# **Compositional Verification for Discovering Failures in Adaptive Flight Control Systems**

Sarah Thompson<sup>\*</sup> SGT, Inc., Moffett Field, CA, 94035

Misty D. Davies<sup>†</sup> and Karen Gundy-Burlet<sup>‡</sup> NASA Ames Research Center, Moffett Field, CA, 94035

Adaptive flight control systems hold tremendous promise for maintaining the safety of a damaged aircraft and its passengers. However, most currently proposed adaptive control methodologies rely on online learning neural networks (OLNNs), which necessarily have the property that the controller is changing during the flight. These changes tend to be highly nonlinear, and difficult or impossible to analyze using standard techniques. In this paper, we approach the problem with a variant of compositional verification. The overall system is broken into components. Undesirable behavior is fed backwards through the system. Components which can be solved using formal methods techniques explicitly for the ranges of safe and unsafe input bounds are treated as white box components. The remaining black box components are analyzed with heuristic techniques that try to predict a range of component inputs that may lead to unsafe behavior. The composition of these component inputs throughout the system leads to overall system test vectors that may elucidate the undesirable behavior.

## I. Introduction

A daptive flight control systems that utilize online learning neural networks (OLNNs) can theoretically allow an aircraft to maintain controllability after catastrophic failure. A prototype adaptive control system was successfully flown on the NASA F-15 ACTIVE aircraft using technology developed by the NASA Intelligent Flight Control (IFCS) project. However, the usefulness of these control systems are limited by their impermeability to standard validation and verification (V&V) techniques. In real systems, unmodeled dynamics are rife. Online learning neural networks will necessarily adapt based on these unmodeled dynamics, and it is difficult to bound the worst-case performance of these changing and highly-nonlinear systems. Even in the cases where the performance of the ideal control system can be bounded, usually by making assumptions which limit the ability of the control system to adapt or by bounding the dynamics of the aircraft, verifying the actual performance of the control system as implemented in code remains a rarely-tackled problem.

<sup>\*</sup> Staff Scientist, Intelligent Systems Division, Mail Stop 269-3.

<sup>&</sup>lt;sup>†</sup> Research Computer Engineer, Intelligent Systems Division, Mail Stop 269-3, AIAA Member.

<sup>&</sup>lt;sup>‡</sup> Research Scientist, Intelligent Systems Division, Mail Stop 269-3, AIAA Associate Fellow.

The Robust Software Engineering (RSE) group within the Intelligent Systems Division at NASA Ames Research Center has developed a suite of techniques that, when used in combination, may speed the discovery of adaptive control system failures as implemented in flight software<sup>1-5</sup>. Compositional verification allows the breakdown of the overall system into multiple components. Each component is tackled separately, and the behaviors of the components are composed to describe the behavior of the entire system. Each component can be analyzed using any of a plethora of formal methods techniques, including model-checking, abstract interpretation, and symbolic execution, with the overall goal of finding component inputs that would produce some undesirable output. This input is then used as the undesirable output for another component, until system level inputs are generated. These system-level inputs are used as a test vector to reproduce the final undesired behavior. The range of possible behaviors for some components in the overall system are not currently tractable to explicit techniques. For these components we use a combination of unsupervised<sup>6</sup> and supervised<sup>7,8</sup> machine learning techniques in a global sensitivity analysis<sup>9</sup> to model the behavioral structure of the component and to predict the component-level input test vectors. This method is heuristic and cannot guarantee that the component is safe; however, the method is likely to uncover unsafe behaviors. The compositional verification technique allows us to bound uncertainty for the overall system by determining which components need further verification. Furthermore, each test iteration improves our behavioral models of the black-box components and allows us to increase our certainty about their behavior.

## II. Methodology

As a prototype for this methodology we are using one of the implementations of the IFCS adaptive flight control system<sup>10-12</sup>. A high-level flow graph is shown in Figure 1. The output from the standard proportional-integral-derivative (PID) controller for the aircraft is sent to the OLNNs. The OLNNs compare the PID controller output with the output from the linearized plane reference model. The OLNNs attempt to drive the error to zero by augmenting the output from the PID controller before it is fed into the nonlinear dynamic inverse. The actuator model allows the control surfaces of the plane to be individually failed at any configuration.

The IFCS adaptive control system we are using was implemented as a Mathworks Simulink model. This kind of model-based design allows for ease of decomposing the overall system into modules, and then autocoding individually-modeled blocks into C/C++ code. Decomposing the system into modules is highly desirable from the point of view of automated verification – many techniques tend toward execution time that is exponential in the size of the code under test, so the benefits are frequently substantially better than linear. Secondly, the techniques that can be applied to a particular subsystem are dependent upon the code itself, with linear algorithms being far more amenable to analysis than their nonlinear counterparts. In order to maximize fidelity, analysis is conducted on the actual C/C++ flight code rather than on the Simulink model, with subsystems rendered in C++ source code separately by Real Time Workshop. The following approaches were used to analyze these subsystems:



**Figure 1. The IFCS adaptive control system.** This is the working version of the IFCS adaptive control system as used in this paper. The original version is modeled in Simulink. Our verification techniques are performed on the autocoded C/C++ from Matlab's Real Time Workshop.

*Partial evaluation/Symbolic Execution.* This approach allows certain functions (with bounded loops, limited use of pointers and strictly linear arithmetic operations) to be transformed into satisfiability modulo theorem (SMT) problems, which allow solvers such as Yices<sup>16</sup> to be used to generate test cases for arbitrarily chosen outputs.

