User Guide for ACTS

ACTS is a test generation tool for constructing *t*-way combinatorial test sets. Currently, it supports t-way test set generation with t ranging from 1 to 6. Combinatorial testing has been shown very effective in detecting faults that are caused by unexpected interactions between different contributing factors. The tool provides both command line and GUI interfaces.

This document is organized as follows. Section 1 provides an overview of the core features of ACTS. Section 2 provides information about the command line interface. Section 3 provides information about the GUI interface.

1 Core Features

1.1 T-Way Test Set Generation

This is the core feature of ACTS. A system is specified by a set of parameters and their values. A test set is a *t*-way test set if it satisfies the following property: Given *any t* parameters (out of all the parameters) of a system, every combination of values of these *t* parameters is covered in at least one test in the test set.

Currently, ACTS supports *t*-way test set generation for $1 \leq t \leq 6$. Empirical studies show that t being up to 6 is sufficient for most practical applications. A special form of 1-way testing, called base-choice testing, is implemented in ACTS. Base-choice testing requires that every parameter value be covered at least once and in a test in which all the other values are base choices. Each parameter has one or more values designated as base choices. Informally, base choices are "more important" values, e.g., default values, or values that are used most often in operation.

Several test generation algorithms are implemented in ACTS. These algorithms include IPOG, IPOG-D, IPOG-F, IPOG-F2, PaintBall. In general, IPOG, IPOG-F, and IPOG-F2 work best for systems of moderate size (less than 20 parameters and 10 values per parameter on average), while IPOG-D and PaintBall are preferred for larger systems.

ACTS supports two test generation modes, namely, *scratch* and *extend*. The former allows a test set to be built from scratch, whereas the latter allows a test set to be built by extending an existing test set. In the *extend* mode, an existing test set can be a test set that is generated by ACTS, but is incomplete because of some newly added parameters and values, or a test set that is supplied by the user and imported into ACTS. Extending an existing test set can save earlier effort that may have already been spent in the testing process.

1.2 Mixed Strength

This feature allows different parameter groups to be created and covered with different strengths. For example, consider a system consisting of 10 parameters, P1, P2, …, and P10. A default relation can be created that consists of all the parameters with strength 2. Then, additional relations can be created if some parameters are believed to have a higher degree of interaction, based on the user's domain knowledge. For instance, a relation could be created that consists of P2, P4, P5, P7 with strength 4 if the four parameters are closely related to each other, and their 4-way interactions could trigger certain software faults. ACTS allows arbitrary parameter relations to be created, where different relations may overlap or subsume each other. In the latter case, relations that are subsumed by other relations will be ignored by the test generation engine.

1.3 Constraint Support

Some combinations are not valid from the domain semantics, and must be excluded from the resulting test set. For example, when we want to make sure a web application can run in different Internet browsers and on different operating systems, the configuration of IE on Mac OS is not a valid combination. A test that contains an invalid combination will be rejected by the system (if adequate input validation is performed) or may cause the system to fail. In either case, the test will not be executed properly, which may compromise test coverage, if some (valid) combinations are only covered by this test.

ACTS allows the user to specify constraints that combinations must satisfy to be valid. The specified constraints will be taken into account during test generation so that the resulting test set will cover, and only cover, combinations that satisfy those constraints. Currently, constraint support is only available for the IPOG algorithm. Constraint support for other algorithms will be added in a future release.

1.4 Coverage Verification

This feature is used to verify whether a test set satisfies t-way coverage, i.e. whether it covers all the t-way combinations. A test set to be verified can be a test set generated by ACTS or a test set supplied by the user (and then imported into ACTS).

1.5 Expected Outcome Specification

This feature allows the user to specify expected outcome, in terms of a number of output parameters and their values, for each test case. Output parameters are specified in the same way as input parameters. However, their values are specified manually by the user. This is in contrast with input parameters whose values are generated by the tool. This feature is designed to facilitate test automation.

2. Command Line Interface

There is a separate jar file for the command line version and for the GUI version. In this section, we assume that the jar file for the command line version is named acts_cmd.jar, and the jar file for the GUI version is named acts_gui.jar. (The actual jar files names in the release package are actually longer, and contain the version numbers.)

