Chapter 10
Compositional Verification for Autonomous Systems with Deep Learning Components

White Paper

Corina S. Păsăreanu, Divya Gopinath, and Huafeng Yu

10.1 Introduction

Autonomy is increasingly prevalent in many applications, ranging from recommendation systems to fully autonomous vehicles, that require strong safety assurance guarantees. However, this is difficult to achieve, since autonomous systems are large, complex systems, that operate in uncertain environment conditions and often use data-driven, machine-learning algorithms. Machine-learning techniques such as deep neural nets (DNN), widely used today, are inherently unpredictable and lack the theoretical foundations to provide the assurance guarantees needed by safety-critical applications. Current assurance approaches involve design and testing procedures that are expensive and inadequate, as they have been developed mostly for human-in-the-loop systems and do not apply to systems with advanced autonomy.

We propose a compositional approach for the scalable verification of learning-enabled autonomous systems to achieve design-time assurance guarantees. The approach is illustrated in Fig. 10.1. The input to the framework is the design model of an autonomous system (this could be given as, e.g., Simulink/Stateflow or prototype implementation). As the verification of the system as a whole is likely intractable we advocate the use of compositional assume-guarantee verification whereby formally defined *contracts* allow the designer to model and reason about learning-enabled components working side-by-side with the other components in the system. These contracts encode the properties *guaranteed* by the component and

C. S. Păsăreanu (✉) · D. Gopinath
Carnegie Mellon University and NASA Ames, Moffett Field, CA, USA
e-mail: corina.pasareanu@west.cmu.edu

H. Yu
Boeing, Chicago, IL, USA

H. Yu et al. (eds.), *Safe, Autonomous and Intelligent Vehicles*,
Unmanned System Technologies, https://doi.org/10.1007/978-3-319-97301-2_10

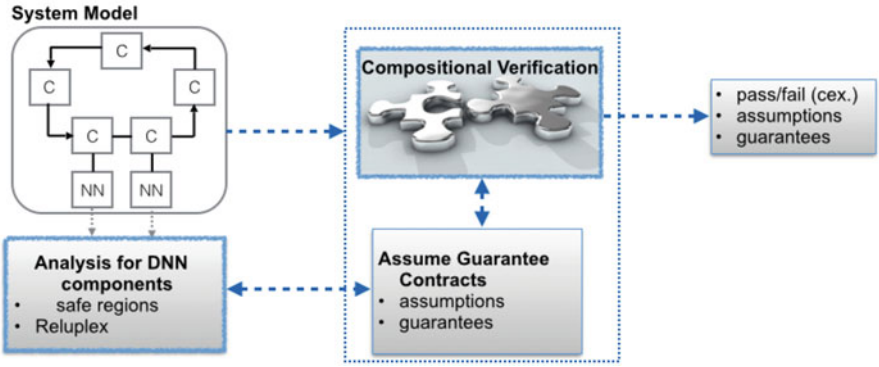

Fig. 10.1 Overview

the environment *assumptions* under which these guarantees hold. The framework will then use compositional reasoning to *decompose* the verification of large systems into the more manageable verification of individual components, which are formally checked against their respective assume-guarantee contracts. The approach enables separate component verification with specialized tools (e.g., one can use software model checking for a discrete-time controller but hybrid model checking for the plant component in an autonomous system) and seamless integration of DNN analysis results.

For DNN analysis, we propose to use clustering techniques to automatically discover *safe regions* where the networks behave in a *predictable* way. The *evidence* obtained from this analysis is *conditional*, subject to constraints defined by the safe regions, and is encoded in the assume-guarantee contracts. The contracts allow us to relate the DNN behavior to the validity of the system-level requirements, using compositional model checking. We illustrate the approach on an example of an autonomous vehicle that uses DNN in the perception module.

10.2 Compositional Verification

Formal methods provide a rigorous way of obtaining strong assurance guarantees of computing systems. There are several challenges to formally modeling and verifying autonomous systems. Firstly, such systems comprise of many *heterogeneous components*; each with different implementations and requirements, which can be addressed best with different verification models and techniques. Secondly, the *state space of such systems is very large*. Suppose we could model all the components of such a system as formally specified (hybrid) models; even ignoring the learning aspect, their composition would likely be intractable. The DNN components make the scalability problem even more serious: for example, the feature space of RGB

1000×600 px pictures for an image classifier used in the perception module of an autonomous vehicle contains $256^{1000\times600\times3}$ elements. Last but not the least, it is not clear how to formally reason about the DNN components as there is no clear consensus in the research community on a *formal definition of correctness for the underlying machine learning algorithms*.

