Compositional Verification without Compositional Specification for Learning-Based Systems

Sanjit A. Seshia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2017-164
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-164.html

November 26, 2017

Compositional Verification without Compositional Specification for Learning-Based Systems

Sanjit A. Seshia
University of California, Berkeley
sseshia@eecs.berkeley.edu

November 23, 2017

Abstract

We consider the problem of performing compositional verification of a system with machine learning components whose behavior cannot easily be formally specified. We present an approach involving a system-level verifier communicating with a component-level analyzer wherein the former identifies a subset of environment behaviors that might lead to a system-level failure while the latter identifies erroneous behaviors, such as misclassifications, of the machine learning component that might be extended to a system-level counterexample. Results on cyber-physical systems with deep learning components used for perception demonstrate the promise of this approach.

1 Introduction

Formal verification critically relies on having a formal specification – a precise, mathematical statement of what the system is supposed to do, and how its environment is expected to behave. The challenge of coming up with high-quality formal specification is well known, even in application domains such as digital design in which formal verification has found considerable success (see, e.g., [3]). However, how can one deal with situations where formally specifying the behavior of a system or a component within that system is extremely difficult or arguably even impossible?

One setting in which to consider this question is that of *compositional verification* — verifying a system by decomposing the overall verification problem into those of verifying the components of the system. Compositional verification — also known as modular verification — has been a cornerstone of getting techniques in formal verification to scale to large circuits and programs. However, compositional verification relies on *compositional specification*, i.e., breaking up the overall system specification into those for its components. The decomposition of the system specification must be performed in a manner such that verifying that the components satisfy their component-level specifications implies that the system satisfies its system-level specification. Frameworks such as *assume-guarantee* (or *rely-guarantee*) reasoning (see, e.g., [2, 11]) and contract-based design [18] provide approaches to coming up with such a decomposition in a principled manner.

The increasing number of systems that make heavy use of *artificial intelligence* (AI) and *machine learning* (ML) pose a particular challenge for compositional specification and verification. Consider, for example, a deep learning [8, 13] based module in an autonomous vehicle that performs object detection and classification, distinguishing cars from bicycles from pedestrians and other objects. One can think of formulating a suitable system-level safety specification for such a system, e.g., that the autonomous vehicle must maintain a safe distance from other objects while it is in motion. However, verifying whether the vehicle satisfies such a system-level specification depends on whether its components work correctly in tandem, including

whether the object detection system correctly identifies objects in the vehicle's vicinity. Further, the verification of such a complex system will inevitably need to be compositional. For example, the rich set of sensors used by an autonomous vehicle results in an extremely high-dimensional input space for verification. Monolithic verification of the entire system is extremely difficult. Thus, the question arises: What is the formal specification for the object detection module?

To answer this question, one will need to formalize visual and perceptual tasks, some of which may involve a version of the Turing test. We believe that such tasks are, in general, not formally specifiable. Even when one might formulate a formal specification for such a perceptual task, it is likely to be brittle due to the inherent ambiguity in the task. As an example, consider Figure 1 which shows three images of cars obtained via a web search. Which of these must an autonomous vehicle consider as a valid car in its environment? Further, even in a simulated environment where "ground truth" may be known, it may be

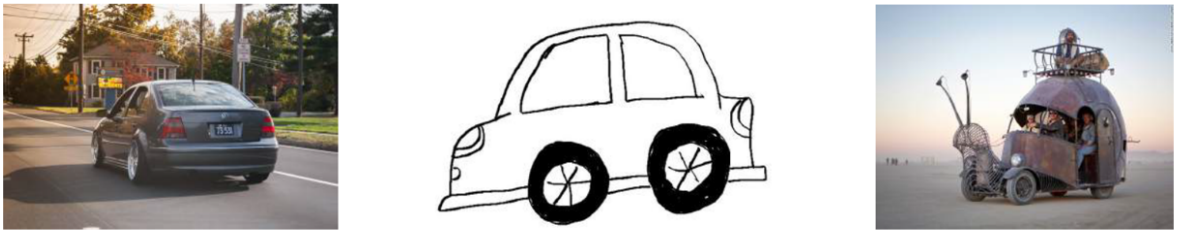

Figure 1: Images of three cars obtained via a web search.

extremely difficult to capture it in a compact formal notation. For instance, using advanced simulators for autonomous driving, one can specify an environment configuration where ground truth may be known for specific environment states, but it is tricky to specify exhaustively and use for formal verification.

