Will This Be Formal?

Steven P. Miller*

Advanced Technology Center, Rockwell Collins,
400 Collins Rd NE, Cedar Rapids, Iowa 52498
spmiller@rockwellcollins.com

Abstract. While adding formal methods to traditional software development processes can provide very high levels of assurance and reduce costs by finding errors earlier in the development cycle, there are at least four criteria that should be considered before introducing formal methods into a project. This paper describes five successful examples of the use of formal methods in the development of high integrity systems and discusses how each project satisfied these criteria.

Keywords: Formal methods, model checking, theorem proving, avionics.

1 Introduction

Adding formal methods to traditional software development processes can provide very high levels of assurance and reduce costs by finding errors earlier in the development cycle. However, to be successful there are at least four criteria that should be considered before introducing formal methods into a project. This paper describes five successful examples of the use of formal methods in industry and discusses how each of the five projects satisfied these criteria.

2 Examples of the Successful Use of Formal Methods

This section describes five successful examples of the use of formal methods in the development of high integrity systems. In three projects model checking was used to verify the functional correctness of Simulink® models. In two projects theorem proving was used to verify security properties of microcode and source code.

2.1 FCS 5000 Flight Control System

One of the first applications of model checking at Rockwell Collins was to the mode logic of the FCS 5000 Flight Control System [1]. The FCS 5000 is a family of Flight Control Systems for use in business and regional jet aircraft. The mode logic determines

* This work was supported in part by the NASA Langley Research Center under contract NCC-01001 of the Aviation Safety Program (AvSP), the Air Force Research Lab under contract FA8650-05-C-3564 of the Certification Technologies for Advanced Flight Control Systems program (CerTA FCS) and the Air Force Research Lab under the EISTS program Delivery Order 4.

which lateral and vertical flight modes are armed and active at any time. While inherently complex, the mode logic consists almost entirely of Boolean and enumerated types and is written in Simulink. The mode logic analyzed consisted of five inter-related mode transition diagrams with a total of 36 modes, 172 events, and 488 transitions.

Desired properties of the mode logic were formally verified using the NuSMV model checker. To accomplish this, the Simulink models were automatically translated into NuSMV using a translation framework developed by Rockwell Collins and the University of Minnesota. This same translation framework also optimized the models for efficient analysis by the NuSMV BDD-based model checker.

Analysis of an early specification of the mode logic found 26 errors, seventeen of which were found by the model checker. Of these 17 errors, 13 were classified by the FCS 5000 engineers as being possible to miss by traditional verification techniques such as testing and inspections. One was classified as being unlikely to be found by traditional verification techniques.

2.2 ADGS-2100 Adaptive Display and Guidance System

One of the most complete examples of model checking at Rockwell Collins was the analysis of the Window Manager logic in the ADGS-2100 Adaptive Display and Guidance System [2]. The ADGS-2100 is a Rockwell Collins product that provides the display management software for next-generation commercial aircraft. The Window Manager (WM) is a component of the ADGS-2100 that ensures that data from different applications is routed to the correct display panel, even in the event of physical failure of one or more components.

Like the FCS 5000 mode logic, the WM is specified in Simulink and was verified by translating it into NuSMV and applying the NuSMV model checker. While the WM contains only Booleans and enumerated types, it is still quite complex. It is divided into five main components that contain a total of 16,117 primitive Simulink blocks that are grouped into 4,295 instances of Simulink subsystems. The reachable state space of the five components ranges from 9.8×10^9 to 1.5×10^{37} states.

Ultimately, 593 properties about the WM were developed and checked, and 98 errors were found and corrected in early versions of the model. As with the FCS 5000 mode logic, this verification was done early in the design process while the design was still changing. While the verification was initially performed by formal methods experts, by the end of the project, the WM developers themselves were doing virtually all the model checking.

2.3 Lockheed Martin Operational Flight Program

The Air Force Research Labs (AFRL) sponsored Rockwell Collins to apply model checking to the Operational Flight Program (OFP) of an Unmanned Aerial Vehicle developed by Lockheed Martin Aerospace as part of the CerTA FCS project [3]. The OFP is an adaptive flight control system that modifies its behavior in response to flight conditions. Phase I of the project concentrated on applying model checking to portions of the OFP, specifically the Redundancy Management (RM) logic, which were well suited to analysis with the NuSMV model checker. While relatively small (the RM logic consisted of three components containing a total of 169 primitive

Simulink blocks organized into 23 subsystems, with reachable state spaces ranging from 2.1×10^4 to 6.0×10^{13} states), they were replicated once for each of the ten control surfaces on the aircraft, making them a significant portion of the total OFP logic.