The command line version can be executed using the following command:

java <options> –jar acts_cmd.jar ActsConsoleManager <input_filename> <output_filename>

The various options are:

-Dmode=scratch|extend
    scratch - generate tests from scratch (default)
    extend - extend from an existing test set
-Dalgo=ipog|ipog_d|bush|rec|paintball|ipof|ipof2|basechoice
    ipog - use algorithm IPO (default)
    ipog_d - use algorithm IPO + Binary Construction (for large systems)
    bush - use Bush's method
    paintball - use the Paintball method
    ipof - use the IPOF method
    ipof2 - use the IPOF method
    basechoice - use Base Choice method
-DfastMode=on|off
    on - enable fast mode
    off - disable fast mode
-Ddoi=<int>:
    specify the degree of interactions to be covered
-Doutput=numeric|nist|csv|excel
    numeric - output test set in numeric format
    nist - output test set in NIST format (default)
    csv - output test set in CSV format
    excel - output test set in EXCEL format -
Dcheck=on|off:
    on - verify coverage after test generation
    off - do not verify coverage (default)
-Dprogress=on|off:
    on - display progress information (default)
    off - do not display progress information
-Dhunit=<int>:
    the number of tests extended during horizontal extension per progress unit -
Dvunit=<int>:
    the number of pairs covered during vertical growth per progress unit
-Ddebug=on|off:
    on - display debug info
    off - do not display debug info (default)
-Drandstar=on|off:
    on – randomize don’t care values
    off – do not randomize don’t care values
-Dcombine=<all>:
    all - every possible combination of parameters

The above usage information can be displayed using the following command:

java -jar acts_cmd.jar

In the command line, <input_file> contains the configuration information of the system to be tested. The format of a configuration file is illustrated using the following example:

```
[System]
-- specify system name
Name: Test Configuration from Rick

[Parameter]
-- general syntax is parameter_name : value1, value2, ....
-- only compare with MINSEP and MAXALTDIFF general

Cur_Vertical_Sep (int) : 299, 300, 601

High_Confidence (boolean) : TRUE, FALSE
Two_of_Three_Reports_Valid (boolean) : TRUE, FALSE

-- Low and High, only compare with Other_Tracked_Alt
Own_Tracked_Alt (int) : 1, 2
Other_Tracked_Alt (int) : 1, 2

 -- only compare with OLEV
 Own_Tracked_Alt_Rate (enum) : 600, 601
Alt_Layer_Value (int) : 0, 1, 2, 3

-- compare with each other (also see NOZCROSS) and with ALIM
Up_Separation (int) : 0, 399, 400, 499, 500, 639, 640, 739, 740, 840
Down_Separation (int) : 0, 399, 400, 499, 500, 639, 640, 739, 740, 840
Other_RAC (enum) : NO_INTENT, DO_NOT_CLIMB, DO_NOT_DESCEND
Other_Capability (enum) : TCAS_TA, OTHER
Climb_Inhibit (boolean): TRUE, FALSE
```

Note that -- represents comments that exist only to improve the readability of the configuration file.

Also note that the default heap size for the Java Virtual Machine may not be adequate for large configurations. The user is recommended to change the default heap size, if necessary, using the following command:

java -Xms <initial heap size> -Xms<max heap size> <options> -jar acts_cmd.jar ActsConsoleManager <input_file> <output_file>

3 GUI Interface

There are two ways to launch the GUI front end. One way is to double-click the jar file for the GUI version, which is an executable jar file. The other way is to execute the jar file for the GUI version on the command prompt as follows:

java -jar acts_gui.jar FireEyeMainWin

The following command can be used to change the default heap size for java virtual machine, if necessary:

java -Xms <initial heap size> -Xms<max heap size> <options> -jar acts_gui.jar FireEyeMainWin

Figs. 1 and 2 show the general layout of the ACTS GUI. The System View component is a tree structure that shows the configurations of the systems that are currently open in the GUI. In the tree structure, each system is shown as a three-level hierarchy. That is, each system (top level) consists of a set of parameters (second level), each of which has a set of values (leaf level). If a system has relations and constraints, they will be shown in the same level as the parameters.

Right to the System View is a tabbed pane consisting of two tabs, namely, Test Result, which is shown in Fig. 1, and Statistics, which is shown in Fig. 2. The Test Result shows a test set of the currently selected system, where each row represents a test, and each column represents a parameter. Output parameters are also displayed as columns. The Statistics tab displays some statistical information about the test set. In particular, it includes a graph that plots the growth rate of the test coverage with respect to the tests in the test set displayed in the Test Result tab. Drawing the graph may involve expensive computations, and thus the graph is shown only on demand, i.e. when the Graph button is clicked.

3.1 Create New System

To create a new system, select menu *System -> New*, or the first icon in the toolbar, to open the *New System* window. The *New System* window contains a tabbed pane of three tabs, namely, *Parameters*, *Relations*, and *Constraints*. The three tabs are shown in Figs. 3, 4, and 5, respectively.

The *Parameters* tab (Fig. 3) allows the user to specify the parameters, as well as the values of those parameters, in the new system. Currently, four parameter types are supported, *Boolean*, *Enum*, *Number*, and *Range*. *Range* is a convenience feature that allows multiple, consecutive integers to be input quickly. Note that parameter names cannot contain spaces. (The characters that can be contained in a parameter name are the same as those in a variable name in Java programs.)

