AN INVESTIGATION INTO THE COMPATABILITY OF ADA* AND FORMAL VERIFICATION TECHNOLOGY

David Preston
IIT Research Institute
4550 Forbes Blvd., Suite 300
Lanham, MD 20706

Karl Nyberg
Grebyn Corporation
P. O. Box 1144
Vienna, VA 22180

Robert Mathis
9712 Ceralene Dr.
Fairfax, VA 22032

August, 1987

To be submitted to The 6th National Conference on Ada Technology

Much of the research on this project was conducted under contract to the Rome Air Development Center (RADC), contract number F30602-86-C0111.

*Ada is a registered trademark of the U. S. Government (Ada Joint Program Office).

ABSTRACT

Formal code verification is the primary focus of this paper; however, other approaches to raising the confidence in software that may be applicable to the security community are investigated.

Of primary interest in this study are two basic questions:

Is the construction of an automated Ada verification environment within the grasp of today's technology?

If yes, what resources would be required to construct the environment?

Initially, several components of the research were identified. One area was to review each Ada construct as defined by the Ada Language Reference Manual (LRM). This review is centered on the impact of each construct on formal verification. The perspective in this component is code verification for Ada using traditional axiomatic verification. The assessment is based on the feasibility to develop a verification axiom, or proof rule, for each construct in isolation. A detailed listing of the constructs and the effect of each on verification was generated. Since Ada was not developed to be a verifiable language, there are some constructs that will defy formal verification; these challenges do not seem to be overwhelming and could presumably be controlled by restrictions to the use of the language. The Ada implementations of tasking and exception handling are the two greatest challenges that the language constructs provide for verification. Of these, tasking is the far greater challenge.

In addition to the axioms, this study investigated the state of the art of other support technologies required for code verification. These include a formal definition and a specification language. Specification languages were viewed both from the perspective of their necessity for formal code verification and from the perspective of their applicability to other techniques, both formal and less formal, for increasing the understanding of software.

The study addresses the issue of resources required to construct an Ada verification environment by providing a brief review of related efforts that are ongoing and by projecting, primarily by use of analogy, anticipated resources for construction of an Ada verification environment.

The study went beyond traditional axiomatic code verification by investigating alternative methods to increase the understanding of what Ada software will, and will not, do. The two alternatives studied are the verification techniques utilized

in the certification of software safety, notably the work of Nancy Leveson, and the human verification of the IBM "cleanroom" approach developed by Harlan Mills.

Also, research in application of formal methods to Ada outside of the code verification domain was investigated. Two specific areas are the application of formal methods to specification analysis and to runtime assertions.

There are a few general conclusions that have been developed during the course of this study:

Reasons not to use Ada at the A1 certification level or below are more culturally based than technically based.

Formality in software development should not be all or nothing.

Analysis of constructs that challenge verification can be a basis for developing coding restrictions on software that does not need to be verified.

There are very few people who adequately understand the application of formal methods.