Verification support environment for Real-Time Java verification

SOGA Tetsunori(510057)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 8, 2007

Keyword: Real-Time Java, Verification Support Environment, Parametric Model Checking, Program Structure.

In this reseach, we present a verification support environment for real-time Java, which we consider as a promising platform for embedded software development.

As far as the reliability is concern, embedded software, while running under restricted resources, needs the guarantee to meet real-time property. The problems about software reliability arise in most embedded software projects, because the software scale has become larger and more complex, however, the development time frame is still the same. This results in high development cost and put more burden on the developers.

In order to solve this problem, formal methods, especially model checking, have recently get a lot of focus. Model checking requires less technical knowledge, and the application of model checking in the software industry is considered more practical compare with other techniques. Additionally, writing programs that deal with real-time property under restricted resource needs a lot of consideration. This craftman task demands highly skilled programmers and inherently causes the problems on development cost and software quality. Employing the real-time Java development platform is a solution to help reducing programmer's tasks.

The goal of this research is to provide a supporting tool for software verification to be used in embedded software development. The verification support environment presented in this research is intended for the embedded software developed by real-time Java. An experiment on a part of the tool and the evaluation are presented.

In the process of model checking real-time Java program, the first step is to construct the corresponding program structure from the program which will be used to create a state transition model. The program structure we mention here is a graph resulted from the analysis of real-time Java program's class file. The graph is constructed by analyzing the bytecode instructions in class file, taking the branching, method invocation, and return

instructions as nodes; other instructions that get executed sequentially are taken as edges. In other words, the program structure divides the real-time Java program into segments where each segment represents a straight execution path of the program. This allows us to estimate the execution cost of each segment by measuring the execution time of each bytecode instruction on the real-time Java execution environment.

The overview of real-time Java program verification using the proposed verification support environment is explaned as follow. The real-time Java program class file is taken as the input, and the the corresponding program structure is constructed by analysing the program's bytecode instructions. We use the bytecode engineering library (BCEL) for dealing with the bytecode analysis. The execution cost table is created by measuring the execution time of each bytecode on the target executing environment. Then, a parametric time structure is created from the program structure with execution cost information. The parametric time structure is a state transition model used in this research for parametric verification. Parametric verification is a verification technique with abstraction by representing undecided values, such as execution time, in variables. Then, we write the property to be verified and apply a parametric verification tool on the parametric model. The result obtained from parametric verification is inequalities. We can use the inequalities to decide optimum values of parameters and fix the program accordingly.

We implement a tool for experimenting on the program structure construction from a Java class file. This experiment is a success of a part in the first step of the verification support environment discussed so far. The problems we have found in this research is that there is a part in the Java program that contains invocations of native code which could not be handled by BCEL. The invocations of native code is simply ignored. The other tools may be necessary for dealing the native code. Finally, the program structure created by the tool is not complete for constructing the parametric time structure. More sophisticated program structure is required to construct the parametric model for verification.