Abstract Interpretation. This approach<sup>13,14</sup> allows some nonlinear functions to be safely approximated, making it possible to derive useful information about their behavior. Though this approach can generate false positives, when performed correctly it can never yield false negatives – consequentially, if code is shown to be correct by this method, this is a mathematically sound result.

*Explicit-state model checking*. Though model checking can be used to generate test cases, the search space for the flight control system is too large for this to be a useful approach. However, given bounded inputs from other techniques, model checking is an effective way to efficiently generate explicit counterexamples that make it very clear why a particular piece of code has failed. The MCP model checker<sup>2</sup> was used, and is capable of directly checking C and C++ code without prior translation or model extraction.

*Monte Carlo Filtering*. Monte Carlo Filtering is a type of global sensitivity analysis in which we choose the inputs and ranges most likely to lead to some output<sup>9</sup>. Most analyses of this type are computationally expensive, and tend to be limited to relatively small numbers of theoretically independent inputs, and also tend to assume that relationships between the inputs and outputs are smooth<sup>17,18</sup>. The types of problems being solved here involve failures—hence they can be non-smooth and of high dimensionality. To overcome the complications involved in finding the correlation coefficients for this sort of problem, we choose in practice to ignore the correlation

coefficients altogether and use machine learning techniques that sample the space and solve the original question directly.

### Acknowledgments

This research was conducted at NASA Ames Research Center. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government.

#### References

<sup>1</sup>Giannakopoulou, D., Kramer, J. and Cheung, S.C. "Analysing the Behaviour of Distributed Systems Using Tracta," *Journal of Automated Software Engineering, special issue on Automated Analysis of Software*, Vol. 6(1), Kluwer Academic Publishers, January 1999, pp. 7-35.

<sup>2</sup>Brat, G. and Thompson, S. "Verification of C++ Flight Software with the MCP Model Checker," *IEEE Aerospace Conference*, Big Sky, March 2008.

<sup>3</sup>Schumann, J., Gundy-Burlet, K., Pasareanu, C., Menzies, T., and Barrett, T. "Tool Support for Parametric Analysis of Large Software Systems", *Proceedings of Automated Software Engineering*, 23<sup>rd</sup> IEEE/ACM International Conference, 2008.

<sup>4</sup>Gundy-Burlet, K., Schumann, J., Barrett, T., and Menzies, T., "Parametric Analysis of a Hover Test Vehicle Using Advanced Test Generation and Data Analysis," *AIAA Aerospace*, 2009.

<sup>5</sup>Gundy-Burlet, K., Schumann, J., Barrett, T., and Menzies, T., "Parametric Analysis of ANTARES Re-entry Guidance Algorithms Using Advanced Test Generation and Data Analysis," 9<sup>th</sup> International Symposium on Artificial Intelligence, Robotics and Automation in Space, 2007.

<sup>6</sup>Fischer, B., and Schumann, J. "Autobayes: A System for Generating Data Analysis Programs From Statistical Models," *Journal of Functional Programming*, Vol. 13, 2003, pp. 483-508.

<sup>7</sup>Hu, Y., "Treatment Learning: Implementation and Application," Masters Thesis, Department of Electrical Engineering, University of British Columbia, 2003.

<sup>8</sup>Hu, Y., and Menzies, T. "Data Mining for Very Busy People," *IEEE Computer*, Vol. 36, No. 11, 2003, pp. 22-29.

<sup>9</sup>Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S., *Global Sensitivity Analysis: The Primer*, Wiley, Chichester, 2008, Chaps. 1, 5.

<sup>10</sup>Rysdyk, R. and Calise, A. "Fault-tolerant Flight Control via Adaptive Neural Network Augmentation," *AIAA American Institute of Aeronautics and Astronautics*, Vol. AIAA-98-4483, 1998, pp. 1722-1728.

<sup>11</sup>Calise, A. and Rysdyk, R. "Nonlinear Adaptive Flight Control Using Neural Networks," *IEEE Control Systems Magazine*, Vol. 21, No. 6, pp. 14-26.

<sup>12</sup>Schumann, J. and Gupta, P. "Monitoring the Performance of a Neuro-Adaptive Controller," *Proceedings of the 24<sup>th</sup> International Workshop on Bayesian Inference and Maximum Entropy Methods in Sciences and Engineering*, MAXENT 2004, 2004.

<sup>13</sup>Futamara, Y. "Partial evaluation of computation process—an approach to a compiler-compiler," *Systems, Computers Control*, Vol. 2, Issue 5, 1971, pp. 45-50.

<sup>14</sup>Jones, N., Gomard, C. and Sestoft, P. Partial evaluation and Automatic Program Generation. Prentice Hall, Englewood Cliffs, NJ, 1993.

<sup>15</sup>Cousot, P. and Cousot, R. "Abstract Intepretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints," *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1977, pp. 238-252.

<sup>16</sup>Dutertre, B. and de Moura, L. "The Yices SMT Solver," Tool paper found at URL: http://yices.csl.sri.com/tool-paper.pdf [cited 4 November 2009].

<sup>17</sup>Rose, K., Smith, E., Gardner, R., Brenkert, A. and Bartell, S. "Parameter Sensitivities, Monte Carlo Filtering, and Model Forecasting Under Uncertainty," *Journal of Forecasting*, Vol. 10, 1991: pp. 117-133.

<sup>18</sup>Oakley, J. and O'Hagan, A. "Probabilistic Sensitivity Analysis of Complex Models: A Bayesian Approach." *Journal of the Royal Statistical Society B*, Vol. 66, 2004, pp. 751-769.