There is a checkbox next to each parameter value in the *New System* window. If the checkbox next to a parameter value is checked, this value is designated as a base choice value. A parameter may have multiple base choice values, meaning that multiple checkboxes can be checked. Base choice values are only used by the base-choice testing algorithm. The base choice values will be highlighted after they are added to the system. As mentioned earlier, base-choice testing is a special form of 1-way testing.

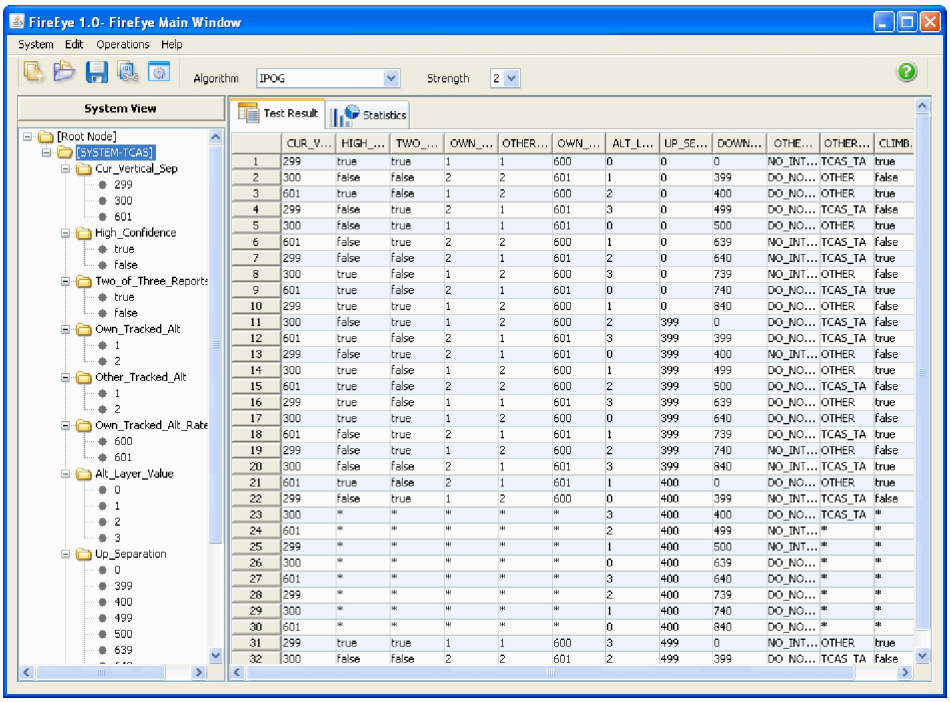

Figure 1 The Main Window – Test Result Tab

A default relation is automatically created that consists of all the parameters that have been specified in the Parameters tab with the default strength (which is specified in the Options window, see Section 1.8). This default relation is provided as a convenience feature (so that the user does not need to do anything in this tab if the user does not want to specify any relation), and can be removed like other user-defined relations.

The *Constraints* tab (Fig. 5) allows the user to specify constraints so that invalid combinations can be excluded from the resulting test set. Generally speaking, a constraint is specified using a restricted form of first-order logical formulas. In the following, we give a formal syntax of the expressions that can be used to specify a constraint:

<Constraint>::=<Simple_Constraint>|<Constraint><Boolean_Op><Constraint>
<Simple_Constraint> ::= <Term><Relational_Op><Term>
<Term>:=<Parameter>|<Parameter><Arithmetic_Op><Parameter>
| <Parameter><Arithmetic_Op> <Value>
<Boolean_Op>:="&&" |"||" |"=>"

<Relational_Op> := “=” | “!=” | “>” | “<” | “>=” | “<=”
<Arithmetic_Op> := “+” | “-” | “” | “/” | “%”*