We propose a compositional assume-guarantee verification approach for the scalable verification of autonomous systems where DNN components are working side-by-side with the other components. Compositional verification frameworks have been proposed before to improve the reliability and predictability of CPS [1, 4, 5, 18], but none of these works address systems that include DNN components. Recent work [6] proposes a compositional framework for the analysis of autonomous systems with DNN components. However, that approach addresses *falsification* in such systems and, while that is very useful for debugging, it is not clear how it can be used to provide assurance *guarantees*.

Assume-guarantee reasoning attempts to break up the verification of a large system into the local verification of individual components, using *assumptions* about the rest of the system. The simplest assume-guarantee rule first checks that a component M_1 satisfies a property P under an assumption A (this can be written as $M_1 \models A \rightarrow P$). If the "environment" M_2 of M_1 (i.e., the rest of the system in which M_1 operates) satisfies A (written as $M_2 \models true \rightarrow P$), then we can prove that the whole system composed of M_1 and M_2 satisfies P. Thus we can decompose the global property P into two local assume-guarantee properties (i.e., contracts) $A \rightarrow P$ and A that are expected to hold on M_1 and M_2, respectively. Other, more involved, rules allow reasoning about the circular dependencies between components, where the assumption for one component is used as the guarantee of the other component and vice versa; if the conjunction of the assumptions implies the specification then the overall system guarantees the system-level requirement. Rules that involve circular reasoning use inductive arguments, over time, formulas to be checked, or both, to ensure soundness. Furthermore, the rules can be naturally generalized to reasoning about more than two components and use different notions for property satisfaction such as trace inclusion or refinement checking.

The main challenge with assume-guarantee reasoning techniques is to come up with assumptions and guarantees that can be suitably used in the assume-guarantee rules. This is typically a difficult manual process. Progress has been made on automating assume-guarantee reasoning using learning and abstraction-refinement techniques for iterative building of the necessary assumptions [19]. The original work was done in the context of systems expressed as finite-state automata, but progress has been made in the automated compositional verification for probabilistic and hybrid systems [2, 14], which can be used to model autonomous systems.

Assume-guarantee reasoning can be used for the verification of autonomous systems either by replacing the component with its assume-guarantee specification in the compositional proofs or by using an assume-guarantee rule such as the above to decompose the verification of the systems into the verification of its components. Furthermore, the assume-guarantee specifications can be used to drive

component-based testing and run-time monitoring, in the cases where the design-time formal analysis is not possible, either because the components are too large or they are *adaptive*, i.e. the component behavior changes at run-time (using, e.g., reinforcement learning).

10.3 Analysis for Deep Neural Network Components

Deep neural networks (DNNs) are computing systems inspired by the biological neural networks that constitute animal brains. They consist of neurons (i.e., computational units) organized in many layers. These systems are capable of *learning* various tasks from *labeled examples* without requiring task-specific programming. DNNs have achieved impressive results in computer vision, autonomous transport, speech recognition, social network filtering, bioinformatics, and many other domains and there is increased interest in using them in safety-critical applications that require strong assurance guarantees. However, it is difficult to provide such guarantees since it is known that these networks can be easily fooled by adversarial perturbations: minimal changes to correctly-classified inputs, that cause the network to misclassify them. For instance, in image-recognition networks it is possible to add a small amount of noise (undetectable by the human eye) to an image and change how it is classified by the network.

This phenomenon represents a safety concern, but it is currently unclear how to measure a network's robustness against it. To date, researchers have mostly focused on efficiently finding adversarial perturbations around select individual input points. The goal is to find an input x' as close as possible to a known input x such that x' and x are labeled differently. Finding the optimal solution for this problem is computationally difficult, and so various approximation approaches have been proposed. Some approaches are *gradient based* [7, 8, 20], whereas others use optimization techniques [3]. These approaches have successfully demonstrated the weakness of many state-of-the-art networks; however, these approaches operate on individual input points, and it is unclear how to apply them to large input domains, unless one does a brute-force enumeration of all input values which is infeasible for most practical purposes. Furthermore, because they are inherently incomplete, these techniques cannot even provide any guarantees around the few selected individual points. Recent approaches tackle neural network verification [10, 13] by casting it as an SMT solving problem. Still, these techniques operate best when applied to individual points and further do not have a well-defined rationale to select meaningful regions around inputs within which the network is expected to behave consistently.