The verification of the RM logic took approximately 130 hours, with about half of that time spent preparing the models to be verified, correcting the errors found, and running the verification on the corrected models. A total of 62 properties were checked and 12 errors were found and corrected.

In Phase II of this project, the translator framework was extended so that an SMT-solver model checker could be used to verify portions of the numerically intensive inner loop control components in the OFP model. This phase is just being completed and will be reported on at a later date.

2.4 AAMP7G Intrinsic Partitioning

The AAMP7G is microprocessor developed by Rockwell Collins for use in its products. The AAMP7G provides high code density, low power consumption, long life cycle, and is screened for the full military temperature range. In addition, the AAMP7G includes a micro-coded separation kernel that provides MILS capability by ensuring the separation of data at different security classification levels.

To formally verify the AAMP7G intrinsic partitioning mechanism, it was first necessary to develop a formal description of what “data separation” means. This definition, now referred to as the GWV theorem, was specified as a formal property in the language of the ACL2 theorem prover [4]. To prove that the GWV theorem was satisfied by the AAMP7G, the microcode implementing the security kernel was modeled in ACL2 and the GWV theorem proven using the ACL2 theorem prover. To ensure that the ACL2 model of the microcode truly specified the behavior of the microcode on the AAMP7G, it was subjected to a painstaking code-to-spec review overseen by the National Security Agency (NSA) [5].

In May of 2005, the AAMP7G was certified as meeting the EAL-7 requirements of the Common Criteria as “… capable of simultaneously processing unclassified through Top Secret Codeword information”.

2.5 Greenhills Integrity-178B Real Time Operating System

The Greenhills Integrity-178B Real Time Operating System implements an ARINC-653 compliant APEX interface that has been certified to DO-178B Level A. It also includes a security kernel written in C that ensures the separation of data at different security classification levels.

To formally verify the Integrity-178B security kernel, the GWV specification of data separation developed for the AAMP7G was generalized to the GWVr2 theorem in order to describe the more dynamic scheduling managed by the OS [6]. As with the AAMP7G, the separation kernel was modeled in ACL2 language and the GWVr2 theorem was proven using the ACL2 theorem prover. To ensure that the ACL2 model of the C code truly specified the behavior of the Integrity-178B security kernel, it was also subjected to a painstaking code-to-spec review overseen by the NIAP/NSA.

Formal verification of the Integrity-178B security kernel satisfied the U.S. Government Protection Profile for Separation Kernels in Environments Requiring High

Robustness and the Common Criteria v2.3 EAL7 ADV requirements. Final certification of the Integrity-178B is now pending completion of NSA penetration testing.

3 Requirements for the Successful Use of Formal Methods

This section identifies four criteria for the successful use of formal verification on a problem and discusses how the examples described earlier satisfy these criteria.

3.1 Is the Problem Important?

While the cost of formal verification has been decreasing with the introduction of more powerful computers and analysis tools, it is still unusual for it to be accepted as an alternative to traditional verification techniques such as reviews and testing. To provide value, formal methods have to either satisfy a need for assurance greater than that provided by traditional means or have to reduce overall development costs by finding errors earlier in the life cycle. In either case, the problem being addressed should be inherently important.

This is the case for each of the examples cited earlier. The ADGS 2100 Window Manager is part of a DO-178B Level A system that provides critical functionality on Air Transport class aircraft. While the FCS 5000 Mode Logic is part of a DO-178B Level C system, errors in its implementation are highly visible to pilots of the aircraft, making the elimination of such errors very desirable. The Lockheed Martin Redundancy Management Logic implements important fault tolerance mechanisms essential for the correct operation of the UAV. Both the intrinsic partitioning mechanism of the AAMP7 and the Green Hills Integrity-178B Real-Time OS needed to provide separation of security domains and to satisfy the Common Criteria.

3.2 Are High Fidelity Models Available for Analysis?

Unlike testing, which verifies the actual implementation of a system, formal verification can only be applied to models of a system such as its design or code. While formal verification will typically find many errors that testing will miss, the availability of high fidelity models is critical for formal verification to be successful.

In each of the examples described earlier, high fidelity models were readily available or could be created at an acceptable cost. For the FCS 5000 Mode Logic, the ADGS 2100 Window Manager, and the Lockheed Martin OFP, unambiguous, machine-readable Simulink models had been created by the designers and used to generate source code. These models were automatically translated into high fidelity models for verification using the NuSMV and PROVER model-checkers.

