The EUI Development Environment for the ERIGONE Model Checker
User's Guide and Software Documentation

Version 1.8.5

Mordechai (Moti) Ben-Ari
Department of Science Teaching
Weizmann Institute of Science
Rehovot 76100 Israel
http://stwww.weizmann.ac.il/g-cs/benari/

December 14, 2012

1 Introduction

EUI is a graphical user interface for the ERIGONE Model Checker that is used for simulating and verifying concurrent programs. It is an adaptation of the JSPIN interface for SPIN. The user interface of EUI is simple, consisting of a single window with menus, a toolbar and three adjustable text areas. ERIGONE option strings are automatically supplied and the ERIGONE output is filtered and formatted. All aspects of EUI are configurable: some at compile time, some at initialization through a configuration file and some at runtime.

References

- M. Ben-Ari. *Principles of the Spin Model Checker*. Springer, 2008.
- M. Ben-Ari. *The* ERIGONE *Model Checker*. `http://code.google.com/p/erigone/`.

2 Installation and execution

2.1 Installation

> The following description is for Windows. For other systems, download and open the EUI zip file and then copy ERIGONE executable to the EUI directory.

Install the Java SDK or JRE (`http://java.sun.com`).[1] EUI needs Java 1.5 at least.

Download the EUI installation file called `eui-N.exe`, where `N` is the version number, and execute the installation file. The installation will create the following subdirectories: `docs` for the documentation, `eui` for the source files, `txts` for the text files (help, about and copyright), and `examples`.

2.2 Execution

To run EUI, execute the command `javaw -jar EUI.jar`. An optional argument names the PROMELA file to be opened initially.

By default, an icon to run EUI is installed on the Windows Desktop. If Java is associated with `jar` files, double-clicking on the icon will run EUI.

[1] The default font for EUI is `Lucida Sans Typewriter`. This font may no longer be available in the JRE you use. If you have the fonts from a previous version you can copy them to the `lib/fonts` directory as explained in `http://java.sun.com/j2se/1.5.0/docs/guide/intl/font.html`. Alternatively, you can change the configuration file to use a monospaced font such as `Courier` that is installed by default.

2.3 Configuration

Configuration data (Appendix A) is in the file `config.cfg`. **When upgrading EUI, erase the configuration file before installing a new version, so that new configuration options will be recognized.** EUI searches for the configuration file in the current directory; if it is not found, EUI searches for it in the directory where the jar file is installed; if it is not found there, a new file with default values is written.

2.4 Building

To build EUI, run:

```
javac -target 1.5 eui\*.java
jar cfm EUI.jar eui\MANIFEST.MF eui\*.class
```

3 EUI user interface

The user interface includes a menu, a toolbar and three text areas. Menu and toolbar commands have keyboard mnemonics or accelerators. These can be easily configured by modifying the file `Config.java` and rebuilding.

The left text area is used to display PROMELA source files. The bottom text area is used to display messages from both ERIGONE and EUI. The right text area is used to display the output from ERIGONE: statements executed, values of variables and results of verifications. The text areas can be resized by dragging the dividers and an area can be hidden by clicking on the triangles in the dividers; the initial sizes can be set in the configuration file. The toolbar button Maximize (Alt-M) toggles between a normal split pane and a maximized right text area that displays the ERIGONE output.

3.1 Menu items

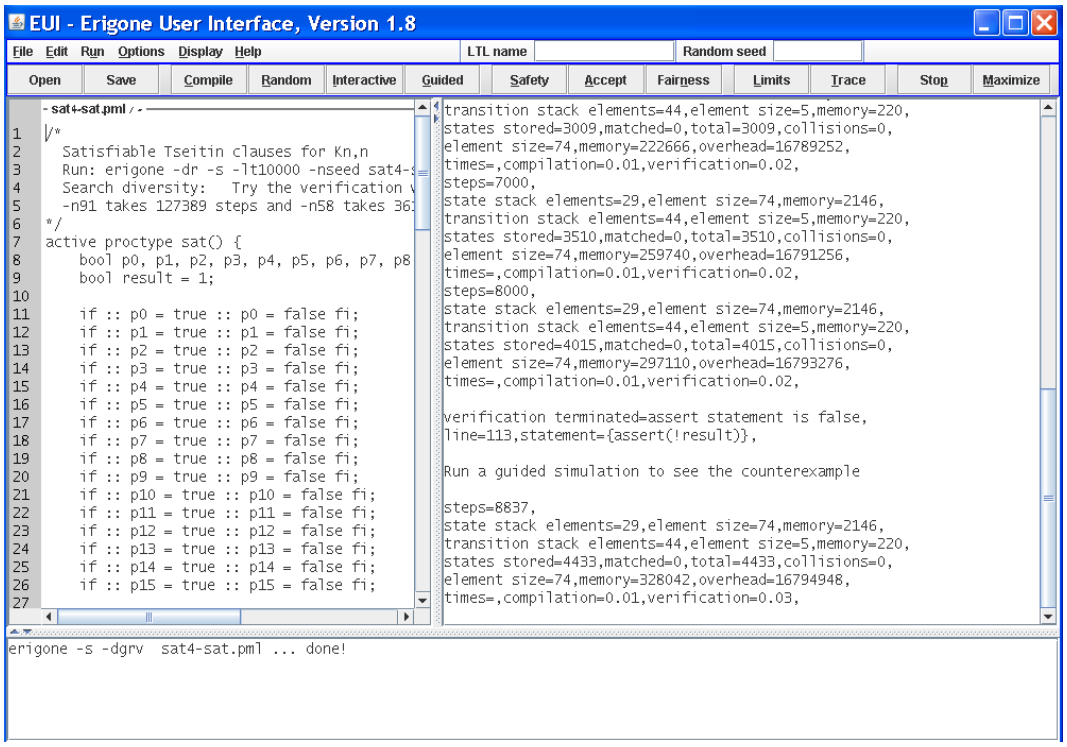
File This menu includes selections for New, Open, Save, Save As, and Exit. Switch file closes the current file and opens the last file that was edited, if any.