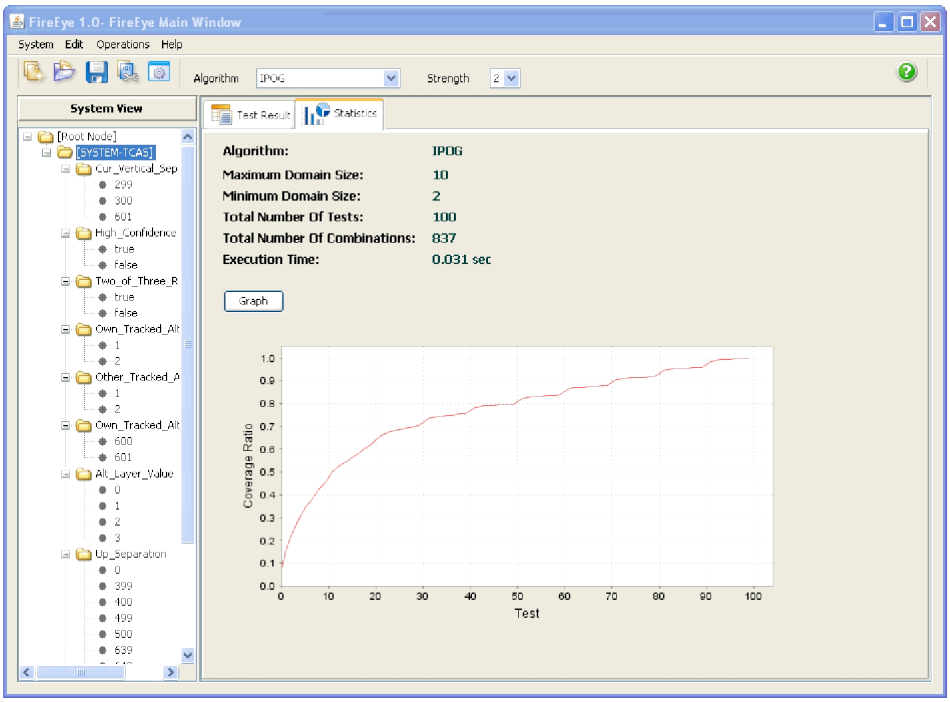

Figure 2. The Main Window - Statistics Tab

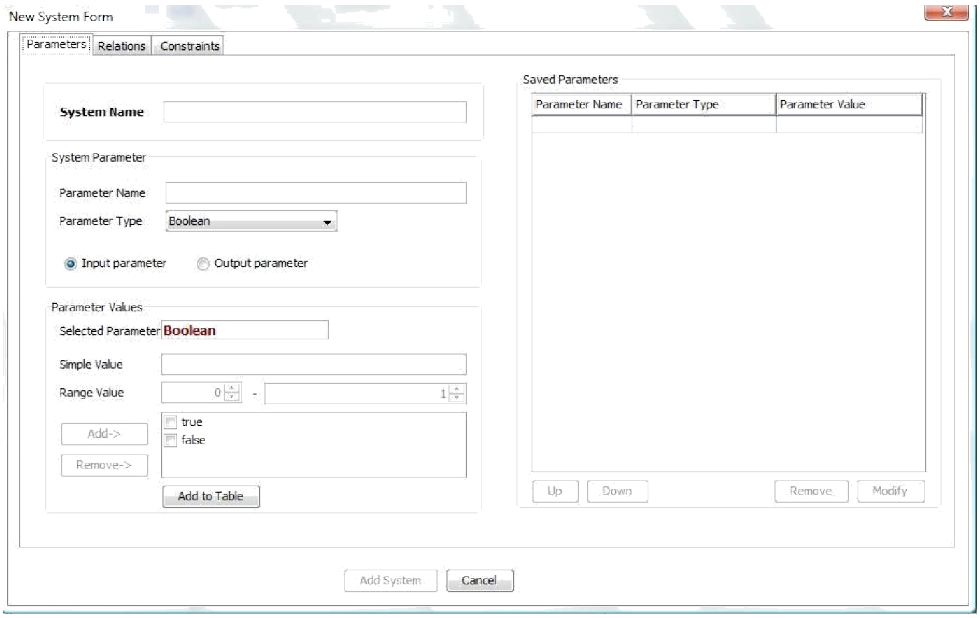

Figure 3. New System Window – Parameters

There are three types of operators: (1) *Boolean* operators (*Boolean_Op*), including &&, ||, =>; (2) *Relational* operators (*Relational_Op*), including =, !=, >, <, >=, <=; and (3)

Arithmetic operators (*Arithmetic_OP),* including +, -, *, /, %. Note that arithmetic operators can appear in a term expression (<*Term*>) only if the parameters involved in the term expression are of type Number or Range. Also, four of the relational operators, namely, >, <, >=, <=, can appear in a simple constraint expression (*Simple_Constraint*) only if both of the terms involved in the simple constraint are evaluated to a parameter value of type Number or Range.

