Confidence Monitoring and Composition for Dynamic Assurance of Learning-Enabled Autonomous Systems*

Position paper

Ivan Ruchkin, Matthew Cleaveland, Oleg Sokolsky, and Insup Lee

University of Pennsylvania

Abstract. Design-time approaches to safety assurance for autonomous systems are limited because they must rely on assumptions about the behaviors of learned components in previously unseen environments. These assumptions may be violated at run time, thus invalidating the guarantees produced at design time. To overcome this limitation, we propose to complement design-time assurance with run-time monitoring that calculates the confidence that those assumptions are satisfied and, therefore, design-time guarantees continue to hold. As the first step in our vision, we elicit the logical relationship between assumption violations and safety violations. Then, we develop a probabilistic confidence monitor for each design-time assumption. Finally, we compose these assumption monitors based on their logical relation to safety violations, producing a system-wide assurance monitor. Our vision is illustrated with a case study of an autonomous underwater vehicle that performs pipeline inspection.

1 Introduction and motivation

A clear technological trend emerged in the past decade: safety-critical systems, such as cars, airplanes, and other vehicles, are becoming increasingly autonomous. The assurance for these systems is becoming increasingly complex for two reasons. First, safety-critical systems often operate in environments that have not been anticipated at design time. The systems now have to understand these newly encountered environments and respond to them in a safe manner. Second, these systems rely on machine learning in various modules, including perception, planning, and control, yielding black-box implementations with high-dimensional inputs. Traditionally, design-time guarantees of safety require modeling the system and its environment, and models are carefully validated to ensure sufficient fidelity. However, the models of the learning-enabled components as well as detailed environment models cannot be fully validated at design time.

We certainly cannot abandon design-time assurance, but we can *complement* it with run-time assurance techniques. Several arguments for run-time assurance

* This work is supported in part by the Air Force Research Laboratory and the Defense Advanced Research Projects Agency as part of the Assured Autonomy program under Contract No. FA8750-18-C-0090

articulated the need for comprehensive self-assessment of mission progress by an autonomous system [2, 9]. It is typical to deploy safety monitoring systems across the system, supported by rigorous systems of monitor generation [6, 10]. Such safety monitors tend to be defined in terms of thresholds over the system's state variables, with a fixed tolerance margin. For example, a collision warning monitor may, in effect, say "you are close to an obstacle and moving too fast towards it." The confidence in such warnings is derived from the pre-determined margin and does not account for unpredictable situations and noisy/missing inputs. This confidence can be improved by predictive safety monitoring [1, 12], which takes into account future behaviors, and combining monitors with state estimators [15], which account for perception/control uncertainties within a limited time horizon.

The challenge is that such advanced monitors also rely on the very models of the system dynamics and perception that we are trying to validate in the first place. The drawback we see in these monitoring approaches is that they do not take into account all of the hard work that went into design-time assurance of those models; instead, they try to estimate the confidence in the mission progress directly from the observations. In this paper, we are advocating for a different approach to run-time assurance, one that builds upon modern model-based verification technologies. The question we are asking at run time is: based on what we observe, how confident are we that our verification guarantees still imply that the system will succeed in its mission?

Our approach investigates, at design time, how the design-time verification guarantees can be invalidated at run time. The process of eliciting these potential problems is similar to the traditional hazard analysis [11, 5], but instead of hazards introduced by component failures, we consider the hazards introduced by the limitations of design-time verification technologies. Specifically, we focus on the *assumptions* that are by necessity made at design time to enable verification. If an assumption is violated, the guarantee may become invalid as well. We consider examples of such assumptions below in the running example.

We note that a violated assumption does not, by itself, mean that the mission is compromised. However, as long as the assumptions are not violated, we know that design-time guarantees hold. Therefore, if we identify all the assumptions subject to violation, our confidence in them can be used as a *lower bound on the probability* that the system's mission is on track.

Illustrative example. Consider a simple example of an autonomous underwater vehicle (AUV) that inspects a pipeline on an ocean floor. This example was used as a case study in the Assured Autonomy program funded by DARPA.

