Bachelor thesis: Guided Navigation in Symbolic Execution Trees

TECHNISCHE UNIVERSITÄT DARMSTADT

Software Engineering Group — Martin Hentschel — hentschel@cs.tu-darmstadt.de

Context

Debugging is a central and unavoidable task in software development which requires a considerable amount of the overall project effort. However, debugging tools evolved slowly in the last decades providing mainly the standard functionality for step wise execution, inspection of the current program state and suspension of the execution before a marked statement is executed.

A difficult task for a user of classical interactive debuggers is to setup the initial program state which leads to an execution of the system exhibiting the bug. This requires often a number of non-trivial steps to construct the required complex data structures. Another reason why debugging is hard is that even if the buggy statement is found, the cause of failure during program execution must be determined. But intermediate states are no longer available and the user has to repeat debugging sessions until the cause of failure is understood.

A way out of this dilemma is *symbolic execution*. It allows to execute directly any method or even a single statement using symbolic in lieu of concrete values. Thus execution cannot follow a single path and has to split whenever multiple paths are possible resulting in a *symbolic execution tree* which captures the entire program behavior up to a certain point.

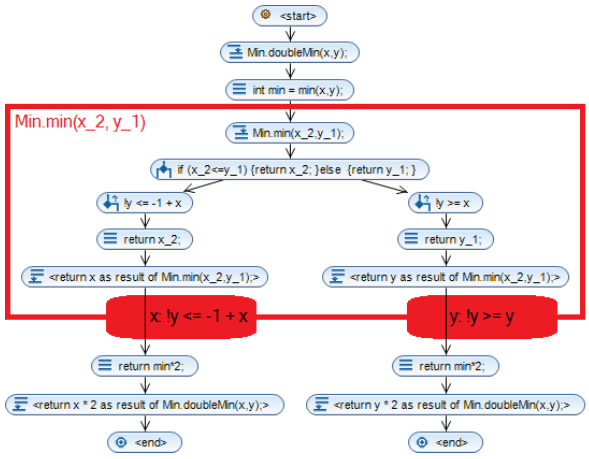
The Symbolic Execution Debugger (SED), available at www.key-project.org, extends the Eclipse debug platform by interactive symbolic execution. It defines first how symbolic execution trees are represented and provides second the functionality to visualize them. A concrete implementation which works for nearly full sequential JAVA based on verification tool KeY is also provided.

Thesis

Symbolic execution trees can be very large and thus it is hard for the user to concentrate on paths of interest. More precise, a tree will be wide in case of many possible execution paths and each path can be deep in case of many performed steps.

The aim of this thesis is to develop concepts to guide the user during symbolic execution focussing of the parts of interest and to improve visualization of symbolic execution trees. Results have to be realized as part of the Symbolic Execution Debugger.

Several ideas should be developed and explored. For instance, related symbolic execution tree nodes (e.g. the execution of a method body) can be summarized. Such a summary can be collapsed to hide contained nodes, but it has still to show possible paths together with the conditions when they are taken. During execution, the summary can be automatically collapsed when it is completed. The red box in the following mockup shows how it could look like. Nodes within the red frame won't be shown if it is collapsed and its size will be reduced to a minimum.



Contact

Martin Hentschel — Software Engineering Group — hentschel@cs.tu-darmstadt.de — S2 02 | A223