Abstract

Nowadays, static code analysis is widely used. It is a kind of automatic verification of computer programs, which does not require the programs' execution. Static code analysis is based on the theory that is using different logical methods. One of such methods is abduction.

Abduction is a form of logical inference. Its goal is to find the minimum additional to the given premise hypothesis, which is necessary for deriving the given conclusion. Considering abduction in the context of separation logic (the expansion of Hoare logic that is used for describing programs specifications) makes it possible to apply the abduction for deriving pre- and post-conditions for the different parts of the code. Besides, there exists the dual to abduction problem. It is called frame inference. The combination of the abduction with the frame inference gives the more common concept, which is called bi-abduction. It is the base for bi-abduction rule. There also exists the algorithm that is using this rule for the memory leaks detection in the programs. This algorithm works by applying the bi-abduction for some certain function in the program and computing its footprint (it is the memory part that is using by the function). Footprints of all the functions are computing independently from each other. This makes possible to use this algorithm for the memory leaks detection in the incomplete programs. Also, this algorithm is scalable, which means that it continues to work correctly even on the very big programs, because, initially, it executes on the small parts of the program and, then, composes the found specifications for the memory leaks detection in the whole program.

The project describes this algorithm, implements it in the software and demonstrates its work on the memory leaks detection in Java programs.