In contrast, formal models of the AAMP7G microcode and the Greenhills Integrity-178B security kernel were created by hand and verified through painstaking code-to-spec reviews that were directly overseen by the NSA. While creation of these models was expensive, the cost was acceptable in order to achieve the extremely high levels of assurance required for certification.

3.3 Can the Properties of Interest be Stated Formally?

Even if a problem is inherently important, distilling the critical requirements into mathematical relationships that can be formally verified is not always possible. For the FCS 5000 Mode Logic, the ADGS 2100 Window Manager, and the Lockheed Martin OFP, the main challenge was identifying all the properties to be verified. Usually, this was done by formalizing the existing requirements and through discussions with the developers. Frequently, this process uncovered undocumented assumptions and missing requirements that had to be resolved through discussion.

In contrast, the challenge in formalizing the separation requirements for the AAMP7G intrinsic partitioning and the Greenhills Integrity-178B security kernel was developing a precise statement of the abstract concept of separation. While this was ultimately stated as a single ACL2 theorem (the GWV theorem for the AAMP7G and the GWVr2 theorem for the Integrity-178B security kernel), the process took several months of discussion and refinement.

3.4 Are the Right Analysis Tools Available?

The final consideration is whether the right analysis tools are available for the problem. To be effective, the formal verification tools must be able to verify the properties of interest, produce results sufficiently quickly, produce results at acceptable cost, and only require expertise that their users can be expected to know or acquire. Typically, the capability of the analysis tools will play as large a role in selecting which problems will be formally verified as will the inherent need for high assurance.

For the FCS 5000 Mode Logic, the ADGS 2100 Window Manager, and the first phase of the Lockheed Martin OFP, the models all had state spaces of less than 10^{50} reachable states, making them well suited for verification with BDD-based model checkers. Even so, the success of all these projects depended on the ability to automatically generate high fidelity models that were optimized for analysis. This was especially true for the ADG-2100 Window Manager where the design models were being revised daily. If the analysis had taken more than a few hours, the project would not have been a success. In the second phase of the Lockheed Martin OFP analysis, the numerically intensive nature of the problem required the use of the Prover model checker. While successful, the greater expertise required to use an SMT-solver instead of a BDD-based model checker poses real challenges to the transfer of this technology to production developments.

Verification of the security separation provided by the AAMP7G and the Greenhills Integrity-178B security kernel was performed using the ACL2 theorem prover. These efforts took several months and significant expertise to complete. Even so, the project was highly successful due to the inherent importance of the problem, the stability of the AAMP7G microcode and Integrity-178B source code, and the availability of experts to formulate the formal properties and complete the proofs.

4 Future Directions

This paper has briefly described five industrial applications of formal methods and identified some of the main reasons those projects were successful. While the availability of the right tools played a key role in each example, there are still many directions for research that could make formal methods even more useful. An obvious need is to extend the domain of models for which model checking is feasible to include numerically intensive models with transcendental functions. While SMT-solvers hold great promise, there may be difficulty in getting practicing engineers to use them on a routine basis. Industrial users could also use help in determining when they have an optimal set of properties. Also valuable would be a sound basis for determining what testing can be replaced by analysis. Finding ways to compose the verification of subsystems to verify entire systems will be essential to cope with the increasing size of digital systems. Techniques for modeling and verifying asynchronous systems using message passing is another area of need.

References

1. Miller, S., Anderson, E., Wagner, L., Whalen, M., Heimdahl, M.: Formal Verification of Flight Critical Software. In: AIAA Guidance, Navigation and Control Conference and Exhibit, AIAA-2005-6431, American Institute of Aeronautics and Astronautics (2005)
2. Whalen, M., Innis, J., Miller, S., Wagner, L.: ADGS-2100 Adaptive Display & Guidance System Window Manager Analysis, CR-2006-213952, NASA (2006)
3. Whalen, M., Cofer, D., Miller, S., Krogh, B., Storm, W.: Integration of Formal Analysis into a Model-Based Software Development Process. In: 12th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2007), Berlin, Germany (2007)
4. Greve, D., Wilding, M., Vanfleet, W.M.: A Separation Kernel Formal Security Policy. In: Fourth International Workshop on the ACL2 Prover and Its Applications (ACL2-2003) (2003)
5. Greve, D., Richards, R., Wilding, M.: A Summary of Intrinsic Partitioning Verification. In: Fifth International Workshop on the ACL2 Prover and Its Applications (ACL2-2004) (2004)
6. Greve, D., Wilding, M., Richards, R., Vanfleet, W.M.: Formalizing Security Policies for Dynamic and Distributed Systems. In: Systems and Software Technology Conference (SSTC 2005), Utah State University (2005)