Edit This menu includes selections for Undo, Redo, Copy, Cut, Paste, Find and Find again.

Run This menu replicates the buttons on the toolbar for running ERIGONE in various modes. See the description in the next section.

Options The following ERIGONE Limits can be set (the values are in thousands):

- Total steps: The total number of steps in an execution of ERIGONE.
- Progress steps: A progress message will be displayed after this number of steps of a verification.
- State stack, Location stack: The size of the verification stacks.



Default restores the default values and Save saves the current options in the configuration file, together with the last directory from which a source file was opened, and the current values of the splitpane dividers. The file can be saved to the current or install directory.

Display This menu controls the display of the output in the right text area.

- Maximize replicates the button on the toolbar.
- Save output saves the contents of the text area in a file with extension `.out`.
- The Trace submenu is discussed in Section 4.

Help Help displays a short help file and About displays copyright information.

3.2 Textfields in the menu bar

The field LTL name is used to select a named LTL (inline) specification as described in Section 4.2.

A numeric (non-blank) value in the field Random seed will be used as the seed for generating random numbers, enabling a random simulation to be repeated. During verification it is used for search diversity. See the ERIGONE User's Guide for details.

4 Running ERIGONE

In the Run menu and on the toolbar are eight selections for running ERIGONE. They all use the PROMELA source file that has been opened, and save the file before execution. During simulation and verification, you can select Stop to terminate the ERIGONE process that has been forked.

Comple Runs the PROMELA compiler.

Random Runs a random simulation.

Interactive Runs an interactive simulation.

Guided Runs a guided simulation using the trail file created for a counterexample found in a verification.

Safety Runs a safety verification.

Accept Runs an acceptance verification.

Fairness Runs an acceptance verification with weak fairness.

If you terminate EUI while ERIGONE is running (for example by entering ctrl-C at the command line), make sure to terminate the ERIGONE process as well. In Windows, press ctrl-alt-del, Task List and Processes. Select erigone.exe and End Process.

4.1 Simulation

4.1.1 Interactive simulation

During an interactive simulation, if there is more than one executable statement or expression in a state, a dialog frame will pop up with the current values of the variables and a list of the statements and expressions that can be executed. The list can be displayed either as a row of buttons, or—if there is not enough room—as a pulldown menu. The choice of the format is determined by the value of the configuration option `SELECT_MENU`. There are also configuration options for setting the width and height of the buttons or menu items.

The dialog can be navigated using the mouse; closing the dialog frame will terminate interactive simulation. For keyboard navigation:

Buttons Tab and Shift-Tab move through the buttons and Space selects the highlighted button. Press Esc to terminate.

Menu Press the Down arrow to display the list and to highlight the item you want; press Return to select it. Press Esc to terminate.

4.1.2 Filtered output

The contents of the ERIGONE output can be changed by selecting Display/Trace. This pops up a dialog that can be used to set the width of the field for the statement executed and the width of the field for each variable. There are text areas for entering a list of strings defining variables to be *excluded* from the display. Any variable containing a string from the list is not displayed; for example, `want` will exclude all variables that have `want` as a substring. If the variable name is prefixed by `+`, it will be included anyway. For example, if you have an array variable `test`, then the entries `test` and `+[1]` will exclude display of the elements of `test` except for `test[1]`. The list is saved in a file with extension `.exc`.

Similarly, a file of excluded statements can be created; a statement is excluded if it contains one of the substrings in the file. The same convention for the prefix `+` enables inclusions of strings that would otherwise be excluded. The file extension is `.exs`. Exclude statements should *not* be used with interactive simulation.

When a non-active `proctype` is used, the local variables such as `P.temp` are only formal names. When `run` creates a process, actual variables names `P_1.temp`, `P_2.temp`, ... are created and the formal variable is removed from the display.

4.2 LTL formulas

A correctness claim is specified by a formula in *Linear Temporal Logic (LTL)*. The formula is written inline within the program:[2]

```
ltl { [](critical <= 1) }
```

If there are named LTL formulas within the program:

```
ltl mutex { []!(csp && csq) }
ltl nostarve { []<>csp && []<>csq }
```

you can enter the name of the formula to be used in the field LTL name.