In [9], we developed a DNN analysis to automatically discover *input regions* that are likely to be robust to adversarial perturbations, i.e. to have the same true label, akin to finding likely invariants in program analysis. The technique takes inputs

with known true labels from the training set and it iteratively applies a clustering algorithm [12] to obtain small groups of inputs that are close to each other (with respect to different distance metrics) and share the same true label. Each cluster defines a *region* in the input space (characterized by the centroid and radius of the cluster). Our hypothesis is that for regions formed from dense clusters, the DNN is well-trained and we expect that all the other inputs in the region (not just the training inputs) should have the same true label. We formulate this as a safety check and we verify it using off-the-shelf solvers such as Reluplex [13]. If a region is found to be *safe*, we provide *guarantees w.r.t all points within that region*, not just for individual points as in previous techniques.

As the usual notion of safety might be too strong for many DNNs, we introduce the concept of *targeted safety*, analogous to targeted adversarial perturbations [7, 8, 20]. The verification checks *targeted safety* which, given a specific incorrect label, guarantees that no input in the region is mapped by the DNN to that label. Therefore, even if in that region the DNN is not completely robust against adversarial perturbations, we give guarantees that it is safe against specific targeted attacks.

As an example, consider a DNN used for perception in an autonomous car that classifies the images of a semaphore as red, green, or yellow. We may want to guarantee that the DNN will never classify the image of a green light as a red light and vice versa but it may be tolerable to misclassify a green light as yellow, while still avoiding traffic violations.

The safe regions discovered by our technique enable characterizing the input–output behavior of the network over partitions of the input space, which can be encoded in the assume-guarantee specifications for the DNN components. The regions will define the conditions (assumptions), and the guarantees will be that all the points within the region will be assigned the same labels. The regions could be characterized as geometric shapes in Euclidean space with centroids and radii. The conditions would then be in terms of standard distance metric constraints on the input attributes. For instance, all inputs within a Euclidean distance r from the centroid cen of the region would be labeled l by the network.

Note that the verification of even simple neural networks is an NP-complete problem and is very difficult in practice. Focusing on clusters means that verification can be applied to small input domains, making it more feasible and rendering the approach as a whole more scalable. Further, the verification of separate clusters can be done in parallel, increasing scalability even further.

In [9] we applied the technique on the MNIST dataset [16] and on a neural network implementation of a controller for the next-generation Airborne Collision Avoidance System for unmanned aircraft (ACAS Xu) [11], where we used Reluplex for the safety checks. For these networks, our approach identified multiple regions which were completely safe as well as some which were only safe for specific labels. It also discovered adversarial examples which were confirmed by domain experts. We discuss the ACAS Xu experiments in more detail below.

10.3.1 ACAS Xu Case Study

ACAS X is a family of collision avoidance systems for aircraft which is currently under development by the Federal Aviation Administration (FAA) [11]. ACAS Xu is the version for unmanned aircraft control. It is intended to be airborne and receive sensor information regarding the drone (the

ownship) and any nearby intruder drones, and then issue horizontal turning advisories aimed at preventing collisions. The input sensor data includes:

- ρ: distance from ownship to intruder;
- θ: angle of intruder relative to ownship heading direction;
- ψ: heading angle of intruder relative to ownship heading direction;
- v_{own}: speed of ownship;
- v_{int}: speed of intruder;
- τ: time until loss of vertical separation; and
- a_{prev}: previous advisory.

The five possible output actions are as follows: Clear-of-Conflict (COC), Weak Right, Weak Left, Strong Right, and Strong Left. Each advisory is assigned a score, with the lowest score corresponding to the best action. The FAA is currently exploring an implementation of ACAS Xu that uses an array of 45 deep neural networks. These networks were obtained by discretizing the two parameters, τ and a_{prev}, and so each network contains five input dimensions and treats τ and a_{prev} as constants. Each network has 6 hidden layers and a total of 300 hidden ReLU activation nodes. We were supplied a set of cut-points, representing valid important values for each dimension, by the domain experts [11]. We generated a set of 2,662,704 inputs (cartesian product of the values for all the dimensions). The network was executed on these inputs and the output advisories (labels) were verified. These were considered as the inputs with known labels for our experiments.

