

## Symbolic PathFinder

Symbolic PathFinder (SPF) combines symbolic execution with model checking and constraint solving for test case generation. In this tool, programs are executed on symbolic inputs representing multiple concrete inputs. Values of variables are represented as numeric constraints, generated from analysis of the code structure. These constraints are then solved to generate test inputs guaranteed to reach that part of code. Essentially SPF performs symbolic execution for Java programs at the bytecode level. Symbolic PathFinder uses the analysis engine of the Ames JPF model checking tool (i.e. jpf-core).

## Features

Symbolic PathFinder

- Performs symbolic execution of Java bytecodes
- Handles complex math constraints, data structures and arrays, multi-threading, pre-conditions, strings (on-going work)
- Applies to (executable) models and code
- Generates test vectors and test sequences that are guaranteed to achieve user-specified coverage (e.g. path, statement, branch, MC/DC coverage)
- Measures coverage.
- Generates JUnit tests, Antares simulation scripts, etc. (output can be easily customizable)
- During test generation process, checks for errors
- Is flexible, as it allows for easy encoding of different coverage criteria
- Is integrated with simulation environment (on-going work)

## Applications

Test input generation for Java container classes, NASA guidance navigation and control (GNC) software; script generation for testing execution engines. Symbolic PathFinder has been used at Fujitsu Labs for testing Web applications - see [Fujitsu press announcement](#).

## Other info

- [Tool Documentation](#) -- documentation for Symbolic PathFinder
- [Combining Unit-level Symbolic Execution and System-level Concrete Execution for Testing NASA Software](#) (paper published in ISSTA 2008 proceedings) -- describes Symbolic PathFinder
- [Generalized Symbolic Execution for Model Checking and Testing](#), (paper published in TACAS 2003 proceedings) -- describes handling of input data structures using "lazy initialization"
- [Symbolic String Execution](#) (Master Thesis) -- describes String analysis
- [Symbolic PathFinder](#) (presentation given at JPF workshop, MSR, ISSTA 2008)
- [Lecture Notes from Marktoberdorf Summerschool 2012](#)
- [Old tool documentation](#)
- [Fujitsu press announcement](#) released about using and extending Symbolic PathFinder ([projects/jpf-symbc](#)) for comprehensive testing of Java web applications

## Repository

The Mercurial repository is on <http://babelfish.arc.nasa.gov/hg/jpf/jpf-symbc>