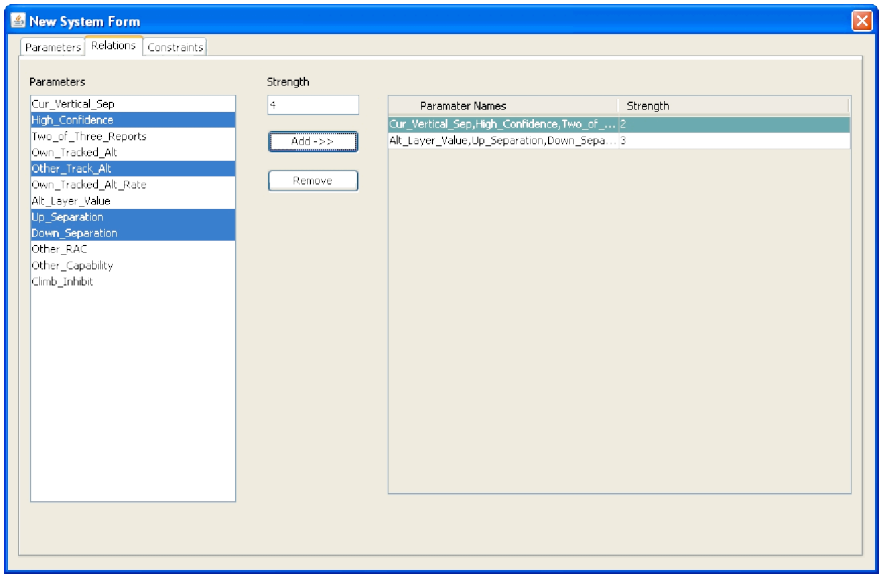

Figure 4. New System Window – Relations

The following are examples of various constraints that can be specified:

Constraint 1: (OS = “Windows”) => (Browser = “IE” || Browser = “FireFox” || Browser = “Netscape”), where OS and Browser are two parameters of type Enum. This constraint specifies that if OS is Windows, then Browser has to be IE, FireFox, or Netscape.

Constraint 2: (P1 > 100) || (P2 > 100), where P1 and P2 are two parameters of type Number or Range. This constraint specifies that P1 or P2 must be greater than 100.

Constraint 3: (P1 > P2) => (P3 > P4), where P1, P2, P3, and P4 are parameters of type Number or Range. This constraint specifies that if P1 is greater than P2, then P3 must be greater than P4.

Constraint 4: (P1 = true || P2 >= 100) => (P3 = “ABC”), where P1 is a Boolean parameter, P2 is a parameter of type Number or Ranger, and P3 is of type Enum. This constraint specifies that if P1 is true and P2 is greater than or equal to 100, then P3 must be “ABC”.

A constraint can be directly typed in the Constraint Editor. The user is provided with the system configuration and the operators that can be used. The left hand side of the Constraint window displays the system configuration in a table format and the operators on the top of the system configuration table. Note that parameter values that are strings must be quoted in double quotes; otherwise, they will be considered as parameter names.

An existing constraint can be removed by selecting the constraint in the Added Constraint table and then clicking on the Remove button. Currently, ACTS does not allow an existing constraint to be directly edited. In order to edit an existing constraint, the user needs to remove the constraint first and then add the desired constraint as a new constraint.

3.2 Build Test Set

To build a test set for a system that is currently open, select the system in the System View, and then select menu *Operations -> Build*. The latter selection brings up the *Options* window, as shown in Fig. 6, which allows the following options to be specified for the build operation:

- *Algorithm*: This option decides which algorithm to be used for test generation. As mentioned in Section 1.1, IPOG, IPOG-F, IPOG-F2 and work best for systems of moderate size, while IPOG-D and PaintBall are preferred for larger systems. Note that relations and constraints are only supported for IPOG. By default, the IPOG algorithm is selected.
- *Max Tries*: This option is used by algorithm PaintBall, and it specifies the number of candidates to be generated randomly at each step.
- *Randomize Don't Care Values*: If this option is checked, then all the *don't care* values in the resulting test set will be replaced with a random value. By default this check box is unchecked.
- *Strength*: This option specifies the default strength of the test set. This strength is used for the default relation. (Recall that the default relation consists of all the parameters.) The user selects the strength from a drop-down list. Currently, ACTS supports a strength value ranging from 2 to 6. Also note that the user may create relations other than the default relation. Different strengths can be specified for those relations during their creation (in the Relation tab as discussed in Section 1.7.) For Base Choice algorithm the strength will be set to 1.
- *Mode*: This option can be *Scratch* or *Extend*. The former specifies that a test set should be built from scratch; the latter specifies that a test set should be built by extending an existing test set (shown in the Test Result tab). Recall that the current test set in the system may not be complete as the system configuration may have changed after the last build or the test set may be imported from outside.

- *Progress*: If this option is turned on, progress information will be displayed in the console. Note that in order to obtain the console, the GUI must be started from a command prompt instead of by double-clicking the executable jar file.

After the build operation is completed, the resulting test set will be displayed in the *Test Result* tab of the *Main* window.

