting (e.g., [51, 11, 12]), and generative approaches based on probabilistic programming (e.g. [32]).

Quantitative Verification on the Semantic Feature Space: Recall the challenge to develop techniques for verification of quantitative requirements – where the output of the verifier is not just YES/NO but a numeric value.

The complexity and heterogeneity of AI-based 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 (e.g. [42, 27, 88]), although they must be applied to the semantic feature space for efficiency [23]. Such falsification techniques can also be used for the systematic, adversarial generation of training data for ML components [23]. Techniques for probabilistic verification, such as probabilistic model checking [45, 18], should be extended beyond traditional formalisms such as Markov chains or Markov Decision Processes to *verify probabilistic programs over semantic feature spaces*. 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., [79, 8]).

Compositional Reasoning: As in all applications of formal methods, modularity will be crucial to scalable verification of AI-based systems. However, compositional design and analysis of AI-based systems faces some unique challenges. First, theories of probabilistic assume-guarantee design and verification need to

be developed for the semantic spaces for such systems, building on some promising initial work (e.g. [57]). Second, we suggest the use of inductive synthesis [74] to generate assume-guaranteee contracts algorithmically, to reduce the specification burden and ease the use of 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 (e.g. [22]) and use such constraints to focus component-level analysis, including adversarial analysis.

4.5 Formal Inductive Synthesis, Safe Learning, and Run-Time Assurance

Developing a correct-by-construction design methodology for AI-based systems, with associated tools, is perhaps the toughest challenge of all. For this to be fully solved, the preceding four challenges must be successfully addressed. However, we do not need to wait until we solve those problems in order to start working on this one. Indeed, a methodology to "design for verification" may well ease the task on the other four challenges.

Formal Inductive Synthesis: First consider the problem of synthesizing learning components correct by construction. The emerging theory of *formal inductive synthesis* [39, 41] addresses this problem. Formal inductive synthesis is the synthesis from examples of programs that satisfy formal specifications. In machine learning terms, it is the synthesis of models/classifiers that additionally satisfy a formal specification. The most common approach to solving a formal inductive synthesis problem is to use an *oracle-guided* approach. In oracle-guided synthesis, a learner is paired with an oracle who answers queries. The set of query-response types is defined by an *oracle interface*. For the example of Fig. 2, the oracle can be a falsifier that can generate counterexamples showing how a failure of the learned component violates the system-level specification. This approach, also known as *counterexample-guided inductive synthesis* [82], has proved effective in many scenarios. In general, oracle-guided inductive synthesis techniques show much promise for the synthesis of learned components by blending expert human insight, inductive learning, and deductive reasoning [73, 74]. These methods also have a close relation to the sub-field of *machine teaching* [89].

Safe Learning by Design: There has been considerable recent work on using design-time methods to analyze or constrain learning components so as to ensure safe operation within specified assumptions. A prominent example is *safe learning-based control* (e.g., [3, 28]). In this approach, a safety envelope is pre-computed and a learning algorithm is used to tune a controller within that envelope. Techniques for efficiently computing such safety envelopes based, for example, on reachability analysis [83], are needed. Relatedly, several methods have been proposed for safe reinforcement learning (see [34]). Another promising direction is to enforce properties on ML models through the use of semantic loss functions (e.g. [87, 25]), though this problem is largely unsolved. Finally, the use of theorem proving for ensuring correctness of algorithms used for training ML models (e.g. [72]) is also an important step towards improving the assurance in ML based systems.

Run-Time Assurance: Due to the undecidability of verification in most instances and the challenge of environment modeling, we believe it will be difficult, if not impossible, to synthesize correct-by-construction AI-based systems or to formally verify correct operation without making restrictive assumptions. Therefore, design-time verification must be combined with *run-time assurance*, i.e., run-time verification and mitigation techniques. For example, the Simplex technique [78] 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 run-time assurance methods (e.g., [71, 19, 20]) 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, the problems of extracting environment assumptions and synthesizing them into runtime monitors (e.g., as described for introspective environment modeling [76]) and devising runtime mitigation strategies remain a largely unsolved problem.

Challenges	Principles
Environment (incl. Human) Modeling	Active Data-Driven, Introspective, Probabilistic Modeling
Formal Specification	Start at System Level, Derive Component Specifications; Hybrid Boolean-Quantitative Specification; Specification Mining
Modeling Learning Systems	Abstractions, Explanations, Semantic Feature Spaces
Scalable Design & Verification of Data and Models	Compositional Reasoning, Controlled Randomization, Quantitative Semantic Analysis
Correct-by-Construction Methods	Formal Inductive Synthesis, Safe Learning by Design, Run-Time Assurance

Table 1: Summary of the five challenges for Verified AI presented in this paper, and the corresponding principles proposed to address them.