Figure 1 illustrates a typical mission of follow the pipe while avoiding an obstacle. The mission specification is as follows: *If no obstacles ahead are present, the AUV should follow the pipeline at a constant distance that allows it to obtain the best resolution of the sonar image. When an obstacle is encountered, the AUV needs to maintain a specified separation from the obstacle without losing the pipeline.* The AUV is equipped with a front-looking sonar to detect obstacles and a side-looking synthetic-aperture sonar to track the pipeline. In order to perform the mission autonomously, the AUV is also equipped with three learning-

enabled components. First, an *obstacle detector* discovers obstacles in the front sonar image and calculates the distance to the nearest obstacle, as well as its size. Second, a *perception component* detects the pipeline in the side sonar image and calculates the distance to the pipeline and the vehicle's heading relative to the pipeline. Finally, a *neural-network controller* takes in the relative heading, the distance to the pipeline, and the obstacle information, and calculates control commands for the actuators on the vehicle.

We assume a scenario where design-time assurance for such a mission is provided by formal closed-loop verification. In our case study, we modeled the AUV and its environment as a hybrid system and used the tool Verisig [7] to perform reachability analysis of the system model and verify that the mission requirements are satisfied. Such analysis is subject to a number of assumptions. In particular, we had to assume that the physical dynamics of the AUV have been accurately captured by the model; that is, the response of the vehicle to an actuation command is within a certain error bound of the model's prediction. Further, the perception component, itself implemented using a neural network, was too complex to be included in the model for verification. Instead, we assumed that it always provides an accurate readings — also up to an error bound — of the AUV's distance and heading relative to the pipeline. These assumptions are likely to be violated at run time.

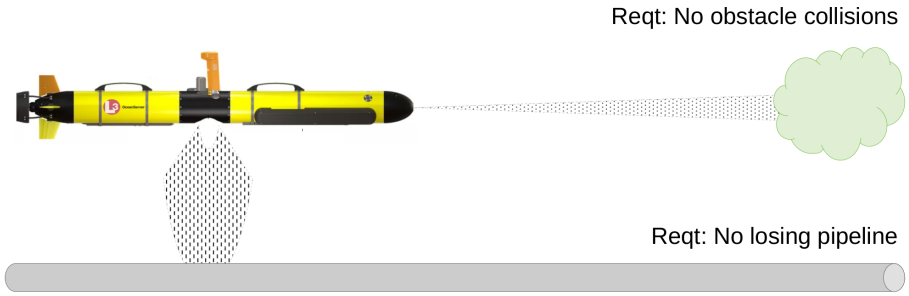

Fig. 1. Requirements for underwater pipeline inspection with an AUV.

2 Problem statement and challenges

We aim to address the following problem: given run-time observations of the system's operation, determine the level of confidence that the guarantees of satisfaction of the safety requirements, established at design time, still hold. A low confidence in these guarantees does not necessarily mean that a violation of safety requirements is imminent. However, a probabilistic interpretation of this confidence would serve as a lower bound on the probability that the mission is

safe. To make this confidence useful, however, we should make the gap between our estimate and the probability of being safe as small as possible.

Revisiting our AUV example, we intend to monitor the operation of learning-enabled components that process the readings from the forward and side sonars and compute control commands, as well as how the vehicle responds to the control commands. From these observations, we intend to compute an estimate of the probability that the design-time verification results apply; namely, that the AUV will neither lose the pipeline nor collide with an obstacle.

Verification results may become invalid if the verification models were inaccurate or if assumptions used in constructing these models are violated at run time. In the case of the AUV, some possible causes of invalid guarantees are that (1) the sonars are less reliable that we expected (e.g., because the water has more dust particles, which degrade sonar images) and (2) the vehicle responds to control commands differently that expected (e.g., because there is a strong current). In either of those cases, the AUV may collide with an obstacle despite the safety proven by design-time verification. Thus, our goal is to estimate our confidence that neither of these two causes (nor other possible ones) are currently applicable.