We were able to prove safety for 177 regions in total (125 regions where the network was completely safe against mis-classification to any label and 52 regions where the network was safe against specific target labels). An example of the safety guarantee is as follows:

$$\forall\, x \in |x - \{0.19, 0.31, 0.28, 0.33, 0.33\}|_{L_1} \leq 0.28 \quad \Rightarrow \quad label(x) = COC \tag{10.1}$$

Here {0.19,0.31,0.28,0.33,0.33} are the normalized values for the five input attributes (ρ,θ,ψ,v_{own},v_{int}) corresponding to the centroid of the region and 0.28 is the radius. The distance is in the Manhattan distance metric (L1). The contract states that under the condition that an input lies within 0.28 distance from the input vector {0.19,0.31,0.28,0.33,0.33}, the network is guaranteed to mark the action for it as COC which is the desired output.

Our analysis also discovered adversarial examples of interest, which were validated by the developers. Figure 10.2 illustrates such an example for ACAS Xu.

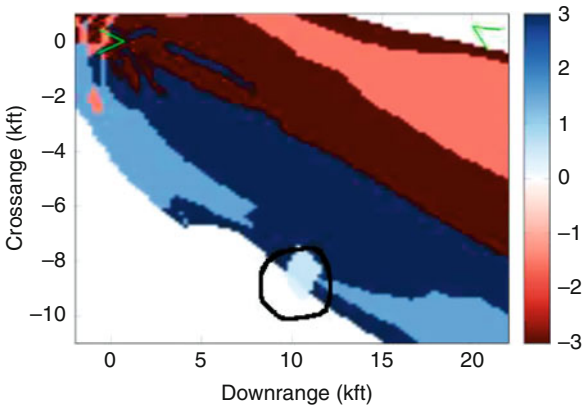

Fig. 10.2 Inputs highlighted in light blue are mis-classified as strong right instead of COC. Crossrange $= \rho \cdot \sin(\theta)$, Downrange $= \rho \cdot \cos(\theta)$

The safety contracts obtained with the region analysis can be used in the compositional verification of the overall autonomous systems, which can be performed with standard model checkers.

10.4 Example

We illustrate our compositional approach on an example of an autonomous vehicle. The platform includes learning components that allow it to detect other vehicles and drive according to traffic regulations; the platform also includes reinforcement learning components to evolve and refine its behavior in order to learn how to avoid obstacles in a new environment.

We focus on a subsystem, namely an automatic emergency breaking system, illustrated in Fig. 10.3. It has three components: the *BreakingSystem*, the *Vehicle* (which, to simplify the presentation, we assume it includes both the autonomous vehicle and the environment), and a *perception module* implemented with a DNN; there may be other sensors (radar, LIDAR, GPS) that we abstract away here for simplicity. The breaking system sends signals to the vehicle to regulate the acceleration and breaking, based on vehicle velocity, distance to obstacles and traffic signals. The velocity information is provided as a feedback from the plant, the distance information is obtained from sensors, while the information about traffic lights is obtained from the perception module. The perception module acts as a classifier over images captured with a camera. Such systems are already employed today in semi-autonomous vehicles where adaptive cruise controllers or lane keeping assist systems rely on image classifiers providing input to the software controlling electrical and mechanical subsystems [6]. Suppose we want to check that the system satisfies the following *safety property*: the vehicle will not enter an intersection if the traffic light at the intersection turns red.