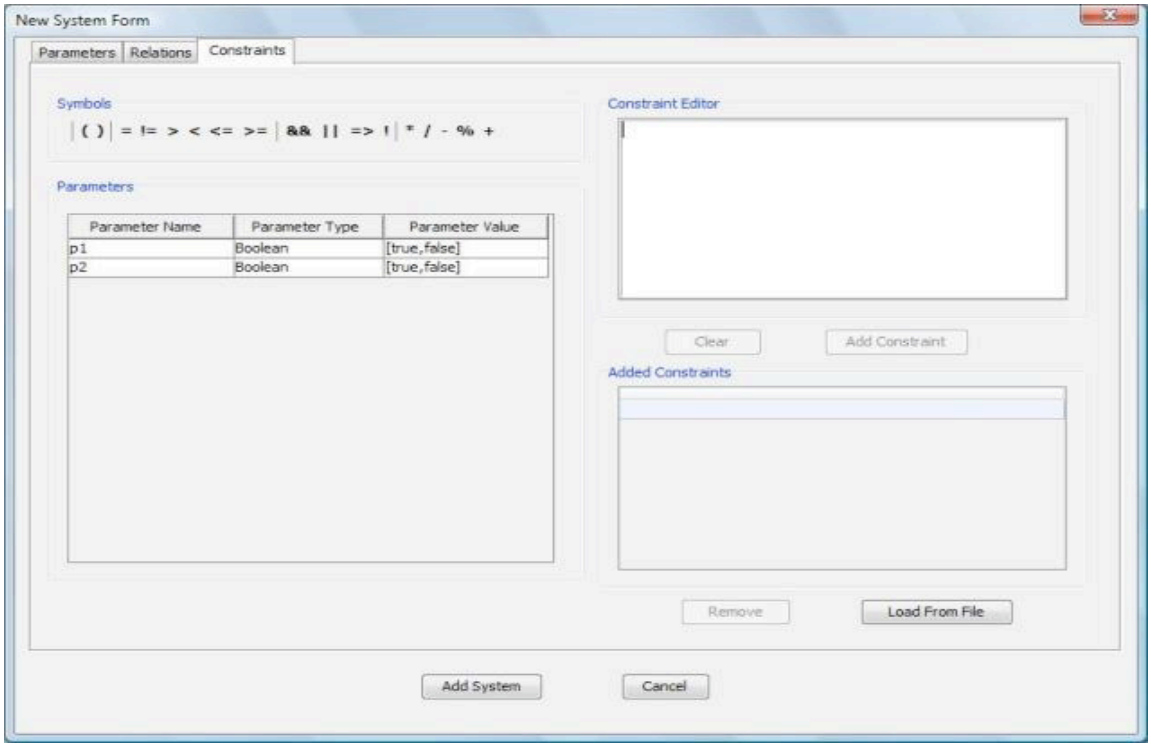

Figure 5. New System Window – Constraints

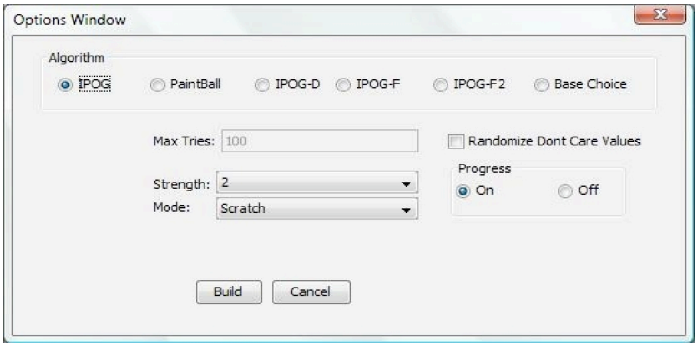

Figure 6. Build Options Window

3.3 Modify System

To modify an existing system, select the system in the tree view, and then select menu *Edit -> Modify*. The *Modify System* window is the same as the *New System* window

except that the name of the system cannot be changed. A parameter cannot be removed if it is involved in a relation other than the default relation or constraint. In this case, the parameter must be removed from the relation or constraint first.

A parameter or an output parameter can be added in the same way as during the New System operation. A parameter can be removed by selecting the parameter in the Saved Parameters table on the right hand side, and then clicking on the Remove button under the table. The values of a parameter can be modified by selecting the parameter on the Saved Parameters table on the right hand side, and by clicking on the Modify button under the table.

A system can also be modified through the tree view. For example, a parameter, or value, or relation, or constraint can be removed by first selecting the parameter, or value, or relation, or constraint, and then selecting menu *Edit -> Delete*.