This estimation faces two major challenges. First, as mentioned above, a violation of assumptions does not necessarily imply a safety violation. In order for the confidence estimate to be useful, the connection should be as tight as possible, excluding assumption violations that currently do not affect safety. Second, both safety requirements and assumptions typically refer to the actual relationship between the system and its environment (e.g., the actual distance from the obstacle). But at run time, we can assess this relationship only indirectly, through the sensing and perception of the vehicle. Quantifying uncertainties in sensors and perception components is therefore critical to the accuracy of our confidence estimate, and more than one component may be involved in this assessment.

Our focus on violations of design-time assumptions allows us to break the problem into two sub-problems and address the challenges in a systematic way.

1. The first sub-problem is to determine how assumption violations influence requirement violations and elicit a logical relationship that takes only relevant assumptions into account.
2. The second sub-problem is to calculate a confidence measure for each assumption, reflecting the probability that the assumption holds, and to compose these confidence measures into an overall assurance measure for the current mission, according to the logical relationship in the first sub-problem.

Addressing these two sub-problems in a coordinated fashion forms the core of our approach, which we discuss in the next section.

3 Outline of the approach

We consider formal verification as a design-time assurance technique. Verification — in our example, closed-loop reachability analysis — is typically performed on

a model of the system and its environment, constructed in a modular fashion. Each module represents either a particular component in the system or a part of the system environment. Unavoidably, models rely on assumptions about the system behavior, which stem either from insufficient knowledge about the actual system or its operating environment, or from simplifications needed to scale up the verification. If these assumptions are violated, the verification results may not hold for the actual system. In our experience, modeling assumptions reflect the modularity of the verification model: each assumption concerns the operation of a component in the system, or some interaction of a component with other components or the system environment. This observation allows us to monitor for assumption violations by focusing on individual components or specific interactions.

As discussed at the end of Section 2, we decompose the problem of computing the confidence estimate into two sub-problems. One is a problem of establishing a relationship between assumption violations and violations of system requirements. We elicit this relationship using the *assumption effect analysis*, establishing a logical structure for composing confidence monitors of individual assumptions into a system-wide *assurance monitor*. The other is a problem of developing confidence monitors for the assumptions based on observed behaviors of system components. Produced by our *probabilistic analysis*, each of the assumption monitors outputs a logical verdict **v** (i.e., whether the assumption is satisfied), as well as a level of confidence **c** in the verdict. Both of these sub-problems are addressed at design time. At run time, assumption monitors are deployed alongside the components they monitor. Confidence outputs of the assumption monitors are fed into the assurance monitor, which outputs the overall confidence measure. Figure 2 illustrates our overall approach.

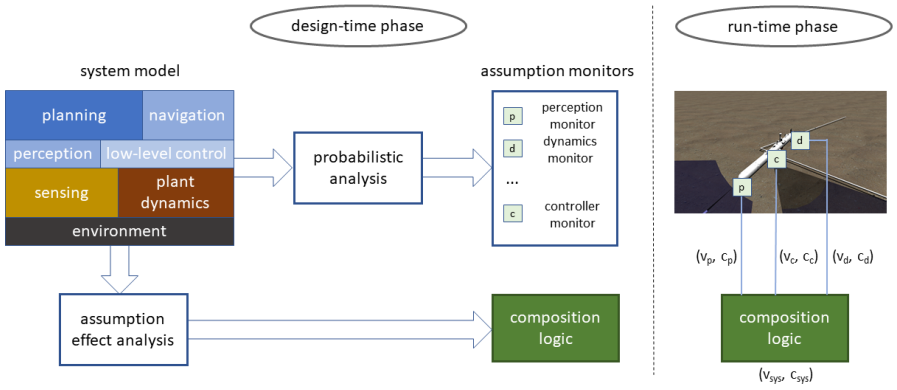

Fig. 2. Overview of the confidence composition approach.