5 Conclusion

Taking a formal methods perspective, we have analyzed the challenge of developing and applying formal methods to systems that are substantially based on artificial intelligence or machine learning. As summarized in Table 1, we have identified five main challenges for applying formal methods to AI-based systems. For each of these five challenges, we have identified corresponding principles for design and verification that hold promise for addressing that challenge. Since the original version of this paper was published in 2016, several researchers including the authors have been working on addressing these challenges; a few sample advances are described in this paper. In particular, we have developed open-source tools, VerifAI [2] and Scenic [1] that implement techniques based on the principles described in this paper, and which have been applied to industrial-scale systems in the autonomous driving [33] and aerospace [30] domains. These results are but a start and much more remains to be done. The topic of Verified AI promises to continue to be a fruitful area for research in the years to come.

Acknowledgments

The authors' work has been supported in part by NSF grants CCF-1139138, CCF-1116993, CNS-1545126 (VeHICaL), CNS-1646208, and CCF-1837132 (FMitF), by an NDSEG Fellowship, by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA, by the DARPA BRASS and Assured Autonomy programs, by Toyota under the iCyPhy center, and by Berkeley Deep Drive. We gratefully acknowledge the many colleagues with whom our conversations and collaborations have helped shape this article.

References

[1] Scenic Environment Modeling and Scenario Description Language. http://github.com/BerkeleyLearnVerify/Scenic.

[2] VerifAI: A toolkit for design and verification of AI-based systems. http://github.com/BerkeleyLearnVerify/VerifAI.

[3] Anayo K Akametalu, Jaime F Fisac, Jeremy H Gillula, Shahab Kaynama, Melanie N Zeilinger, and Claire J Tomlin. Reachability-based safe learning with Gaussian processes. In *53rd IEEE Conference on Decision and Control*, pages 1424–1431, 2014.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] Andrea Censi, Konstantin Slutsky, Tichakorn Wongpiromsarn, Dmitry Yershov, Scott Pendleton, James Fu, and Emilio Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8536–8542. IEEE, 2019.

[11] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. Distribution-aware sampling and weighted model counting for SAT. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1722–1730, July 2014.

[12] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. On parallel scalable uniform sat witness generation. In *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 304–319, April 2015.

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

[14] 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.

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

[16] 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.

[17] Committee on Information Technology, Automation, and the U.S. Workforce. Information technology and the U.S. workforce: Where are we and where do we go from here? http://www.nap.edu/24649.

[18] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *International Conference on Computer Aided Verification (CAV)*, pages 592–600. Springer, 2017.

[19] Ankush Desai, Tommaso Dreossi, and Sanjit A. Seshia. Combining model checking and runtime verification for safe robotics. In *Runtime Verification - 17th International Conference, RV 2017, Seattle, WA, USA, September 13-16, 2017, Proceedings*, pages 172–189, 2017.

[20] Ankush Desai, Shromona Ghosh, Sanjit A. Seshia, Natarajan Shankar, and Ashish Tiwari. A runtime assurance framework for programming safe robotics systems. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2019.

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

[22] Tommaso Dreossi, Alexandre Donze, and Sanjit A. Seshia. Compositional falsification of cyber-physical systems with machine learning components. In *Proceedings of the NASA Formal Methods Conference (NFM)*, May 2017.

[23] Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A. Seshia. VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *31st International Conference on Computer Aided Verification (CAV)*, July 2019.

[24] Tommaso Dreossi, Shromona Ghosh, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. A formalization of robustness for deep neural networks. In *Proceedings of the AAAI Spring Symposium Workshop on Verification of Neural Networks (VNN)*, March 2019.

[25] Tommaso Dreossi, Somesh Jha, and Sanjit A. Seshia. Semantic adversarial deep learning. In *30th International Conference on Computer Aided Verification (CAV)*, 2018.

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

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

[28] Jaime F Fisac, Anayo K Akametalu, Melanie N Zeilinger, Shahab Kaynama, Jeremy Gillula, and Claire J Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

[29] Harry Foster. *Applied Assertion-Based Verification: An Industry Perspective*. Now Publishers Inc., 2009.

[30] Daniel J. Fremont, Johnathan Chiu, Dragos D. Margineantu, Denis Osipychev, and Sanjit A. Seshia. Formal analysis and redesign of a neural network-based aircraft taxiing system with verifai. In *32nd International Conference on Computer-Aided Verification (CAV)*, pages 122–134, 2020.

[31] 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.

[32] Daniel J. Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Scenic: A language for scenario specification and scene generation. In *Proceedings of the 40th annual ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI)*, June 2019.

[33] Daniel J. Fremont, Edward Kim, Yash Vardhan Pant, Sanjit A. Seshia, Atul Acharya, Xantha Bruso, Paul Wells, Steve Lemke, Qiang Lu, and Shalin Mehta. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2020.

[34] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[35] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. AI2: Safety and robustness certification of neural networks with abstract interpretation. In *IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2018.

[36] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018.