In this note, we formulate an alternative approach for the compositional verification of such systems where compositional specification is not possible (at least in the traditional sense). We focus on *learning-based systems*, i.e., systems which make substantial use of machine learning components. An initial demonstration of this approach has been made for verifying cyber-physical systems with machine learning components [5]. This problem is one of the interesting problems arising from the application of formal methods to AI-based systems. For a fuller discussion of the challenges for verified AI and some ideas for addressing them, see [19].

2 Cooperating Verifiers

Let S denote the model of the system S under verification, E denote a model of its environment, and Φ denote the specification to be verified. Φ is a set of behaviors of the closed system obtained by composing S with E, denoted $S\|E$ (we limit ourselves to trace properties). Further, let C denote a component of S for which we cannot (easily) write a formal specification – for instance, the deep learning module discussed in Sec. 1 (and in more detail in Sec. 2.1 below). We focus in particular on the setting where C is designed for a sensing/perceptual task on the environment E. We wish to verify whether the composite system $S\|E$ satisfies Φ, and to do so compositionally by decomposing the verification problem between one that analyzes just the component C and another than analyzes the behavior of S while abstracting away most or all of the details of C.

In this section, we present some initial ideas for tackling this problem, beginning with a motivating example (Sec. 2.1), the core approach (Sec. 2.2), and sample results (Sec. 2.3). We will assume a single component C, although the ideas presented herein could be extended to multiple such components.

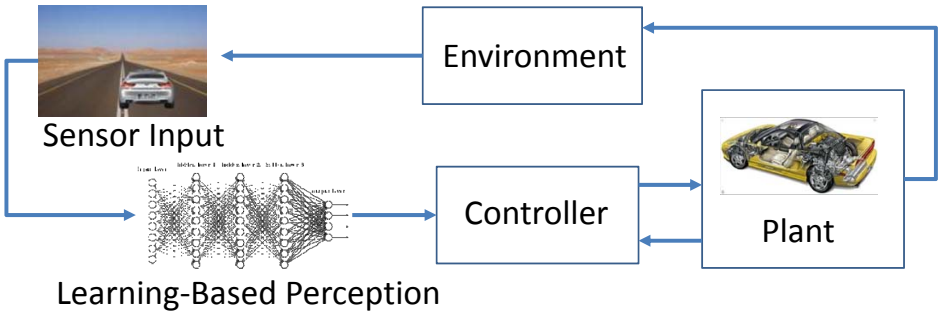

Figure 2: Automatic Emergency Braking System (AEBS) in closed loop. An image classifier based on deep neural networks is used to perceive objects in the ego vehicle's frame of view.

2.1 Example Problem

As an illustrative example, let us consider a simple model of an Automatic Emergency Braking System (AEBS), that attempts to detect objects in front of a vehicle and actuate the brakes when needed to avert a collision. Figure 2 shows the AEBS as a system composed of a controller (automatic braking), a plant (vehicle sub-system under control, including transmission), and an advanced sensor (camera along with an obstacle detector based on deep learning). The AEBS, when combined with the vehicle's environment, forms a closed loop control system. The controller regulates the acceleration and braking of the plant using the velocity of the subject (ego) vehicle and the distance between it and an obstacle. The sensor used to detect the obstacle includes a camera along with an image classifier based on deep neural networks. In general, this sensor can provide noisy measurements due to incorrect image classifications which in turn can affect the correctness of the overall system.

Suppose we want to verify whether the distance between the ego vehicle and a preceding obstacle is always larger than 2 meters. Such a verification requires the exploration of a very large input space comprising the control inputs (e.g., acceleration and braking pedal angles) and the machine learning (ML) component's feature space (e.g., all the possible pictures observable by the camera). The latter space is particularly large — for example, note that the feature space of RGB images of dimension 1000×600px (for an image classifier) contains $256^{1000\times600\times3}$ elements.

In the above example, $S \| E$ is the closed loop system in Fig. 2 where S comprises the deep neural network and the controller, and E comprises everything else. C is the deep neural network used for object detection and classification. The system-level specification Φ can be captured in a metric temporal logic specifying that the distance between the ego vehicle and a preceding obstacle is always larger than 2 meters.

