ESBMC Plug-in for Eclipse

The ESBMC plug-in is developed in Eclipse Helios, release 3.6 with the JRE 1.6 running on a Linux operating system. Before you start using it, make sure that you have installed on your machine JDK v1.6 (or higher). At this moment, we do not support Windows and Mac distributions.

This document is intended for the plug-in users and is thus split into three main sections as follows: *how to install, how to use,* and *how to uninstall the ESBMC plug-in*.

1. How to install the ESBMC plug-in

There are two different ways to install the plug-in depending on the package that you have downloaded (i.e., jar or zip file) from the website http://users.ecs.soton.ac.uk/lcc08r/esbmc/.

1.1.Plug-in Package (jar file)

In this case, the installation process consists of three simple steps:

1) Extract the package on your file system;
2) Copy the jar file to your */Eclipse_home/plugins/* directory (see Figure 1);
3) Restart Eclipse.

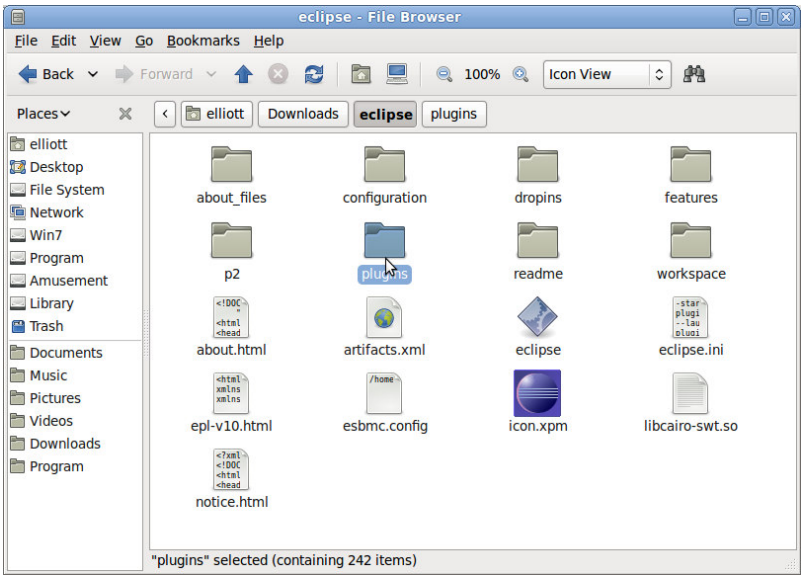

Figure 1: Structure of the Eclipse (Helios) Directory

1.2.Site Package (zip file)

Download the plug-in site package, and then extract it to your local file system. After that, you should open Eclipse, and click on the tool bar *Help->Install New Software* (See Figure 2).

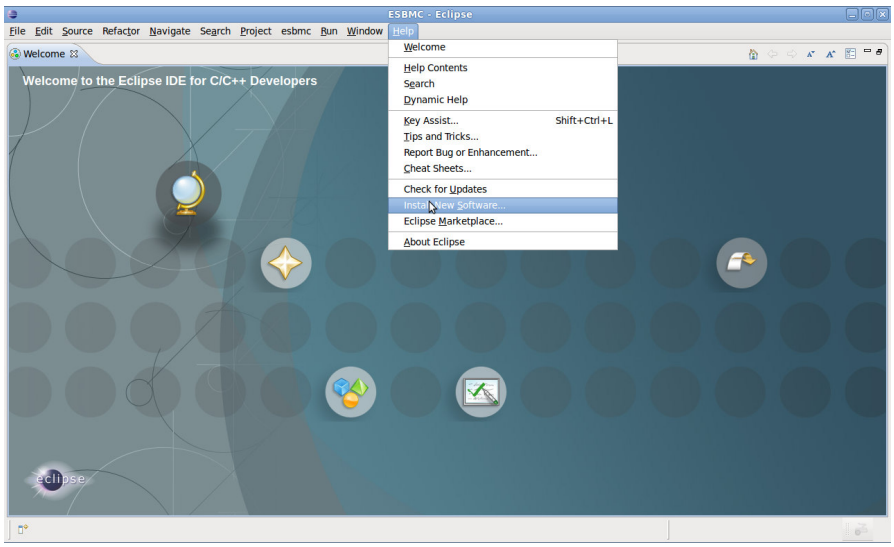

Figure 2: Install New Software

After clicking on *Install New Software*, you will see the wizard to install the plug-in as shown in Figure 3.