3.1 Logical structure of the assurance monitor

The first step in designing an assurance monitor is to establish a relationship between design-time modeling assumptions and verification guarantees. We capture this relationship as a logical formula over Boolean variables that represent the satisfaction of each assumption. Naively, we can always represent this relationship as a conjunction over all assumptions. That is, verification guarantees hold only if all assumptions are satisfied. This formula would often be too conservative, resulting is a much lower confidence than necessary. Some assumptions may be relevant only to certain safety properties and in certain modes of operation; hence, by making this relation explicit in the formula, we can build a more accurate assurance monitor and thus tightening the lower bound on the probability of mission success.

Assumption effect analysis. In order to identify assumptions and analyze the effects of their violations on safety guarantees, we follow a process similar to traditional hazard analysis [11, 5]. Hazard analysis is an important step in the design of safety-critical system that identifies potential failures in system components and reasons about their effect on system safety and mitigations against these failures. Similarly, our *assumption effect analysis* determines which component models and which assumptions need to be monitored — and how. To perform this analysis, we consider each assumption in turn and analyze the effect of its violation on the system behavior. Ultimately, we identify the logical condition under which violation of the assumption can lead to the safety violation.

Example: assumption effect analysis for AUV missions. Consider a subset of assumptions that are made during the verification of the AUV model:

A1 The current true state is the set of initial states for which verification passed
A2 Vehicle behavior is consistent with the dynamics model used in verification
A3 Obstacle detector has bounded error on the obstacle distance
A4 Perception module has bounded errors on the range/heading to the pipeline
A5 No false-negative obstacle detections
A6 No false-positive obstacle detections

Assumptions A1, A2, and A4 always need to be satisfied. However, if an obstacle is currently detected, assumption A5 is irrelevant. This is because A5 deals with false negative detections of obstacles, but we would currently have a positive detection. Conversely, if an obstacle is not detected, assumption A6 would be irrelevant; furthermore, assumption A3 would also be irrelevant, because the distance to the obstacle is not used in computing the control command. As a result of this analysis, we obtain the following logical specification for the assurance monitor in terms of assumption monitors:

$$(\neg obstDet \wedge S_1) \vee (obstDet \wedge S_2),$$

where *obstDet* is a Boolean flag indicating that the obstacle is detected, $S_1 = A1 \wedge A2 \wedge A4 \wedge A6$ is the set of assumptions relevant in the absence of detected obstacles, and $S_2 = A1 \wedge A2 \wedge A3 \wedge A4 \wedge A5$ is the set of assumptions relevant in the presence of detected obstacles.

3.2 Assumption monitors

Most of the assumptions we encounter in practice are either associated with a particular system component or can be trivially decomposed into component-based sub-assumptions. For example, assumption A1, which concerns the physical state of the vehicle, naturally decomposes into an assumption A1o, which constrains the true distance from an obstacle and its size and is associated with the forward sonar and the obstacle detector — and an assumption A1p, which constrains the true distance from the pipeline and its heading and is associated with the side sonar and the pipeline detector. Such sub-assumptions are often stochastically independent, allowing us to monitor assumptions associated with each component independently of other components and compose the outcomes based on the logical structure elicited above. A single component may be associated with multiple assumptions. For example, assumptions A1o, A3, A5, and A6 are all associated with the same component — the obstacle detector.

Many assumptions can be specified as assertions over the true values of physical variables (e.g., the distance to an obstacle) and the observable values produced by a component. For example, if DT is the true distance from an obstacle, DO is the observed distance from the obstacle, and $obstDet$ is the flag denoting an obstacle detection, then assumption A3 is $|DT - DO| \leq \epsilon$ and assumption A6 is $obstDet \implies DT < \delta$, for some positive constants ϵ and δ. Since DT is not observable directly, these assumptions can be ascertained only in a probabilistic fashion, with a measure of confidence representing the probability that the assumption is indeed satisfied.