This case study has been implemented in Matlab/Simulink[1] in two versions that use two different Convolutional Neural Networks (CNNs): the Caffe [9] version of AlexNet [12] and the Inception-v3 model created with Tensorflow [14], both trained on the ImageNet database [1]. Further details about this example can be obtained from [5].

2.2 Composition and Abstraction

Our approach rests on the twin notions of *abstraction* and *compositionality* that have been central to much of the advances in formal verification. The high-level idea is to have a *system-level verifier* (or system-level analyzer) that abstracts away the component C while verifying Φ on the resulting abstraction. This

[1] https://github.com/dreossi/analyzeNN

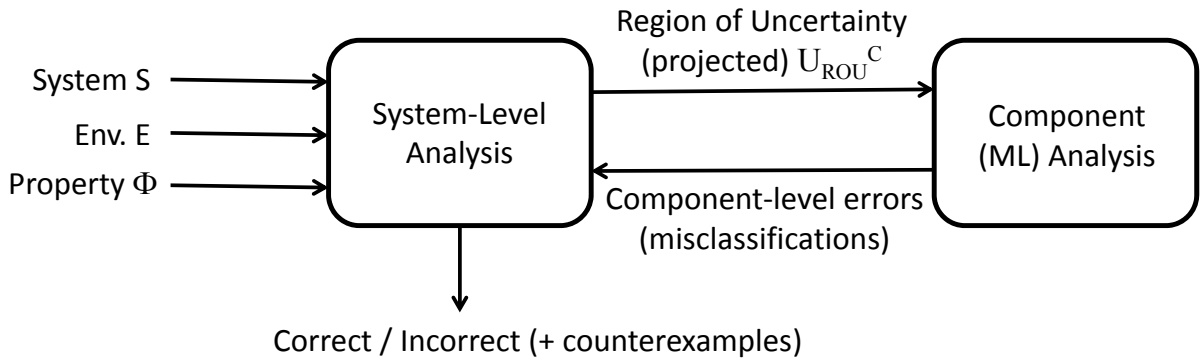

Figure 3: Compositional Verification Approach. A system-level verifier cooperates with a component-level analysis procedure (e.g., adversarial analysis of a machine learning component to find misclassifications).

system-level verifier communicates with a component-level analyzer that identifies interesting component-level behaviors that could lead to violations of the system-level specification Φ. Figure 3 illustrates this approach.

We formalize this approach while trying to emphasize the intuition. Let T denote the set of all possible traces of the composition of the system with its environment, $S||E$. Given a specification Φ, let T_Φ denote the set of traces in T satisfying Φ. Let U_Φ denote the projection of these traces onto the state and interface variables of the environment E. U_Φ is termed as the *validity domain* of Φ, i.e., the set of environment behaviors for which Φ is satisfied. Similarly, the complement set $U_{\neg\Phi}$ is the set of environment behaviors for which Φ is violated.

Our approach works as follows:

1. The System-level Verifier initially performs two analyses with two extreme abstractions of the ML component. First, it performs an *optimistic* analysis, wherein the ML component is assumed to be a "perfect classifier", i.e., all feature vectors are correctly classified. In situations where ML is used for perception/sensing, this abstraction assumes perfect perception/sensing. Using this abstraction, we compute the validity domain for this abstract model of the system, denoted U_Φ^+. Next, it performs a *pessimistic* analysis where the ML component is abstracted by a "completely-wrong classifier", i.e., all feature vectors are misclassified. Denote the resulting validity domain as U_Φ^-. It is expected that $U_\Phi^+ \supseteq U_\Phi^-$.

   Abstraction permits the System-level Verifier to operate on a lower-dimensional search space and identify a region in this space that may be affected by the malfunctioning of component C — a so-called "region of uncertainty" (ROU). This region, U_{ROU}^C is computed as $U_\Phi^+ \setminus U_\Phi^-$. In other words, it comprises all environment behaviors that could lead to a system-level failure when component C malfunctions. This region U_{ROU}^C, projected onto the inputs of C, is communicated to the ML Analyzer. (Concretely, in the context of our example of Sec. 2.1, this corresponds to finding a subspace of images that corresponds to U_{ROU}^C.)