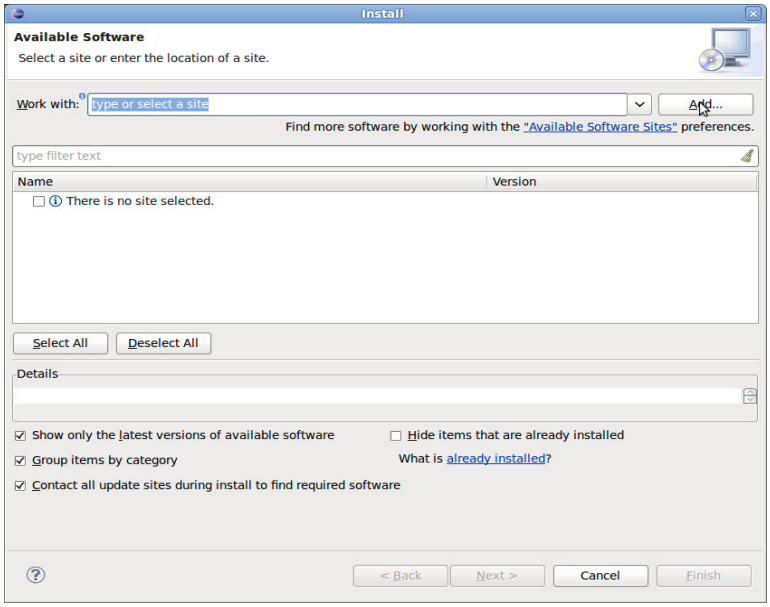

Figure 3: Available Software

You should now click on *Add...* and then on *Local...* so that you can browse and choose the plug-in package (see Figure 4). Alternatively, you can also type in the location field the address of the ESBMC plug-in website: http://users.ecs.soton.ac.uk/lcc08r/esbmc/ESBMCPlugin.

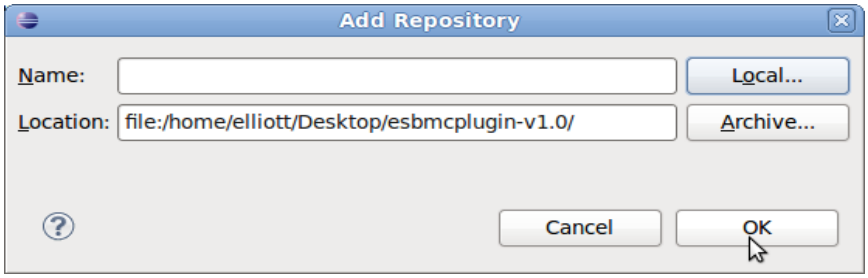

Figure 4: Add Repository

Note that you can also choose the name of the plug-in (e.g., *Esbmcfeature*) that you will install on your machine as shown in Figure 5. After selecting the plug-in package, you should click on the *Next* button to continue with the installation process.

