CS4215 Programming Language Implementation

Challenge B: Byte-code Verification for sVML

1 Setting

Early in the project (by Monday, 12/3), the interested student indicates his/her intention to work on the project by email to the lecturer, with a rough outline of the project schedule. Discussions with tutors and lecturer are done on a per-need basis. A mid-way check will be around 20/3, with an informal meeting (tutor and/or lecturer). The project completion date is on Friday, 30/3.

2 Goal

The security of the Java Virtual Machine relies not on the language Java, but on the security of the runtime system to execute Java Virtual Machine code. Indeed, JVM code does not have to be compiled using Java at all.
The Java Runtime System verifies every `.class` file before it executes it. The goal of this challenge is to design and implement a byte code verifier for sVML that implements one aspect of bytecode verification, namely protection from operand stack overflow.
Every function in our low-level machine (see Lab Task 7) indicates the maximal size to which its operand stack may grow. So the runtime system can limit the data structure for the operand stack to that size. However, when the function uses a larger size than declared, the machine would crash. So the constant `MAXSTACKSIZE` in `LDF` is a *contract* between code and runtime system.
Your task is to write a program that verifies that any execution of the body of a method will stay within the declared maximum operand stack size, regardless of the path that the execution will take.
Hint: To complete the task, you will have to follow every possible path through each function body, starting from the first instruction of the function, and check the size of the operand stack at each reachable instruction.

3 Requirements

- Download `http://www.comp.nus.edu.sg/~cs4215/labtasks/challenge_verification.zip`

- Look at `simPLvmSecure`. The runtime system in `simPLvmSecure.simpl` calls a function `verify` in `VerifyMachineCode`. Implement this function to achieve the goal stated above.
- Submit the resulting file `VerifyMachineCode.java`, and a brief text document that describes the approach you have taken.

4 Submission and Assessment

After project completion, the student sends all software and other documents in a zip file to the lecturer via email. Please include instructions how to install and run the application. The submission will be assessed by the tutors and lecturer. If the project goals are achieved, the student will be asked to present the solution in person to the lecturer and tutors. Sufficient achievement leads to issuing of an Assignment Voucher, which the student can use at the end of the semester to replace any module assignment score by full score.