2. The Component-level Analyzer, also termed as a Machine Learning (ML) Analyzer, performs a detailed analysis of the projected ROU U_{ROU}^C. Several options are available for such an analysis. One extreme is to perform an exhaustive search over all possible images (assuming suitable discretization of the space), but this is not tractable for practical examples. Another approach, presented in [5], uses systematic sampling techniques to explore the input space of C. We believe the growing litera-

ture on adversarial analysis of machine learning systems, especially deep neural networks, could be useful in designing the ML analyzer (see, e.g., [7, 17, 15, 16, 4]). Even though a component-level formal specification may not be available, each of these adversarial analyses has an implicit notion of "misclassification." We will refer to these as *component-level errors*.

3. When the Component-level (ML) Analyzer finds component-level errors (e.g., those that trigger misclassifications of inputs whose labels are easily inferred), it communicates that information back to the System-level Verifier, which checks whether the ML misclassification can lead to a violation of the system-level property Φ. If yes, we have found a system-level counterexample. If no component-level errors are found, and the system-level verification can prove the absence of counterexamples, then it can conclude that Φ is satisfied. Otherwise, if the ML misclassification cannot be extended to a system-level counterexample, the ROU is updated and the revised ROU passed back to the Component-level Analyzer.

The communication between the System-level Verifier and the Component-level (ML) Analyzer continues thus, until we either prove/disprove Φ, or we run out of resources.

2.3 Some Results

We have applied the above approach to the problem of *compositional falsification* of cyber-physical systems (CPS) with machine learning components [5]. For this class of CPS, including those with highly non-linear dynamics and even black-box components, one cannot expect to prove system correctness. Instead, simulation-based falsification of temporal logic properties is an approach that has proven effective in industrial practice (e.g., [10, 20]).

In [5], we discuss the application of our approach to the AEBS example discussed in Sec. 2.1 above. We present just a few highlights of our results here, referring the reader to more detailed descriptions in our other papers on the topic [5, 6].

In Figure 4, we illustrate the validity domains obtained with the optimistic and pessimistic analyses as well as the region of uncertainty. Each point in the ROU represents an environment configuration — i.e., a combination of initial velocity of the ego vehicle and initial distance between the ego vehicle and environment vehicle — for which a misclassification could potentially lead to a system-level safety violation.

In Figure 5 we show one result of our analysis for the Inception-v3 deep neural network. This figure shows both correctly classified and misclassified images on a range of synthesized images where (i) the environment vehicle is moved away from or towards the ego vehicle (along z-axis), (ii) it is moved sideways along the road (along x-axis), or (iii) the brightness of the image is modified. These modifications constitute the 3 axes of the figure. Our approach finds misclassifications that do not lead to system-level property violations and also misclassifications that do lead to such violations. For example, Figure 5 shows two misclassified images, one with an environment vehicle that is too far away to be a safety hazard, as well as another image showing an environment vehicle driving slightly on the wrong side of the road, which is close enough to potentially cause a violation of the system-level safety property (of maintaining a safe distance from the ego vehicle).

For further details about this and other results with our approach, we refer the reader to [5, 6].

3 Conclusion

In this note, we discussed the challenge of applying compositional verification methods to a system with components for which formal specification is extremely difficult or even impossible. We presented an approach based on cooperating analyzers involving a system-level verifier communicating information with

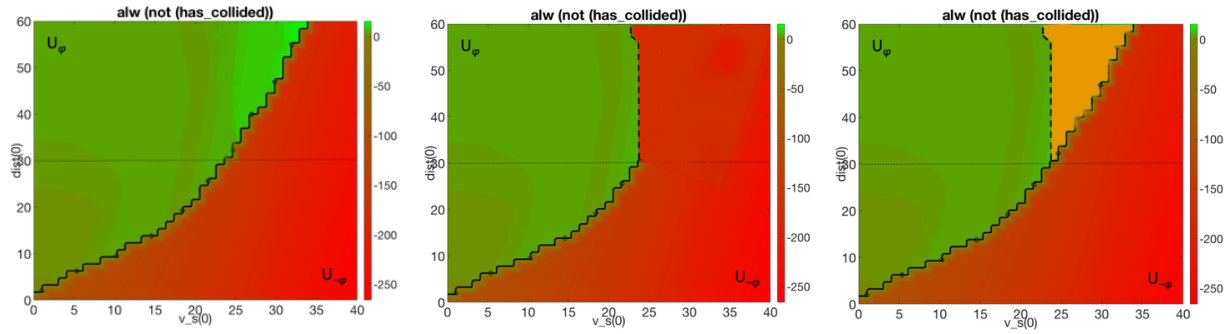

Figure 4: Validity domain for system-level property with different abstractions of ML component. The initial velocity and distance – initial environment configuration – are on the x and y axes respectively. The dotted (horizontal) line is the distance at which the image classifier is active in this design. Green indicates combinations of initial velocity and distance for which the property is satisfied and red indicates combinations for which the property is falsified. Our ML analyzer performs both optimistic (left) and pessimistic (middle) abstractions of the neural network classifier. On the right-most image, the yellow region denotes the region of uncertainty (ROU).