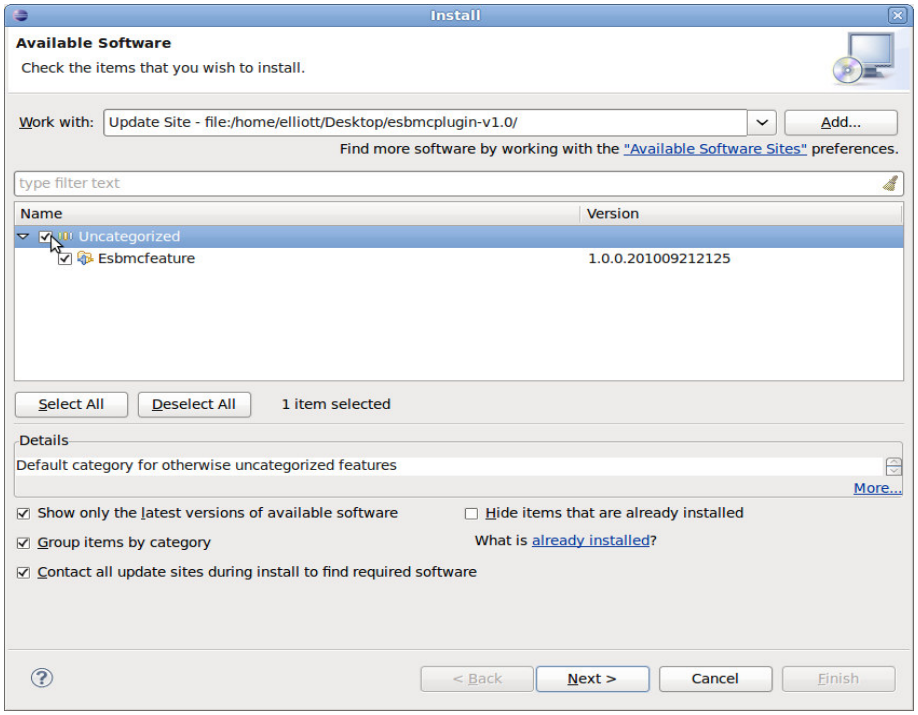

Figure 5: Install

After clicking on *Next*, you should read carefully the license and if you agree, check the *I accept the terms of the license agreement.* Finally, in order to start installing the plug-in, you should click on *Finish* (see Figure 6).

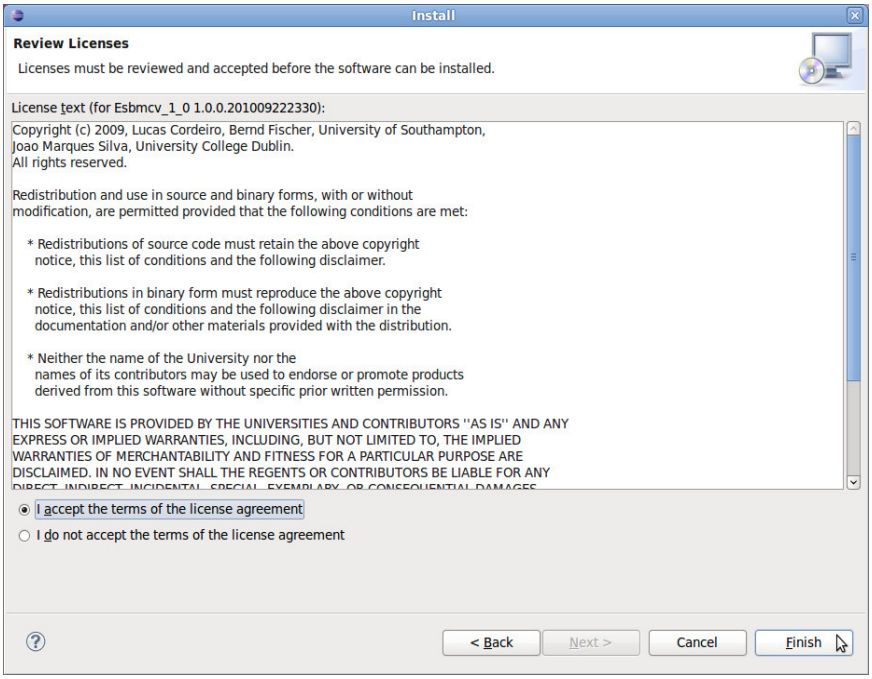

Figure 6: License agreement

After accepting the license terms, the installation process of the ESBMC plug-in is started. After installation, a certificate message will be shown. To proceed, you should accept the certificate and click on *OK* to complete the installation (see Figure 6).

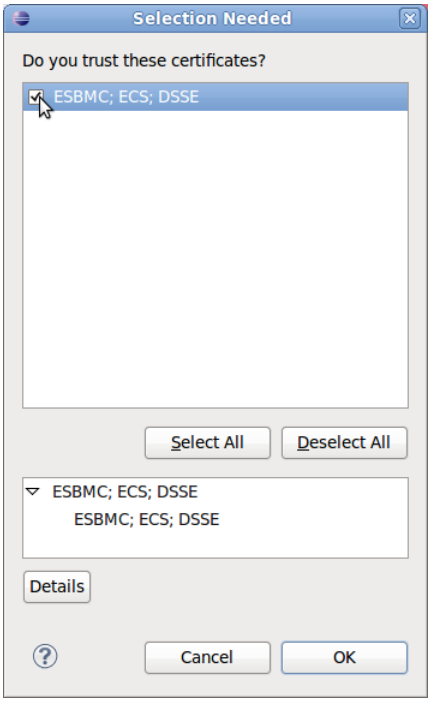

Figure 6: Certificate selection

You finally need to restart Eclipse for the changes to take effect.

1.3.Initialization

After restarting Eclipse, you will see on your main tool bar the ESBMC button with the message *Configure And Run ESBMC*, which indicates that you have correctly installed the ESBMC plug-in (see Figure 7).