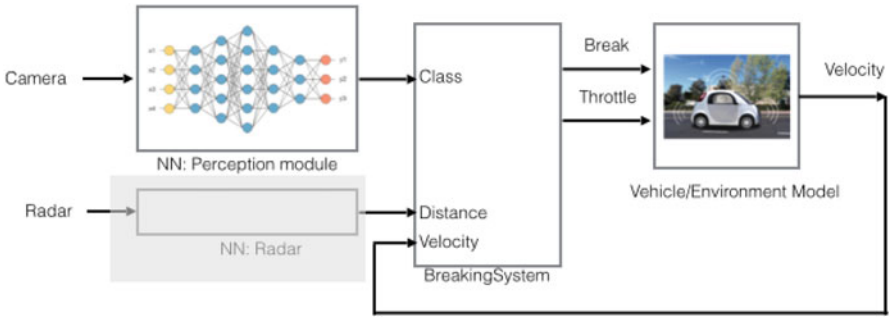

Fig. 10.3 Example

We write the system as the composition: $BreakingSystem||Vehicle||NN$. Each component has an interface that specifies its input and output variables (ports), and their parallel composition is formed by connecting components via ports. We write the property as follows (using Linear Temporal Logic, LTL, assuming discrete time): globally (G) if the semaphore (input image x) is red, then eventually (F), within 3 s, the velocity becomes 0:

$$P :: \quad G((x = red) \quad \Rightarrow \quad F_{T<3s}(velocity = 0))$$

In practice, we would also need to encode in P the assumption that the distance to traffic light is less than some threshold, but we simplify here to ease the presentation. We are thus interested in checking that the system satisfies property P, written as $S \models P$. We decompose the system into two subsystems: $M_1 = BreakingSystem||Vehicle$ and $M_2 = NN$ and define two assume-guarantee contracts C_1 and C_2 for the two subsystems. Suppose (part of) the contract for M_1 is:

$$C_1 :: \quad G((Class = red) \quad \Rightarrow \quad F_{T<3s}(velocity = 0))$$

The contract states that *assuming* the input (Class) to the subsystem M_1 is red then the vehicle is *guaranteed* to stop in at most three time units. We can further decompose the verification of M_1 into the separate verification of its components using additional contracts and perform component-wise verification. It remains to formally characterize the input–output behavior of the DNN in a contract that can be used in the compositional proofs. This is a difficult problem because DNN are known to be vulnerable to adversarial perturbations [15, 20]: a small perturbation added to an image that shows a red semaphore might lead the NN misclassifying it as having $Class = green$.

To address the problem, we use clustering over the training set (see Sect. 10.3) to automatically find regions where the network is likely to be robust to adversarial perturbations. The result is a finite set $\mathcal{R}$ of well-defined regions, where a region $\rho \in \mathcal{R}$ is characterized by a pair (c, r); c is the centroid and r is the radius of the region. We then use a verification tool (such as Reluplex) to check that, *for all*

inputs x within each region, the NN classifies them to the same label as that of known inputs (and of c):

$$C_\rho :: \quad |x - c| < r \quad \Rightarrow \quad Label(x) = Label(c)$$

The training data available and the amount of noise could impact the validity of the check. In such cases we may need to refine the contracts to include Bayesian estimates of uncertainty [17]. Let $Uncert(x)$ denote the uncertainty in the output of the NN for an input x. We can then refine the contract to check that the label is as expected *and* the uncertainty level is below a threshold. The DNN's *safety contract* C_2 could then be the union of all the constraints of the form C_ρ that are proved valid.

We are now ready to perform the compositional proof: if $M_1 \models C_1$ and $M_2 \models C_2$ and furthermore $C_1 \wedge C_2 \Rightarrow P$, it follows that $M_1 || M_2 \models P$; thus, we prove that the whole system satisfies the property, without composing its (large) state space. This proof can be performed with standard model checkers.

10.4.1 Run-Time Monitoring and Control

We note that the evidence we obtain from the analysis is *conditional*; we can only prove that the property holds for the region contracts that we found to be safe. The information encoded in the contract assumptions will need to be used to synthesize *run-time guards* that monitor inputs that fall outside the conditions and instruct the system to take appropriate, fail-safe actions. Note also that this compositional approach enables separate verification of individual components: we can thus replace some of the verification tasks for individual components with testing or simulation, which will increase scalability but will give only empirical guarantees.

Furthermore, if the system contains *adaptive* components, the verification of those components can be done at runtime, whereas the static components only need to be checked once, at design time. Adaptive learning-enabled components pose additional challenges over time. We can again use model uncertainty to identify situations in which the adaptive learning-enabled system is not confident about its decisions, and take appropriate actions in such cases.

