BTestBox
-
User Manual

January 30, 2017

1 Revisions

Version	Date	Comment
0.1	23/01/2017	Initial Version

Contents

2 Introduction

This document is the User manual for the tool BTestBox. It describes how the tool must be used to test the translation and the coverage of B0 implementations.

Main principles of the testing will be explained, and different ways to launch the tool will be described.

2.1 Dependencies

BTestBox relies on ProB capability of evaluating a predicate and the target translator proficiency of translating the B0 code. Also, it uses some Atelier-B functions while generating the predicate. BTestBox uses the BXML tree file created when Atelier-B makes the PO.

2.2 Limitations

BTestBox inherits the limitations from ProB and the target translator. Furthermore, in actual state BTestBox don't handle arrays very well. CONSTRAINTS clause still not well defined and only working with natural constraint type. Right now, BTestBox only works on Windows, but as future work will also be executable on Linux or Mac OS.

3 Testing an Implementation

3.1 Input B0 Language

The input B0 language that is supported by the tool and the produced result depend on the target translator used during the test, and each of them has their supported subset of the B0 language. Those are described in their own documents [1] and [2], respectively C4B and B2LLVM. HIA and TRATADA don't have proper documentation, but the document [3] provided by ClearSy might give some help.

3.2 Coverage Criteria

BTestBox allows the user to choose between several coverages. Each coverage has their different meaning and result.

This section presents the different profiles.

3.2.1 Provided Coverages

The different provided coverages are Code Coverage, Branch Coverage, Path Coverage and Clause Coverage. Each of them makes BTestBox act differently to achieve the coverage.

Code Coverage: To achieve this coverage, each command on the code has to be executed at least once;

Branch Coverage: Given a graph of the code, each branch of this graph shall be visited at least once;

Path Coverage: Given a graph of the code, each path of this graph shall be executed at least once;

Clause Coverage: Given all the clauses of a predicate, each clause has to be evaluated to false and to true.

3.3 Operation To Test

For now, this option is unable to be changed, and BTestBox will always try to cover every operation in the implementation. But in the future, the user will choose which of the operations he want to be tested.

3.4 Test Project Name

BTestBox needs to add some functions to the user implementations and machines, due to this a name of a folder is asked to settle down the tests and copied files. BTestBox overwrites the previous files with the same name than the created.

BTestBox copies the machines and implementations necessary for the test, and create the files:

- TestSet_**COVERAGE_MACHINE_NAME**.mch;
- TestSet_**COVERAGE_IMPLEMENTATION_NAME**.imp;
- runTest_**COVERAGE_MACHINE_NAME**.mch;

- runTest_**COVERAGE_IMPLEMENTATION_NAME**.imp.

And also the translation of the files are generated in the folder lang\in the proper folder of the translation. Additionally, a main file for running the tests is created.

3.5 Compiler

To run the translated files is necessary the user give the compiler that he wants to use.

3.6 Optional Arguments

Right now, the user has no reason to use this area, but in the future, this will handle strategies to approach the tests.

4 Launching BTestBox in AterlierB

4.1 Configuration resources

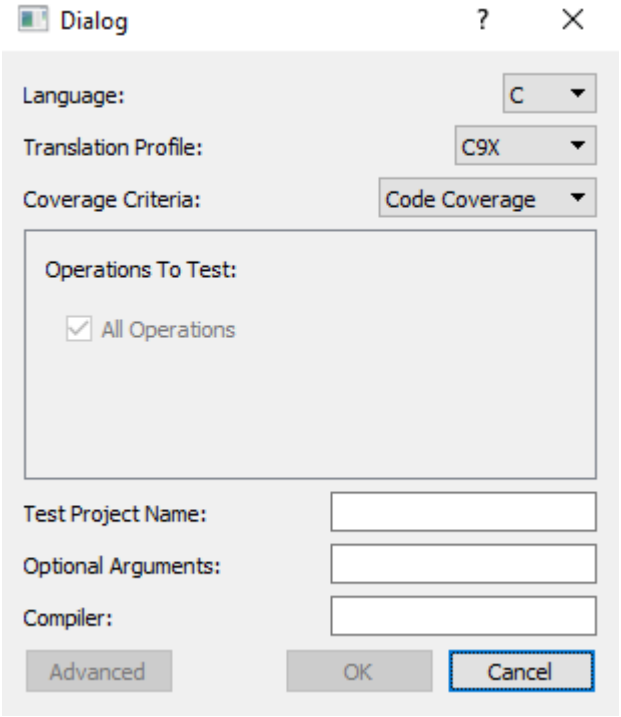
The resource that is used to launch BTestBox from AtelierB tools is btest-box.etool file in the extensions folder. It must contain the path to the BTestBox interface executable.

The resource is automatically positioned in AtelierB interface if the etool extension file is in the extensions folder of AtelierB.

4.2 Testing a target translator and the implementation coverage

To test a component in AtelierB, hit "Test Code Translation" entry in the Component menu. A window will pop up with the options to a test [4.2].