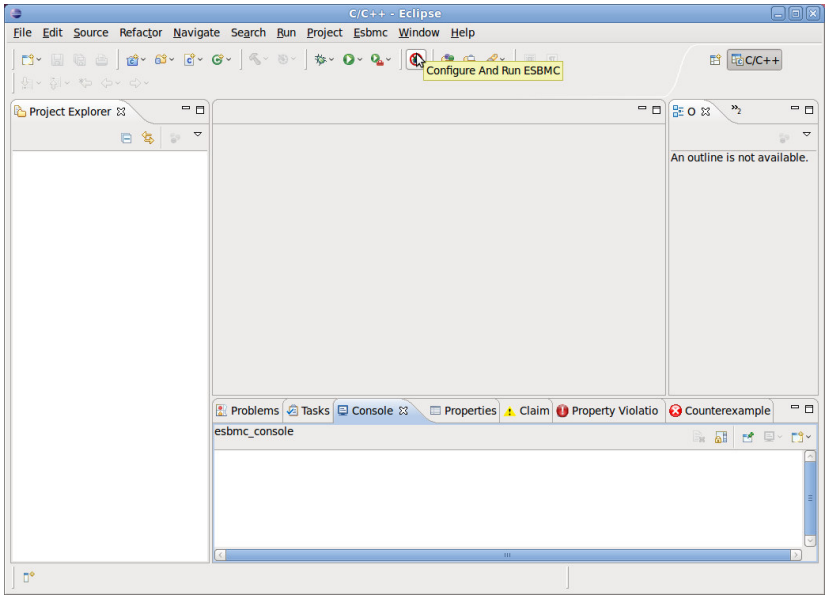

Figure 7: ESBMC button in the Eclipse tool bar

Alternatively, you can also configure and run ESBMC by clicking on the menu bar, where you can find two options: *Verify Current File* and *Configure And Run ESBMC,* as shown in Figure 8. *If you cannot use the plug-in by selecting it in the tool or menu bar, please check the version of your JRE.*

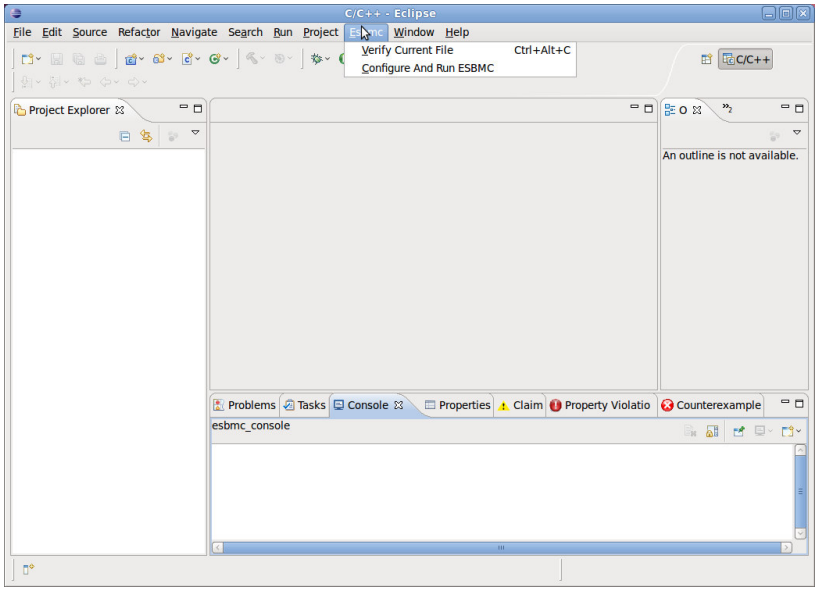

Figure 8: ESBMC entry in Eclipse menu bar

After clicking on the ESBMC button, you should set the path that leads to the installation of the ESBMC model checker, as shown in Figure 9.

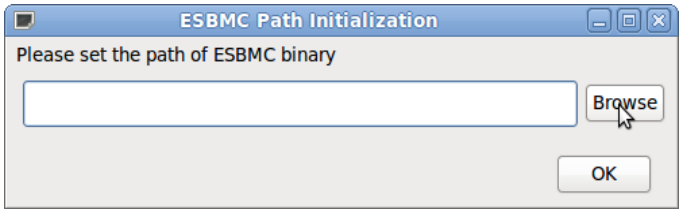

Figure 9: ESBMC path selection

You can click on the *Browse* button and set ESBMC path (i.e., choose the directory in which the ESBMC binary file is located). After that, you should click on *OK,* which will open the configuration window of the ESBMC model checker (see Figure 10). Note that once you set the ESBMC path, it will be stored in a file so that you do not need to set it again when you restart the Eclipse environment.