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

[38] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer, 2017.

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

[40] Susmit Jha, Tuhin Sahai, Vasumathi Raman, Alessandro Pinto, and Michael Francis. Explaining AI decisions using efficient methods for learning sparse boolean formulae. *J. Autom. Reasoning*, 63(4):1055–1075, 2019.

[41] Susmit Jha and Sanjit A. Seshia. A Theory of Formal Synthesis via Inductive Learning. *Acta Informatica*, 2017.

[42] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy Deshmukh, and Sanjit A. Seshia. Mining requirements from closed-loop control models. *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 34(11):1704–1717, 2015.

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

[44] 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.

[45] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer Aided Verification (CAV)*, pages 585–591. Springer, 2011.

[46] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

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

[48] 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.

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

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

[51] Kuldeep S. Meel, Moshe Y. Vardi, Supratik Chakraborty, Daniel J. Fremont, Sanjit A. Seshia, Dror Fried, Alexander Ivrii, and Sharad Malik. Constrained sampling and counting: Universal hashing meets SAT solving. In *Beyond NP, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 12, 2016.*, 2016.

[52] Brian Milch, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L Ong, and Andrey Kolobov. Blog: Probabilistic models with unknown objects. *Statistical Relational Learning*, page 373, 2007.

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

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

[55] 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.

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

[57] Pierluigi Nuzzo, Jiwei Li, Alberto L. Sangiovanni-Vincentelli, Yugeng Xi, and Dewei Li. Stochastic assume-guarantee contracts for cyber-physical system design. *ACM Trans. Embed. Comput. Syst.*, 18(1), January 2019.

[58] 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.

[59] Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019.

[60] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989*, pages 179–190, 1989.

[61] 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.

[62] 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.

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

[64] 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.

[65] Stuart J Russell. Rationality and intelligence. *Artificial Intelligence*, 94(1-2):57–77, 1997.

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

[67] 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.

[68] 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.

[69] Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. Information gathering actions over human internal state. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[70] Alberto Sangiovanni-Vincentelli, Werner Damm, and Roberto Passerone. Taming Dr. Frankenstein: Contract-based design for cyber-physical systems. *European journal of control*, 18(3):217–238, 2012.

[71] John D Schierman, Michael D DeVore, Nathan D Richards, Neha Gandhi, Jared K Cooper, Kenneth R Horneman, Scott Stoller, and Scott Smolka. Runtime assurance framework development for highly adaptive flight control systems. Technical report, Barron Associates, Inc. Charlottesville, 2015.

[72] Daniel Selsam, Percy Liang, and David L. Dill. Developing bug-free machine learning systems with formal mathematics. In *Proceedings of the 34th International Conference on Machine Learning, (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 3047–3056. PMLR, 2017.

[73] 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.

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

[75] Sanjit A. Seshia. Compositional verification without compositional specification for learning-based systems. Technical Report UCB/EECS-2017-164, EECS Department, University of California, Berkeley, Nov 2017.

[76] Sanjit A. Seshia. Introspective environment modeling. In *19th International Conference on Runtime Verification (RV)*, pages 15–26, 2019.

[77] Sanjit A. Seshia, Ankush Desai, Tommaso Dreossi, Daniel Fremont, Shromona Ghosh, Edward Kim, Sumukh Shivakumar, Marcell Vazquez-Chanlatte, and Xiangyu Yue. Formal specification for deep neural networks. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis (ATVA)*, pages 20–34, October 2018.

[78] Lui Sha. Using simplicity to control complexity. *IEEE Software*, 18(4):20–28, 2001.

[79] Yasser Shoukry, Pierluigi Nuzzo, Alberto Sangiovanni-Vincentelli, Sanjit A. Seshia, George J. Pappas, and Paulo Tabuada. Smc: Satisfiability modulo convex optimization. In *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control (HSCC)*, April 2017.

[80] Joseph Sifakis. System design automation: Challenges and limitations. *Proceedings of the IEEE*, 103(11):2093–2103, 2015.

[81] Herbert A Simon. Bounded rationality. In *Utility and Probability*, pages 15–18. Springer, 1990.

[82] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, and Vijay A. Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 404–415. ACM Press, October 2006.

[83] Claire Tomlin, Ian Mitchell, Alexandre M. Bayen, and Meeko Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.

[84] Marcell Vazquez-Chanlatte, Jyotirmoy V. Deshmukh, Xiaoqing Jin, and Sanjit A. Seshia. Logical clustering and learning for time-series data. In *29th International Conference on Computer Aided Verification (CAV)*, pages 305–325, 2017.

[85] Marcell Vazquez-Chanlatte, Susmit Jha, Ashish Tiwari, Mark K. Ho, and Sanjit A. Seshia. Learning task specifications from demonstrations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 5372–5382, December 2018.

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

[87] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning, (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 5498–5507. PMLR, 2018.

[88] 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.

[89] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018.