ERIGONE requires a LTL formula for verification of *acceptance* (with or without fairness). A formula is optional for verifying safety; if a formula is not needed, be sure to erase any data in the LTL field.

[2] ERIGONE supports specifying LTL formulas in an external file, but this is not supported in EUI.

5 Software structure

`EUI` is the main class and contains declarations of the GUI components and instances of the classes `Editor` and `RunSpin`. Method `init` and the methods it calls initialize the GUI. Method `actionPerformed` is the event handler for all the menu items and buttons; it calls the appropriate methods for handling each event. The About and Help options are implemented by reading files `copyright.txt` and `help.txt`, respectively, and displaying them in the righthand pane.

`Filter` performs the filtering of the ERIGONE output. Since it is called for each line separately there are quite a number of flags to maintain state between lines. `ArrayLists` are used to store the excluded variables and statements, while `variables` of type `TreeMap` is used to map variable names into values. Whenever a state is displayed in the output (for example, `"temp=6,"`, `storeVariables` is called to search for the line for each key in the map and set the value associated with the name. `variablesToString` iterates over the map to create a string of the values of the variables, while `collectionToString` creates the title lines with the variable names. The actual processing of the output is performed in three separate methods, one for compilation, one for simulation and one for verification.

`EUIFileFilter` is used with a `JFileChooser` when opening and saving files: PROMELA source files, LTL property files and ERIGONE save output files.

`Limits` is the dialog frame for Options/Limits and `Trace` is the dialog frame for editing the list of variable strings to be excluded from the display.

`Config` contains declarations of compile time configuration items. Method `init` calls `setDefaultProperties` to initialize the instance of `Properties` with the default values of the dynamic configuration items; it then attempts to load the configuration file, and if unsuccessful, the default properties are written to a new file.

`Editor` implements an editor using operations on a `JTextArea`. It implements the interface `DocumentListener` to flag when the source has been modified. The class is also responsible for the LTL formula `JTextField`. `EUI` calls method `writeFile` to write `out` files, and method `readFile` to read the text files to be displayed.

`LineNumbers` extends a `JComponent` to create line numbers for the `RowHeaderView` of the editor `JScrollPane` (thanks to Niko Myller for this code).

`UndoRedo` was extracted from an example on the Java web site.

The event handler in `EUI` calls `run` in class `RunSpin` to execute ERIGONE. `run` creates a thread of class `RunThread`, and uses `ProcessBuilder` to set up the command, directory, merge the output and error streams, and set up streams for I/O. The call `runAndWait` is used for short calls like Compile; this call does not return until the completion of the subprocess. The call `run` will return immediately after it has created the thread. In this case, the event handler in `EUI` calls `isSpinRunning` to create a thread to poll for termination of ERIGONE; by creating a separate thread, the event handler is freed to accept a selection of Stop.

When more than one executable transition occurs during an interactive simulation, the method `select` is called. This method pops up a dialog to enable the user to make a selection. A `JFrame` is created in a new thread of the inner class `SelectDialog` to wait for the selection. `select` polls `selectedValue` which is set with the selected button value or zero if the frame is closed or Esc pressed. In that case, `q` is sent to ERIGONE to terminate the simulation.

A Configuration file

These tables give the properties in the configuration file and their default values.

Directories and files	
SOURCE_DIRECTORY	examples
ERIGONE	erigone.exe
HELP_FILE_NAME	txt\help.txt
ABOUT_FILE_NAME	txt\copyright.txt

Options for executing ERIGONE	
COMPILE_OPTIONS	-c -dbprv
RANDOM_OPTIONS	-r -dcmoprv
INTERACTIVE_OPTIONS	-i -dcemoprv
TRAIL_OPTIONS	-g -dcmoprv
SAFETY_OPTIONS	-s -dgrv
ACCEPT_OPTIONS	-a -t -dgrv
FAIRNESS_OPTIONS	-f -t -dgrv
TOTAL_STEPS	100
PROGRESS_STEPS	1
STATE_STACK	2
LOCATION_STACK	3
SEED	0
SINGLE_QUOTE	false

Trace options	
PROCESS_WIDTH	7
STATEMENT_WIDTH	18
VARIABLE_WIDTH	10
LINES_PER_TITLE	20

Text settings	
WRAP	true
TAB_SIZE	4
FONT_FAMILY	Lucida Sans Typewriter
FONT_STYLE	java.awt.Font.PLAIN
FONT_SIZE	14

Frame size	
WIDTH	1000
HEIGHT	700

Interactive dialog settings	
SELECT_BUTTON	120
SELECT_HEIGHT	70
SELECT_MENU	5

Location of dividers	
LR_DIVIDER	400
TB_DIVIDER	500
MIN_DIVIDER	50

Delay while waiting for user input	
POLLING_DELAY	200

Display ERIGONE output that is piped to EUI	
DEBUG	false