The assumptions associated with one component need to be monitored together, because they are almost always correlated. For example, if the obstacle detector encounters a blurry sonar image, due to murky water or sensor noise, the detector may suffer a false positive or false negative, and the distance projection it produces is likely to be meaningless as well. To account for such dependencies, we create a *probabilistic confidence model* for each component, the nature of which depends on the circumstances. Generally, a broad range of state estimators, such as Kalman and particle filters, and probabilistic models, such as Bayesian nets, can be used for this purpose. In some cases, a component may already have a monitor that produces a confidence estimate in its correct operation. For example, neural network classifiers often produce a confidence in the classification outcome. In this case, our probabilistic confidence model would relate this value to confidence estimates of related assumptions. In other cases, e.g., to check whether a dynamics model is valid, we may need to apply a statistical detection method that would yield a distribution over the unknown variables.

For our AUV case study, we concentrated on the assumptions about vehicle dynamics and perception components. For the vehicle dynamics, we plan to use a lightweight technique for model invalidation [3], which applies classification-based techniques to conclude that observed responses of the vehicle to a sequence of actuator commands do not comply with a dynamics model. The technique provides us with a level of confidence, which, if properly calibrated, corresponds to a probability that the assumed model is not valid. For monitoring percep-

tion, we will apply our recent work on confidence calibration for neural network classifiers [8], which would allow us to derive confidence in perception outputs.

In general, we would require a taxonomy of approaches to produce calibrated confidence for different kinds of assumptions by monitoring a variety of system components. For now, it remains a part of our future work.

3.3 Composition of assumption monitors

As discussed in Section 3.1, the outcome of the assumption effect analysis is a logical formula that describes the composition of assumption monitors to obtain the assurance monitor for the whole system. However, we are composing monitors that produce not just a Boolean-valued verdict, but also a value that describes the confidence in the verdict. The challenge is to perform this composition in a way that the resulting assurance monitor provides a confidence value in the formula turning into "true". In particular, since the assumptions in sub-formulas f_1 and f_2 are neither independent nor mutually exclusive, the composition is not as simple as replacing $P(f_1 \wedge f_2)$ with $P(f_1) \cdot P(f_2)$ and $P(f_1 \vee f_2)$ with $P(f_1) + P(f_2)$. Instead, the composition should be based on the dependencies between the monitors.

To address this challenge, we plan to extend our recent work on logical composition of stochastic detectors [13], which are run-time monitors that deliver Boolean verdicts with probabilistic guarantees in terms of false positive and false negative rates. This framework allows us to develop composite detectors that are based on formulas of a temporal logic — and derive probabilistic estimates of their performance. Since assurance monitors also perform logic-based composition, we expect them to be realizable in our framework. In addition, we intend to represent the framework with fitting representations of dependencies between monitors, such as covariances/correlations, conditional probabilities, odds ratios, and copula functions. These representations can be learned at design time from training data and simulations or possibly estimated at run time.

4 Summary and discussion

We have proposed a technique to dynamically estimate our confidence in the operation of an autonomous system by leveraging design-time safety guarantees. Our first step is to identify the assumptions underlying these guarantees and develop run-time monitors of probabilistic confidence in these assumptions. We then consider ways in which violations of these assumptions can lead to safety violations and use this relationship to compose assumption monitors into a system-wide assurance monitor.

We are currently in the process of implementing this vision in case studies of autonomous systems. These case studies will help us better understand the kinds of assumptions that can be effectively monitored and improve confidence composition techniques, as well as evaluate the utility of the proposed vision.

Open questions. There are several aspects of dynamic assurance that the described approach does not yet handle.

Confidence-based adaptation. For autonomous systems it is not enough to detect that the confidence in the mission progress is unacceptably low. Ideally, the system needs to react to the potential problem in a way that would restore the confidence or potentially abort the mission in order to save the vehicle from being lost. Several approaches loosely based on Simplex architecture [14] have been proposed [4]. The difference in our case is that reactions are triggered not by fault detection but by a drop in confidence. Such a reaction requires that we not only detect the problem, but also identify its causes. Here, this identification may be supported by the multi-level compositional structure of the assurance monitor. Extending our framework to support confidence-driven adaptation is another important research direction.

