Static Analysis of Autocoded Software for Aerospace Systems

Eric Feron
School of Aerospace Engineering
Georgia Institute of Technology, Atlanta, GA 30332
feron@gatech.edu
Phone: (404) 894-3062

Arnaud Venet
Kestrel Technology LLC
4984 El Camino Real #230, Los Altos, CA 94022
arnaud@kestreltechnology.com
Phone: (650) 967-4408

I. Overview of our Approach

Embedded software-based control systems are commonly constructed using model-based design environments such as MATLAB/Simulink™ from MathWorks. These environments allow the system designer to establish critical properties ensuring the reliability of the system (stability, disturbance rejection, etc.) directly at the model level, using a rich mathematical toolset. However, the software implementation substantially transforms the mathematical model by introducing numerous programming artifacts (aggregate data structures, pointers) and altering the numerical representation (platform-dependent floating/fixed-point arithmetic, and, in the most extreme cases, conversion from continuous-time dynamics to discrete-time dynamics). Verifying that the reliability properties of the system are preserved by the implementation is extremely challenging, yet in many cases critically important. Model-based design environments usually come with an *autocoder* i.e., a code generation tool that automatically synthesizes an implementation of the embedded controller from the specification of its model. Autocoders are getting increasingly used in practical applications for they greatly simplify the implementation process. In aerospace industry however, autocoding is essentially precluded because its properties are considered to be not adequately trustworthy.

Static program analysis tools have recently proven successful in tackling the certification of embedded software-based control systems. ASTREE [1], developed by P. Cousot's team in France, can automatically verify the consistency of floating-point arithmetic in the electric-command control system of the A380, Airbus' super-jumbo carrier. C Global Surveyor [2], developed by Kestrel Technology LLC, can verify the absence of pointer manipulation errors in the mission-control software of NASA's Mars Exploration Rovers. However, the scope of static analysis has been essentially limited to robustness properties, i.e., ensuring the absence of runtime errors during the execution of the program. Verifying functional properties by static analysis for a system to be used in the field requires (1) translating a reliability property of the model into the implementation setting, using the appropriate data structures and numerical libraries, and (2) tracking the evolution of this property over all execution paths using abstract interpretation techniques. This process requires a tight coupling between the model description and its implementation. While such tight coupling is rarely achieved in practice, it can exist at least when the implementation is automatically generated from the model. Autocoders, like Real-Time Studio™ for Simulink™, are increasingly being used in industry for the development of embedded control software. This means that static analysis techniques specialized for codes automatically generated from high-level models can be developed to meet market needs.

The approach we are investigating consists of translating the formal proof of reliability properties of an embedded logic into a dedicated static analyzer that automatically carries out the corresponding proof on the code generated from the model. Whereas today's commercial (and even known academic) static analyzers for embedded mission- and safety-critical software are handcrafted, we propose to use our prior research to construct the dedicated static analyzer automatically. Ultimately, the system designer would be provided with a fully automated engine that performs verification of the generated code without requiring any additional information other than the high-level model specification.

II. Challenges

There are two major challenges in developing a tool for the automated verification of reliability properties of embedded control software automatically generated from a high-level model:

1) How do we translate a property of the high-level model into a property of the code generated from that model? What are the features of the autocoder we must know to effectively build this translator?
2) How do we specify the basic components of the static analyzer required to verify the desired property? How do we specialize the analyzer to the code generated by a given autocoder?

We discuss these challenges in the following subsections.

A. Property Translator

The autocoder of the model-based design environment generates code from the model in a predictable way. Therefore, we expect to be able to map a property of the model's variables into a property of the data structures used in the model's implementation. Commercial model-based design environments offer rich libraries of basic components for building systems and the properties of interest may greatly vary depending on the nature of the system designed. The family of systems we have been investigating is that of closed-loop dynamical systems, represented on the one hand by a family of differential equations that capture the system's physics, and on the other hand the closed-loop control algorithm and its associated code. The functional properties we are interested in include closed-loop systems stability, closed-loop system performance (eg tracking performance), and reachability analyses. We are currently interested in describing how to map that property to the implementation by conducting an extensive study of the code generated from a benchmark of systems in that family.

B. Static Analysis Specification Framework

Checking the transposed property of the model on the code may require a specific analysis algorithm that takes into account the underlying computational model (floating/fixed-point) and the nature of the reliability property (linear or ellipsoidal invariants). Moreover, the code generated by the model-based design environment may use programming language constructs (like pointers or union types in C) that pose a difficulty for the static analyzer. These constructs may require dedicated analysis algorithms (pointer analysis, type analysis) specially tailored for the particular structure of code produced by the autocoder. These static analyzers are strongly specialized toward the family of models and properties considered. This does not require rewriting the analyses from scratch for each of these configurations. We need a library of baseline static analysis algorithms (pointer analysis, floating-point analysis, type analysis, etc.) and a framework for combining them in a way to address each configuration's unique requirements. We are in the process of establishing a taxonomy of static analysis algorithms and describing a specification framework for expressing arbitrary combinations of these algorithms.

REFERENCES

[1] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE Analyser. In *Proceedings of the European Symposium on Programming (ESOP'05)*, volume 3444 of *Lecture Notes in Computer Science*, pages 21–30, 2005.

[2] A. Venet and G. Brat. Precise and efficient static array bound checking for large embedded C programs. In *Proceedings of the International Conference on Programming Language Design and Implementation*, pages 231–242, 2004.

Dr. Feron is the Dutton/Ducoffe Professor of Aerospace Software Systems at Georgia Institute of Technology since 2005. Prior to that, he was a tenured associate Professor of Aeronautics and Astronautics at MIT from 1993 to 2005. He holds degrees from Ecole Polytechnique, Ecole Normale Superieure and Stanford University in Applied Mathematics, Computer Science and Aeronautics and Astronautics. His research focuses on the applications of control system theory and optimization to the design and analysis of real-time systems ranging from Air Traffic Control to autonomous helicopter flight to control system software analysis. His research on autonomous aerobatic helicopter flight is featured in MIT's list of Research Firsts as the lone entry for the year 2002 and has received world-wide news coverage. Eric Feron's research in static software analysis aims at applying advanced optimization methods to automatically search for invariants in time-critical control software. Eric Feron has authored or co-authored two books and over 100 research publications. He has three pending patents. He has been the associate editor for several academic journals, most lately the International Journal of Field Robotics. Eric Feron is an Advisor for the French Academy of Technologies. He has supervised the research of 10 PhD students and more than 40 MS students.

Dr. Venet is a specialist in the application of Abstract Interpretation to static analysis. Since his thesis work, he has held R&D positions at PolySpace and Trusted Logic in France, and Kestrel Technology (KT) in the United States. Dr. Venet was one of the early contributors at PolySpace, helping to create the first commercial static analyzer for industrial-size applications. There he led the industrialization of PolySpace Verifier for C and designed a version of PolySpace Verifier for JavaCard. At KT, his initial assignments were in support of NASA's high assurance goals. There he designed and developed C Global Surveyor (CGS). CGS is a large-scale static array-bound checker that currently holds the distinction for the largest code set analyzed with 100% verification assurance for array bound compliance. CGS was successfully applied to the mission control software of Mars Exploration Rovers and science components of the ISS. Currently he is developing a platform for the design and deployment of customizable, industrial-scale static analyzers for robust large scale integrated systems, malware detection, and other applications. Dr. Venet holds patents on the application of static analysis for JavaCard bytecode compression and software watermarking. Dr. Venet received a Ph.D. in Computer Science from Ecole Polytechnique (France) and an M.Sc. in Computer Science from Ecole Normale Superieure (France).