

# Online Test Generation with PathCrawler. Tool demo.\*

Nikolai Kosmatov

Bernard Botella

Muriel Roger

Nicky Williams

CEA, LIST, Software Safety Laboratory, PC 94  
91191 Gif-sur-Yvette France  
Email: [firstname.lastname@cea.fr](mailto:firstname.lastname@cea.fr)

## Extended Abstract

**Introduction.** Structural testing is widely used in industrial verification processes of critical software. Automation of test case generation brings obvious benefits. In critical systems processes where structural testing is required by the development norm, manually creating tests from the specification fails to achieve complete satisfaction of the coverage criterion. In this case, automatic methods help to reach the objectives which are not covered and provide corresponding path conditions that may be used to refine the specification if needed. They may also determine whether the objectives which are not yet covered are really infeasible. When the development process does not impose any structural testing activity, the use of a structural test generation tool is a way to increase the quality of the software with a very low cost overhead.

PathCrawler [1, 3] is a structural test generation tool developed at CEA LIST that may be used to automate structural testing. PathCrawler generates tests for C functions respecting the all-paths criterion, or the  $k$ -path criterion (for a given  $k \geq 0$ ), which restricts the generation to the paths with at most  $k$  consecutive iterations of each loop. The user provides the compilable ANSI C source files containing the function under test, which we denote by  $f$ , and all other functions called in it. Test generation with PathCrawler contains two major phases.

In the first phase, PathCrawler extracts the inputs of  $f$  and instruments the source code in order to create a test driver. This phase uses the Frama-C tool<sup>1</sup>, developed at CEA LIST. The extracted inputs include the formal parameters of  $f$  and the non constant global variables used by  $f$ . A test case will provide a value for each input of  $f$ . The user may remove some variables from the inputs, define the domains of the inputs, a test context and an oracle.

The second phase generates test cases for  $f$  with the se-

lected criterion. Implemented in Eclipse constraint logic programming system<sup>2</sup>, the generator combines symbolic execution in constraints and concrete execution. The paths of  $f$  are explored in a depth-first search. The test-input generation method implemented in the PathCrawler tool has proved its effectiveness in the successful generation of test cases covering different paths in numerous examples of C code. Fig. 1 shows an example of test generation results.

**The Demo.** This demonstration presents a new version of PathCrawler developed in an entirely novel form: that of a test-case server which is freely accessible online at [2]. The user uploads the C source code to be tested and the server displays the test-cases generated by PathCrawler and a detailed justification of the coverage. The user can define the test context and browse the results using specialised interfaces in the form of web-pages. The server allows many test-case generation sessions to be run in parallel in a completely robust and secure way.

The advantage of making the tool available in this form is that it does not have to be downloaded and installed. Instead, it can be immediately run either on the programs which are provided, or on the user's own code. The user can easily try out different test contexts, so as to appreciate their significance, and can also upload an oracle and see the verdict of each test. We have used pathcrawler-online as a teaching aid for university students and as a way for our industrial partners to evaluate the tool.

The pathcrawler-online test server is an early form of cloud computing. Test-case generation is well-suited to deployment "in the cloud". Indeed, it needs powerful servers and specific tools which are expensive to install and maintain and are used only during the phase of testing. Sharing a testing service "in the cloud" has obvious benefits. Our future work includes extending pathcrawler-online to explore the parallelisation and distribution of different components of the test-case generation process.

\*This work has been partially funded by the ANR-PREDIT MAS-COTTE, ITEA SPICES, ITEA TWINS and ANR CAVERN projects.

<sup>1</sup><http://www.frama-c.com/>

<sup>2</sup><http://www.eclipseclp.org>

## Menu

Summary

Test-cases

Paths  
explored

Trace

## Test-cases generated

## General test session information

Function under test: Bsearch

Coverage criterion:  all feasible paths

## Test-cases generated





Test case ID	Verdict 	Time, sec. 	Prefix ID 	Path 
TC_1	failure	0	P_1	bsearch.c : +18;+21;+23;+18;+21;+23;+18;+21;+23;-18;-30;
TC_2	failure	0	P_4	bsearch.c : +18;+21;+23;+18;+21;+23;+18;+21;-23;-18;-30;
TC_3	success	0	P_7	bsearch.c : +18;+21;+23;+18;+21;+23;+18;-21;-23;-18;-30;
TC_4	failure	0	P_12	bsearch.c : +18;+21;+23;+18;+21;-23;+18;+21;+23;-18;-30;
TC_5	failure	0	P_15	bsearch.c : +18;+21;+23;+18;+21;-23;+18;+21;-23;-18;-30;
TC_6	success	0	P_18	bsearch.c : +18;+21;+23;+18;+21;-23;+18;-21;-23;-18;-30;
TC_7	success	0	P_23	bsearch.c : +18;+21;+23;+18;-21;-23;+18;+21;+23;-18;-30;
TC_8	success	0	P_27	bsearch.c : +18;+21;+23;+18;-21;-23;+18;-21;-23;-18;-30;
TC_9	success	0	P_34	bsearch.c : +18;+21;-23;+18;-21;-23;-18;-30;
TC_10	failure	0	P_38	bsearch.c : +18;+21;-23;+18;+21;+23;+18;+21;+23;-18;-30;
TC_11	failure	0	P_41	bsearch.c : +18;+21;-23;+18;+21;+23;+18;+21;-23;-18;-30;
TC_12	success	0	P_44	bsearch.c : +18;+21;-23;+18;+21;+23;+18;-21;-23;-18;-30;
TC_13	success	0	P_49	bsearch.c : +18;+21;-23;+18;+21;-23;-18;-30;
TC_14	success	0	P_53	bsearch.c : +18;-21;-23;+18;-21;-23;-18;+30;+30b;
TC_15	failure	0	P_54	bsearch.c : +18;-21;-23;+18;-21;-23;-18;+30;-30b;
TC_16	success	0	P_58	bsearch.c : +18;-21;-23;+18;+21;+23;+18;+21;+23;-18;-30;
TC_17	success	0	P_62	bsearch.c : +18;-21;-23;+18;+21;+23;+18;-21;-23;-18;-30;

Figure 1. Example of test generation results.

## References

- [1] B. Botella, M. Delahaye, S. Hong-Tuan-Ha, N. Kosmatov, P. Mouy, M. Roger, and N. Williams. Automating structural testing of C programs: Experience with PathCrawler. In *AST'09*, Vancouver, Canada, May 2009.
- [2] PathCrawler. Online version of the PathCrawler test generation tool, 2010–2011. <http://pathcrawler-online.com/>.
- [3] N. Williams, B. Marre, P. Mouy, and M. Roger. PathCrawler: automatic generation of path tests by combining static and dynamic analysis. In *EDCC'05*, pages 281–292, Budapest, Hungary, April 2005.