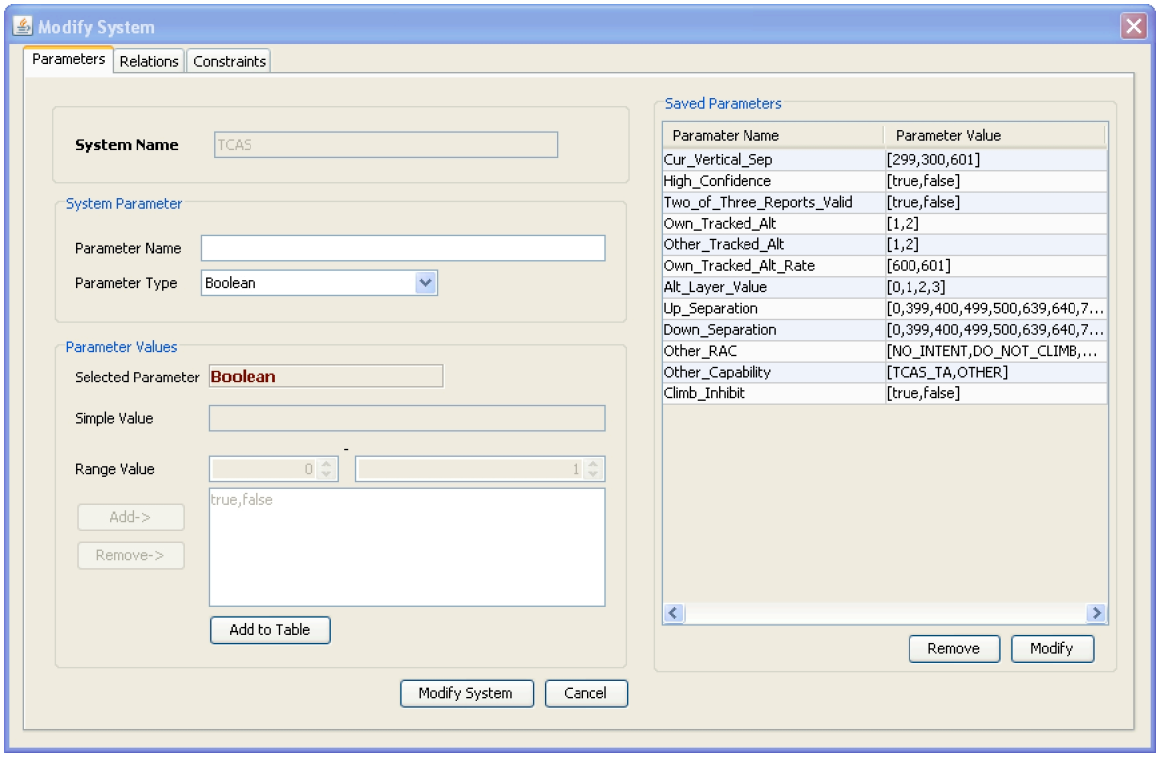

Figure 7. Modify System Window

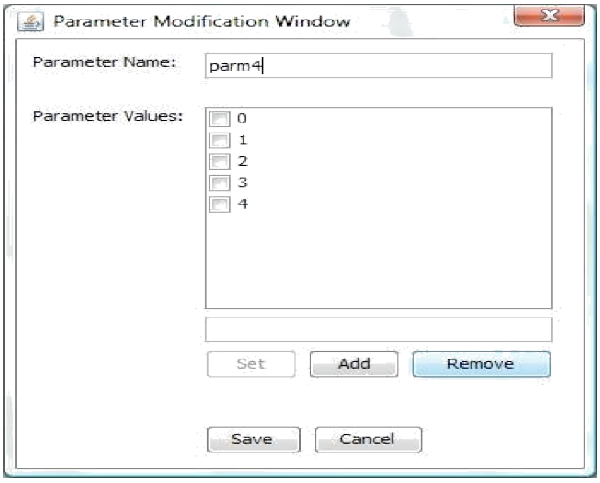

Figure 8. Parameter modification window.

3.4 Save/Save As/Open System

To save an existing system, select the system in the tree view, and then select menu *System -> Save* or *Save As*. When a newly created system is saved for the first time, or when *Save As* is select, a standard file dialog will be brought up, where the user can specify the name of the file to be saved, as shown in Fig. 9. The system will display a confirmation window if the file to be saved already exists.

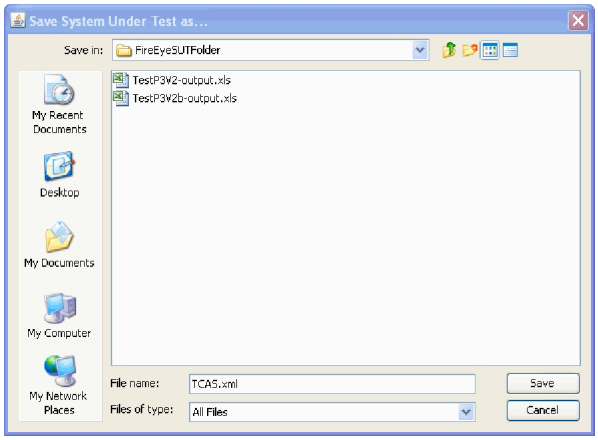

Figure 9. Save As Window

3.5 Import/Export Test Set

To import a test set of a system, the user must first create the system, in terms of adding its parameters and values into ACTS, as described in Section 3.1. Then, select menu *Operations -> Import*, as shown in Fig. 10, and select the format of the file containing the test set. Currently, two file formats are supported: CSV-R, which stands for Comma Separated Values with Row headers, and CSV-RC, which stands for Comma Separated

Values with Row and Column headers. (CSV-RC is mainly used to facilitate integration with Excel.) The following are two example files, one for each format:

CSV-R format:

P1,P2,P3,P4,P5
0,2,2,3,6
3,2,4,2,2
2,1,2,1,3
3,2,5,0,5

CSV-RC format:

,P1,P2,P3,P4,P5
Test1,0,2,2,3,6
Test2,3,2,4,2,2
Test3,2,1,2,1,3
Test4,3,2,5,0,5

The parameter values in each row must be separated by ",". There can be arbitrary space between two values. After the file format is selected, a standard file selection window appears through which the user can browse through the system and select the file containing the test set to be imported.