2. How to use the ESBMC plug-in

The ESBMC model checker can be started in two different ways:

- By clicking on the ESBMC button or selecting the *Esbmc ->Configure And Run ESBMC* menu item entry, which allows you to change the file and configuration parameters of the model checker.
- By selecting *ESBMC->Verify Current File*, it is run on the current file with the current settings of the options.

2.1.ESBMC Run-time Option Configuration Window

This window comprises five tabs that allow you to set the different run-time options of ESBMC.

2.1.1. Front-end Options

Figure 10 shows the options available in the ESBMC front-end.

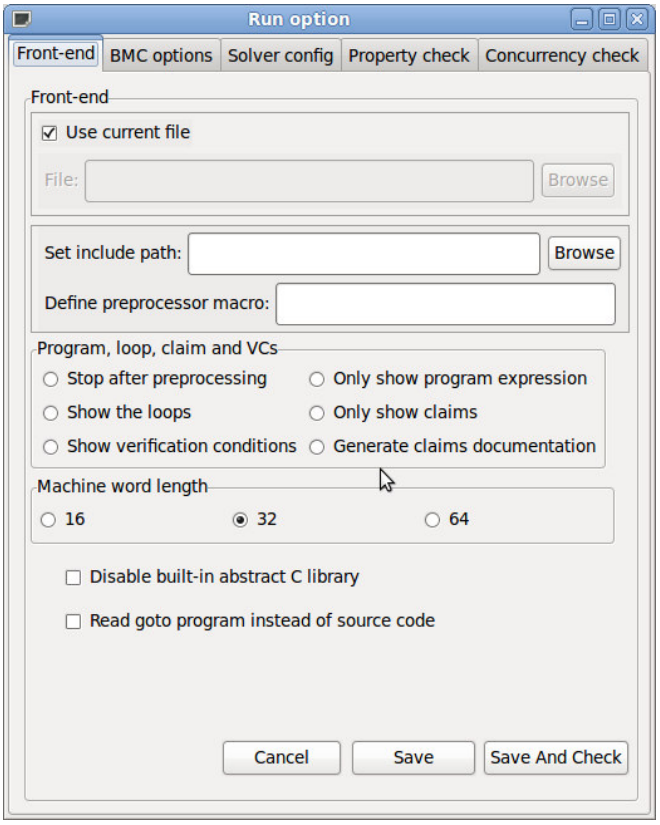

Figure 10: Front-end options

Use current file: You can analyze the file that is open in your current editor. If you want to analyze other files located in your file system, then uncheck the box *Use current file* and click on *Browse* to choose the program that you want to analyze. If your application consists of more than a single C program, then you can specify them as a sequence (e.g., */home/esbmc/file1.c file2.c*).

Set include path: You can set the include path, which contains the .h files, by clicking on the *Browse* button.

Define preprocessor macro: You can define C preprocessor macro in this text area by just providing the name without # as directives.

Program, loop, claim and VCs: You can choose the options to show the preprocessed program, all the claims (or properties) given as assertions by the designers as well as a range of language-specific safety (such as the absence of arithmetic under- and overflow, out-of-bounds array indexing, or nil-pointer dereferencing), show the verifications conditions that are generated during BMC, the identification of the loops in the program, the expressions of the program in single static assignment (SSA) form, and the documentation (in Latex) of the generated claims. However, note that all these options are mutually exclusive, because they produce an output to the same (Console) view, i.e., you can visualize one of them on each time.

Machine word length: You can set your machine word length, the default is 32.

Disable built-in abstract C library: The C programs usually use functions of the ANSI-C library (e.g., *strcmp, printf*), which contain information that are irrelevant from the verification point of view. We thus provide an abstract ANSI-C library implemented internally in the model checker, which comprises a small set of the functions. If you do not want to use the built-in ANSI-C library, then you should select this option.

Read goto program instead of source code: This option allows you to model check the goto programs (i.e., control-flow graphs) generated by the *goto-cc* tool (available for downloading at http://www.cprover.org/goto-cc/).

