Analytic Verification at NASA
A View from the Trenches

Willem C. Visser

July 31, 1999

Autonomous software is considered a major enabling technology for NASA in order to achieve the goal of their slogan: "better, cheaper, faster". As software systems become more independent from human intervention, the consequences of errors in the interactions between different systems increases. For example, in a recent experiment (called RAX[1]) where autonomous software was used during an actual flight (as part of the DEEP SPACE-1 mission) a deadlock occurred which meant valuable hours were lost and the system had to be reset from earth. The software used in this case was rigorously tested, but the deadlock never occurred during testing. Current testing techniques cannot reliably find errors that are due to subtle interactions between concurrent components, since these interactions are often time dependent.

The *Automated Software Engineering*[2] (ASE) group at NASA Ames Research Center is developing analytic verification techniques to augment traditional testing to ensure the reliability of the software being developed at NASA. Specifically we are using a technique called *model checking* which allows one to check whether a behavioral property is satisfied by a system. Model checking has had much success in the area of design verification, specifically hardware design verification. However, very few people have tried *model checking of software systems*. Model checking software systems has been one of the main focus areas of the ASE group for the past two years.

In this presentation the lessons we have learned from applying model checking to software systems will be discussed as well as how we have refined our goals accordingly. A major component of the presentation will highlight the success stories we have had so far, e.g. finding 5 errors in sections of the RAX software before flight and then also finding the actual deadlock (that occurred in a part that we had not model checked before!) in a 2 day experiment after the flight. The *JAVA PathFinder*, a model checker that can check properties of JAVA programs directly on the source code of the program, will be discussed and a demo shown (time permitting). Since software often have (potentially) infinite state spaces we use abstraction techniques to reduce the size of the systems to make it amenable to model checking. Currently we are working on ways to automatically abstract a JAVA program when given as input a function mapping concrete data domains to abstract domains, for example mapping integers to an abstract *signs* domain where there are only three values *positive*, *negative* and *zero*. Some of our most recent results in this area will be presented.

The talk will be concluded by showing the group's future research directions.

[1]This software won the "Software of the Year" award at NASA in 1999. url: rax.arc.nasa.gov
[2]url: ase.arc.nasa.gov