Figure 1: BTestBox Inteface



The user shall select the language that he wants the code to be translated. When choosing a language that has a profile, the profile selector will be visible.

The box for coverage criteria has all the coverage supported by BTestBox. It is not possible to click the "OK" button until a name is given for the test project and the compiler.

Once clicked "OK" BTestBox searches in AtelierB resource code for the Probcli path, if there is none, a new window [4.2] pop up to the user set it.

Setting the Probcli path and hitting "OK" again will make BTestBox start the tests. When it finish, including error or success, a report window is visible.

Figure 2: BTestBox demanding probcli path

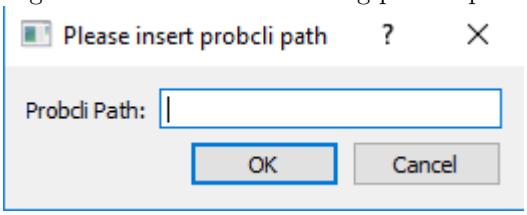
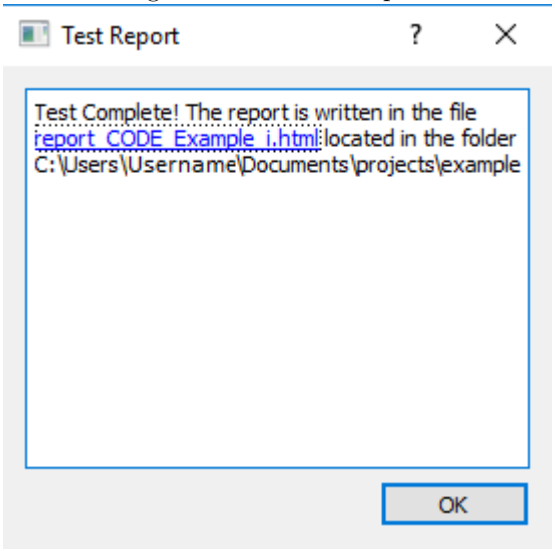


Figure 3: BTestBox report



4.2.1 Output Directory

Produced files are written in the folder with the given name for the test project name area in the interface. On the other hand, this folder is found in the same folder than the project.

5 Launching BTestBox in command line

5.1 Sintax

BTestBox has a easy syntax with the command parameters found in the following table [1]. For calling BTestBox from the command line, use the btestbox file.

Table 1: BTestBox arguments

Parameter	Description
target_language	The language to translate.
translator_profile	The translator profile.
coverage	The coverage to test.
atelier	Atelier-B directory.
project_directory	Directory of the component.
copy_directory	Target directory name to do the tests.
b_module	Name of the B module (implementation)
compiler	The compiler to compile the translated code.
probcliPath	Probcli path
-v –verbose	Outputs some information while running.

Right now, BTestBox has no interesting optional argument to the user. So is very simple to call it from the command line. The order described in the table is the order of the commands.

6 Report

The report is summarized in an HTML file located in the test project directory. Each report create 3 HTML files:

1. **COVERAGE_IMPLEMENTATION_NAME**_FilesLib.html
2. reportText_**COVERAGE_IMPLEMENTATION_NAME**.html
3. report_**COVERAGE_IMPLEMENTATION_NAME**.html

The main file of the report is the third archive of the enumeration, and it leads to the other two. It summarizes the whole test in an easy interface, showing the tested operation and the achieved percentage.

When finishing a successful test using the atelier-B interface, the report window pop up with a hyperlink that opens the report HTML.

7 Known bugs and errors

BTestBox shall work fine for simple components, but when the implementation starts to be complex, generating a big predicate, or one that demands a big set expansion by ProB, evaluating the predicate start to be time-consuming.

BTestBox only works with a limited case of arrays, and it is not advisable to use it with implementations that have arrays.

When calling BTestBox from the interface, it will only return if the program failed or not.

Following, a list of known errors:

- The component is a machine and there is no implementation.
- Any machine on the clauses IMPORTS, SEES, EXTENDS, INCLUDES are not implemented. Not having an implementation is not mandatory of making BTestBox fail, but it will always search for the implementation first and only will use the machine if not find the former. Since a machine may be non-deterministic, it is not prudent to use, and a warning will be printed on the command line.
- One or more components don't have their PO generated and consequently the BXML file.
- BTestBox found an unknown tag in the BXML tree and don't know how to generate the predicate for that or generate amiss.
- The path to Probcli is not provided or it is wrong.
- BTestBox files are not in the correct folder of Atelier-B.
- The compiler is invalid.
- The files created by BTestBox are not compatible with the translator B0 subset.

Following, a list of known bugs:

- Arrays with more than more than two dimension.
- Sequential arrays (arrays with one dimension).
- Constraints that are not natural type.

References

[1] Clearsy. *C4B - User Manual*. 2016.

[2] David Deharbe and Valerio Medeiros Jr. *Translation of B Implementations to the LLVM: Specification*. 1 edition.

[3] ClearSy, 4 edition, 2017.