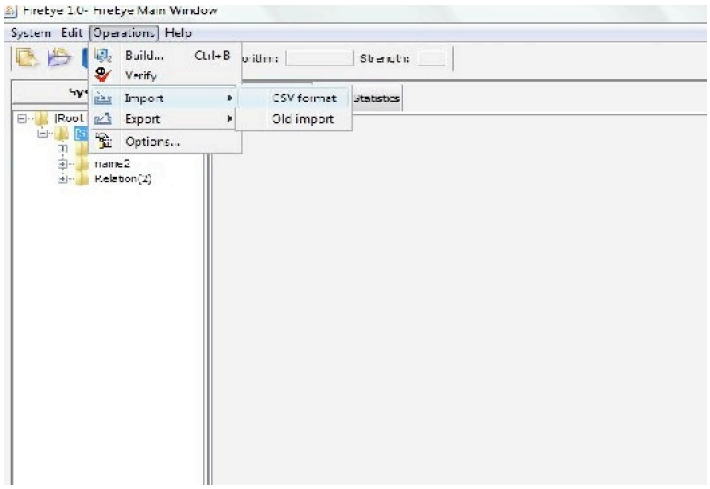

Figure 10 : Import test set window

To export a test set that exists in the GUI, first select the corresponding system so that the test set is displayed in the *Test Result* tab of the *Main* window, and then select *Operations -> Export*. Currently, three formats are supported, namely, NIST Format, Excel Format and CSV Format. A snippet of an exported test set in the NIST format is shown below:

Default degree of interaction coverage: 2
Number of parameters: 4
Number of configurations: 6

Parameters:
P1:[true, false]
P2:[true, false]
P3:[true, false]
P4:[true, false]

Relations :
[2,(P1, P2, P3, P4)]

----------Test Cases----------

Configuration #1:

1 = P1=true
2 = P2=true
3 = P3=true
4 = P4=true

Configuration #2:

….

A snippet of an exported test set in the CSV Format is shown below:

ACTS Test Suite Generation: Wed Jun 10 02:24:23 CDT 2009
* represents don't care value
Parameters:
P1:[true, false]
P2:[true, false]
P3:[true, false]
P4:[true, false]

Relations :
[2,(P1, P2, P3, P4)]

Tests
P1,P2,P3,P4
true,true,true,true
true,false,false,false

:

A snippet of an exported test set in the Excel Format is shown below:

Parameters:

P1	[true, false]
P2	[true, false]
P3	[true, false]
P4	[true, false]

Relations:
[2,(P1, P2, P3, P4)]

Test Case#	P1	P2	P3	P4
0	true	true	true	True
1	true	false	false	false

3.6 Verify T-Way Coverage

To verify the *t*-way coverage of a test set, the user can select menu *Operations -> Options*, and specify a desired strength in the *Options* window. Then, select menu *Operations -> Verify*. If the test set achieves the coverage for the specified strength, the message window in Fig. 11 will be displayed; otherwise, the message window in Fig. 12 will be displayed.

Note that this operation is typically used to verify the coverage of a test set that is imported from outside of ACTS.

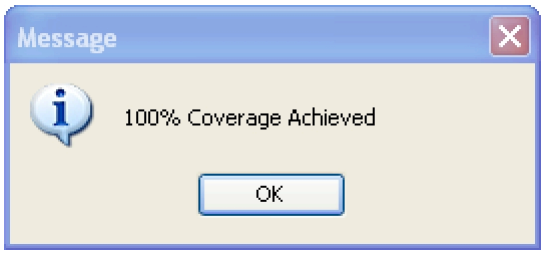

Figure 11. Coverage Achieved Window

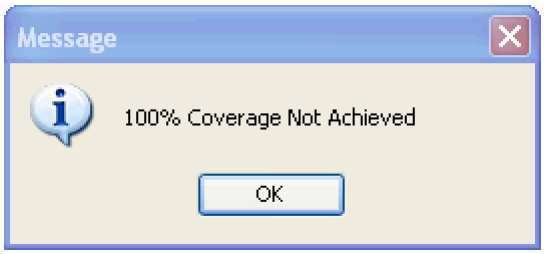

Figure 12. Coverage Not Achieved Window