10.5 Conclusion

We presented a compositional approach for the verification of autonomous systems. The approach uses assume-guarantee reasoning for scalable verification and can naturally integrate reasoning about the learning-enabled components in the system. We are working on evaluating the proposed approach on various simulation and real

autonomous platforms, including self-driving cars (discussed briefly in Sect. 10.4), autonomous quadcopters, and airplanes. These case studies cover perception, decision making, control and actuation of autonomous systems, and they include safety-critical cyber-physical components as well as DNN components.

References

1. S. Bak, S. Chaki, Verifying cyber-physical systems by combining software model checking with hybrid systems reachability, in *2016 International Conference on Embedded Software, EMSOFT 2016*, Pittsburgh, Pennsylvania, USA, October 1–7, 2016 (2016), pp. 10:1–10:10
2. S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C.S. Pasareanu, A. Podelski, T. Strump, Assume-guarantee abstraction refinement meets hybrid systems, in *Proceedings of Hardware and Software: Verification and Testing - 10th International Haifa Verification Conference, HVC 2014*, Haifa, Israel, November 18–20, 2014 (2014), pp. 116–131
3. N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in *Proceedings of 38th IEEE Symposium on Security and Privacy* (2017)
4. C. Chilton, B. Jonsson, M.Z. Kwiatkowska, An algebraic theory of interface automata. Theor. Comput. Sci. **549**, 146–174 (2014)
5. C. Chilton, B. Jonsson, M.Z. Kwiatkowska, Compositional assume-guarantee reasoning for input/output component theories. Sci. Comput. Program. **91**, 115–137 (2014)
6. T. Dreossi, A. Donzé, S.A. Seshia, Compositional falsification of cyber-physical systems with machine learning components, in *Proceedings of NASA Formal Methods - 9th International Symposium, NFM 2017*, Moffett Field, CA, USA, May 16–18, 2017 (2017), pp. 357–372
7. R. Feinman, R.R. Curtin, S. Shintre, A.B. Gardner, Detecting adversarial samples from artifacts. Technical Report (2017). http://arxiv.org/abs/1703.00410
8. I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples. Technical Report (2014). http://arxiv.org/abs/1412.6572
9. D. Gopinath, G. Katz, C.S. Pasareanu, C. Barrett, Deepsafe: a data-driven approach for checking adversarial robustness in neural networks, in *Proc. ATVA'18* (2017). https://arxiv.org/abs/1710.00486
10. X. Huang, M. Kwiatkowska, S. Wang, M. Wu, Safety verification of deep neural networks, in *Proceedings of 29th International Conference on Computer Aided Verification (CAV)* (2017), pp. 3–29
11. K. Julian, J. Lopez, J. Brush, M. Owen, M. Kochenderfer, Policy compression for aircraft collision avoidance systems, in *Digital Avionics Systems Conference (DASC)* (2016), pp. 1–10
12. T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, A.Y. Wu, An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 881–892 (2002)
13. G. Katz, C. Barrett, D. Dill, K. Julian, M. Kochenderfer, Reluplex: an efficient SMT solver for verifying deep neural networks, in *Proceedings of the 29th International Conference on Computer Aided Verification (CAV)* (2017), pp. 97–117
14. A. Komuravelli, C.S. Pasareanu, E.M. Clarke, Assume-guarantee abstraction refinement for probabilistic systems, in *Proceedings of Computer Aided Verification - 24th International Conference, CAV 2012*, Berkeley, CA, USA, July 7–13, 2012 (2012), pp. 310–326
15. A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, 2016. Technical Report. http://arxiv.org/abs/1607.02533
16. Y. LeCun, C. Cortes, C.J.C. Burges, The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/
17. Y. Li, Y. Gal, Dropout inference in Bayesian neural networks with alpha-divergences, in *International Conference on Machine Learning* (2017), pp. 2052–2061

18. J. Li, P. Nuzzo, A.L. Sangiovanni-Vincentelli, Y. Xi, D. Li, Stochastic assume-guarantee contracts for cyber-physical system design under probabilistic requirements. CoRR (2017). http://arxiv.org/abs/1705.09316
19. C.S. Pasareanu, D. Giannakopoulou, M.G. Bobaru, J.M. Cobleigh, H. Barringer, Learning to divide and conquer: applying the l* algorithm to automate assume-guarantee reasoning. Formal Methods Syst. Des. **32**(3), 175–205 (2008)
20. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, 2013. Technical Report. http://arxiv.org/abs/1312.6199