

# JPF-BDD

## Abstract

jpf-bdd is a Java PathFinder extension that provides Binary Decision Diagram-based execution for Java programs. It "replaces" the handling of a user-defined set of boolean variables. These variables are not handled by jpf-core anymore. Instead, jpf-bdd manages the values of the variables using Binary Decision Diagrams. On various programs we improve both the runtime and the memory consumption of the verification (compared to verification with jpf-core).

## Contact

student: Alexander von Rehin

mentor: Franco Raimondi

## Repository

<https://bitbucket.org/rhein/jpf-bdd>

## Description

This JPF extension provides BDD-based execution for Java programs. It changes the execution instructions for some byte-code statements to store the current program state in BDDs. Only byte-code operations that affect a user-defined set of Boolean variables are changed. When these are executed, BDDs are used to manage the value of the variables.

More information on the project is available in our wiki: <https://bitbucket.org/rhein/jpf-bdd/wiki/Home>

This is a short summary:

The Boolean variables that should be handled with BDDs have to be marked with the Annotation `@TrackWithBDD` (notice that jpf-bdd is currently incompatible with any other plugin that needs a special `InstructionFactory`).

To run the extension, simply include jpf-bdd as any other jpf project. This is described in the configuration documentation:

<http://babelfish.arc.nasa.gov/trac/jpf/wiki/user/config>

When the verification is started jpf-bdd, will manage the marked variables with BDDs. jpf-bdd modifies the verification in two ways:

1. It uses Breadth First Search.
2. It changes the generated reachability tree, so some statements might be executed less often than in a standard jpf-core verification. This might have influence on the generated output (e.g. if these statements write on the console).
3. We do not assign a value to the variable during instantiation. This behavior is different than the standard Java semantics which define the default Boolean value to be "false". If your program depends on this behavior, please assign an initial value to the variable ("boolean x = false;").

With the exception of the behavioral changes discussed in bullet 3 we do not change the execution semantics.