Predictive confidence. The meaning of confidence introduced above relies on the satisfaction of design-time assumptions up to the current moment. However, if the assumptions are confidently satisfied now, we should intuitively be confident in safety in the immediate future. So far we have not explored the relationship between the level of confidence and the time horizon for safety. It also will be an important avenue of our future work.

Making an explicit connection to a time horizon for safety may also allow us to reduce conservatism in our confidence estimate. As discussed in the first two sections, a violation of design-time assumption does not necessarily mean that a safety violation is imminent. Thus, the confidence in the satisfaction of the assumptions is expected to be lower than the confidence in system safety. This lower bound may turn out to be quite conservative and cause false alarms. One possible way to achieve a tighter connection between assumptions and safety confidence is to incorporate predictive monitors [1, 12] in our confidence calculation.

Confidence vs. robustness. We may be able to establish a connection between our notion of confidence and robustness of safety satisfaction. The idea is that, if the system operates near the boundary of the safety region, a small perturbation can make it unsafe. And since we cannot reliably observe small perturbations, our confidence cannot be too high in this situation. Therefore, a high confidence may probabilistically guarantee some minimal distance from the boundary. This is a promising yet challenging direction of research because formally, confidence and robustness are related through complex verification and monitoring models.

References

1. M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, Aug. 2014.
2. E. E. Alves, D. Bhatt, B. Hall, K. Driscoll, A. Murugesan, and J. Rushby. Considerations in assuring safety of increasingly autonomous systems. Technical Report NASA/CR–2018-220080, NASA, July 2018.
3. T. J. Carpenter, R. Ivanov, I. Lee, and J. Weimer. ModelGuard: Runtime Validation of Lipschitz-continuous Models. In *7th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'21)*, 2021. arXiv: 2104.15006.

4. A. Desai, S. Ghosh, S. A. Seshia, N. Shankar, and A. Tiwari. SOTER: A runtime assurance framework for programming safe robotics systems. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 138–150, 2019.
5. C. Ericson. *Hazard Analysis Techniques for System Safety*. Wiley, 2005.
6. N. B. Haupt and P. Liggesmeyer. A runtime safety monitoring approach for adaptable autonomous systems. In *Proceedings of SAFECOMP 2019: Computer Safety, Reliability, and Security*, volume 11699 of *LNCS*, pages 166–177, 2019.
7. R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178, 2019.
8. S. Jang, R. Ivanov, I. lee, and J. Weimer. Confidence Calibration with Bounded Error Using Transformations. In *In Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS'21)*, Feb. 2021.
9. P. Koopman and M. Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 9(1):90–96, 2017.
10. M. Machin, J. Guiochet, H. Waeselynck, J.-P. Blanquart, M. Roy, and L. Masson. SMOF - a safety monitoring framework for autonomous systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(5):702–715, May 2018.
11. D. J. Pumfrey. *The principled design of computer system safety analyses*. PhD thesis, University of York, 1999.
12. V. R. Royo, D. Fridovich-Keil, S. L. Herbert, and C. J. Tomlin. A classification-based approach for approximate reachability. In *International Conference on Robotics and Automation (ICRA)*, pages 7697–7704, 2019.
13. I. Ruchkin, O. Sokolsky, J. Weimer, T. Hedaoo, and I. Lee. Compositional probabilistic analysis of temporal properties over stochastic detectors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.
14. L. Sha. Using simplicity to control complexity. *IEEE Software*, 18(4):20–28, July/August 2001.
15. S. D. Stoller, E. Bartocci, J. Seyster, R. Grosu, K. Havelund, S. A. Smolka, and E. Zadok. Runtime Verification with State Estimation. In S. Khurshid and K. Sen, editors, *Runtime Verification*, Lecture Notes in Computer Science, pages 193–207. Springer Berlin Heidelberg, 2012.