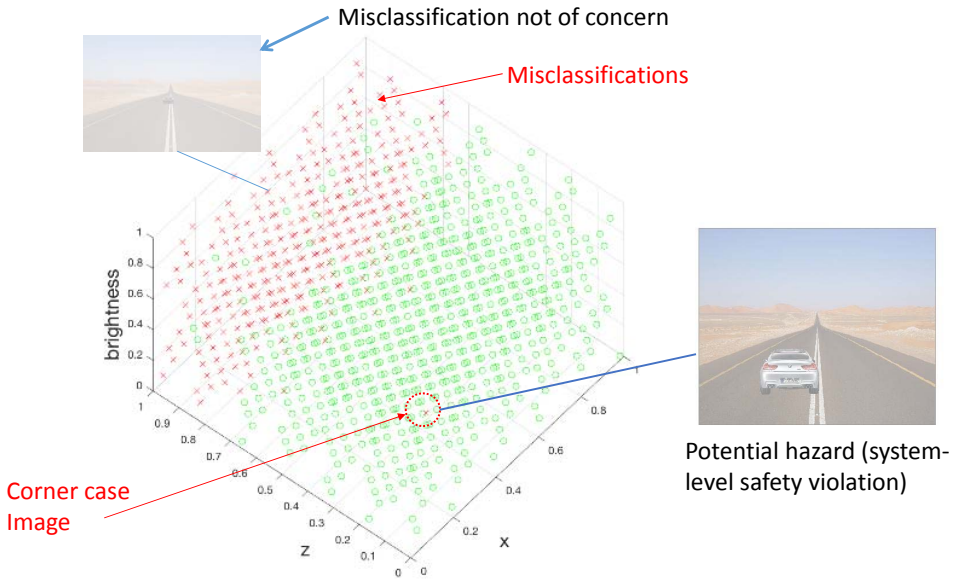

Figure 5: Misclassified Images for Inception-v3 Neural Network (trained on ImageNet with TensorFlow). Red crosses are misclassified images and green circles are correctly classified. Our system-level analysis finds a corner-case image that could lead to a system-level safety violation.

component-level analyzers. Overall, our experience has been that the compositional approach is effective at applying formal verification methods to the analysis of realistic systems with machine learning used in perceptual tasks that are hard to formally specify.

There are several directions for future work (see [19] for details), including a further formalization of the compositional architecture described in this article, as well as applying it more widely to a range of machine learning-based systems where strong guarantees of safety and correctness are required.

Acknowledgments

The author's work is supported in part by NSF grants CNS-1545126 and CNS-1646208. Tommaso Dreossi's contributions to some of the material presented in this article are gratefully acknowledged.

References

[1] Imagenet. http://image-net.org/.

[2] Rajeev Alur and Thomas A. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[3] 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.

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57, 2017.

[5] 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.

[6] Tommaso Dreossi, Alexandre Donzé, and Sanjit A. Seshia. Compositional falsification of cyber-physical systems with machine learning components. *CoRR*, abs/1703.00978, 2017.

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

[8] Geoffrey Hinton et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia Conference, ACMMM*, pages 675–678, 2014.

[10] 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.

[11] Cliff B. Jones. Tentative steps toward a development method for interfering programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 5(4):596–619, 1983.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[14] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[15] 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.

[16] 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.

[17] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 372–387, 2016.

[18] 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.

[19] Sanjit A. Seshia, Dorsa Sadigh, and S. Shankar Sastry. Towards verified artificial intelligence. *CoRR*, abs/1606.08514, 2016.

[20] Tomoya Yamaguchi, Tomoyuki Kaga, Alexandre Donzé, and Sanjit A. Seshia. Combining requirement mining, software model checking, and simulation-based verification for industrial automotive systems. In *Proceedings of the IEEE International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, October 2016.