CIT-BACH 1.1 User's Guide

Tatsuhiro Tsuchiya

March 24, 2016

1 Overview

CIT-BACH (pronounced as "sit back" or "sea eye tea back") is a test case generation tool for combinatorial interaction testing. CIT-BACH is written completely in Java and requires JRE 1.6.0 or higher to run.

CIT-BACH is free software under the zlib license. It uses JDD for BDD manipulation. JDD is also free software under the zlib license. The source code is available at the tool's web page.

2 Running CIT-BACH

To run the tool, run the command

```
java -jar cit-bach.jar [options]
```

For example, suppose `model.txt` is the following text file.

```
# parameters and values
Display (16MC 8MC BW)
Viewer (Graphical Text None)
Camera (2MP 1MP None)
VideoCamera (Yes No)
VideoRingtone  (Yes No)

# group: ensures all value combinations are covered
{Display Viewer Camera}

# constraints
#1
(if (== [Viewer] Graphical) (or (== [Display] 16MC) (== [Display] 8MC)))

#2
(if (== [Camera] 2MP) (or (== [Display] 16MC) (== [Display] 8MC)))

#3
(if (== [Viewer] Graphical) (not (== [Camera] 2MP)))

#4
(if (== [Display] 8MC) (<> [Camera] 2MP))
```

```
#5
(if (== [VideoCamera] Yes)
   (and (or (== [Camera] 2MP) (== [Camera] 1MP))
        (or (== [Display] 16MC) (== [Display] 8MC))
   )
)

#6
(if (== [VideoRingtone] Yes) (== [VideoCamera] Yes))

#7
(not (and (== [Display] 16MC) (== [Viewer] Text) (== [Camera] 2MP)))
```

Then the command

```
java -jar cit-bach.jar -i model.txt
```

produces the following output.

```
#SUCCESS,4358,i,Model1.txt,s,,o,,c,2,random,4358,repeat,1
Display,Viewer,Camera,VideoCamera,VideoRingtone
16MC,None,2MP,Yes,Yes
16MC,Graphical,1MP,No,No
8MC,Graphical,1MP,Yes,Yes
16MC,Text,1MP,Yes,No
8MC,Text,1MP,No,No
BW,Text,1MP,No,No
16MC,None,1MP,No,No
8MC,None,1MP,No,No
BW,None,1MP,No,No
16MC,Graphical,None,No,No
8MC,Graphical,None,No,No
16MC,Text,None,No,No
8MC,Text,None,No,No
BW,Text,None,No,No
16MC,None,None,No,No
8MC,None,None,No,No
BW,None,None,No,No
16MC,None,2MP,No,No
16MC,Text,1MP,Yes,Yes
```

3 Options

The options are as follows:

- -c [2 |3 |4 |5 |all] : Strength (default: 2)
- -i file: Model file
- -o file: Output file

- -s file: Seeding file
- -random N: Random seed
- -repeat N: N times repetition
- -lang [JP |EN] : Language
- -policy: License notice

3.1 -c: Strength

This option specifies the strength of tuples that should be covered. For example, if `-c 2` is used, then the resulting test suite will satisfy pair-wise coverage. If `all` is specified, then an exhaustive test suite will be generated.

3.2 -i: Model file

This specifies the input file that describes the model of the *System Under Test (SUT)*. A model file consists of at most three parts. Lines starting with # are comments.

The first part represents the parameters and the possible values of these parameters. Each line is used to specify each parameter and its values. For example,

```
Display (16MC 8MC BW)
```

states that parameter Display takes either one of the three values: 6MC, 8MC, and BW.

The second part is used to specify groups of parameters. This part is optional. Groups can be used to specify parameters whose interactions should be all covered. That is, for any of these groups, all value combinations among the group's parameters will be completely covered in the resulting test suite. For the above example, the three parameters, Display, Viewer, and Camera, form a group and thus, all testable combinations for these parameters occur in the above test suite.

The third part specifies constraints. This part is also optional. Constraints induce the condition that every test case must satisfy. Constraints are written in a sort of parenthesized Polish prefix notation. The operators supported by CIT-BACH are as follows:

- Logic operators: not, ==, <>, if, ite, or, and
- String comparators: ==, <>
- Arithmetic comparators: ===, !==, <, >, <=, >=

Basically all raw values are treated as strings. An exceptional case is when an arithmetic comparator is used. In that case the operands will be cast to the Java double type.

The value of a `paramter` is specified as `[paramter]`.

3.3 -o: Output file

This can be used to specify the output file.

3.4 -s: Seeding file

A seeding file allows one to construct a test suite by extending an incomplete test suite. Below is an example of a seed file.

```
# parameters
Display, Viewer, Camera, VideoCamera, VideoRingtone

# seeding rows
16MC   , None  , 1MP   , Yes         , Yes
       ,       , 1MP   ,             , No
16MC   ,       ,       , No          ,
```

A seeding file must start with a row that lists parameter names. After that, one non-blank row represents a complete or incomplete test case. The tool constructs a test suite by filling in unspecified values and adding new test cases to the seeding test cases.

3.5 -random: Random seed

CIT-BACH uses random numbers in generating test cases and hence the resulting test suites vary for different runs. This option can be used to reproduce the same result.

The first row of the output contains the random seed that was used. For the above example of the output, the random seed is 4358.

3.6 -repeat: N times repetition

This option enforces the tool to internally generate N test suites with different random seeds and then output the smallest one.