2.1.2. BMC Options

The BMC options of the ESBMC plug-in are shown in Figure 11. It consists of the following features:

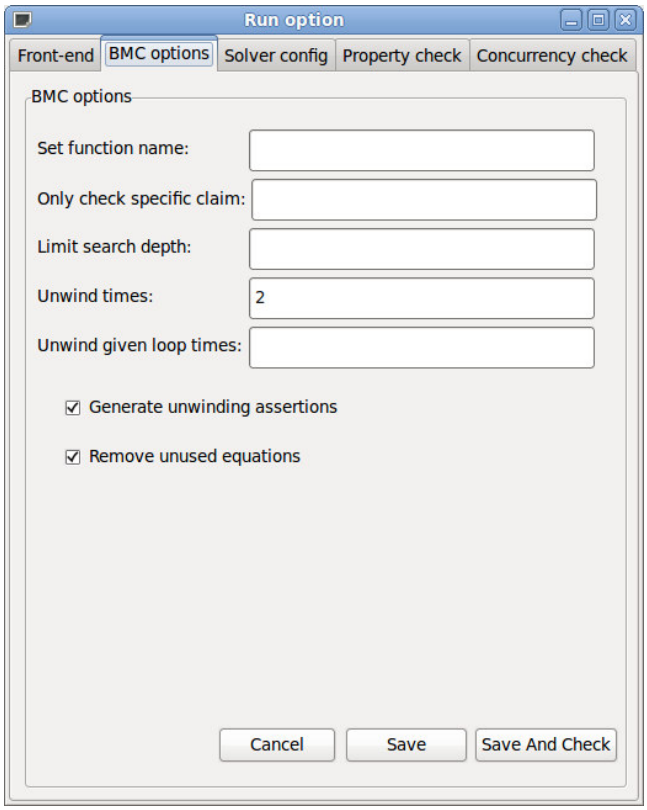

Figure 11: BMC options

Set function name: You can set the main function name here.

Only check specific claim: You can check for a specific claim, so please input the number of the identification of the claim.

Limit search depth: You can limit search depth, so please input an integer number.

Unwind times: Set the unwind bound in here, the default is 2. You have to provide an integer number.

Unwind given loop times: This option allows you to unwind a specific loop in your program. Here, you should provide the identification of the loop.

Do not generate unwinding assertions: If you do not want to check that you have unrolled enough the loops in your program, then you should select this option.

Do not remove unused equations: If it is unchecked, unused equations are removed automatically during the symbolic execution.

2.1.3. SMT Solver Configuration

The solver configuration options tab is shown in Figure 12. It consists of the following options:

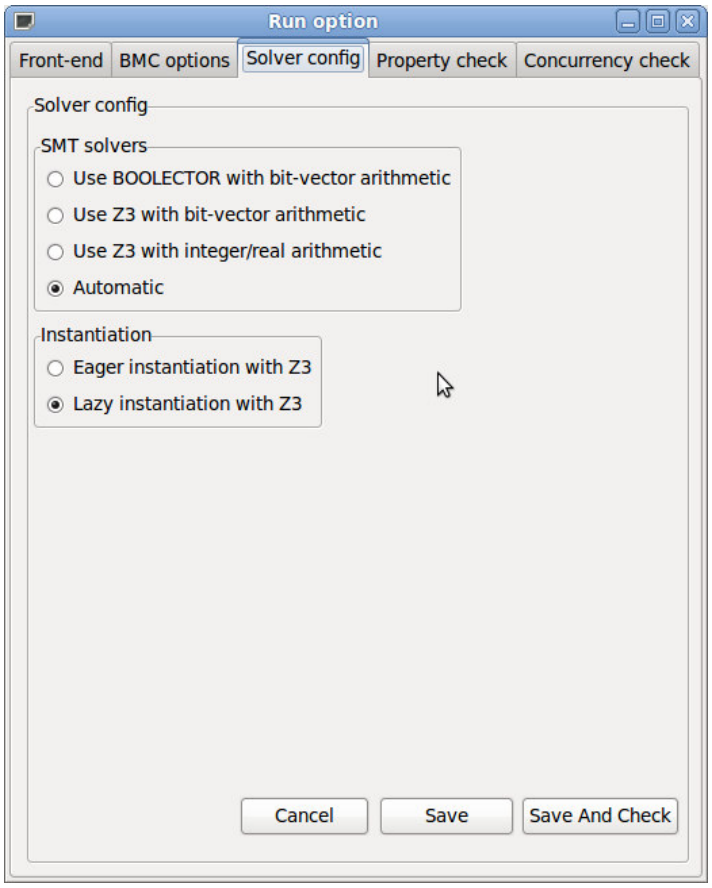

Figure 12: SMT Solver Configuration

SMT solvers: If you choose the first one, the model checker will use BOOLECTOR with bit-vector arithmetic as a decision procedure to model check your program. The second option is to use Z3 with bit-vector arithmetic, and in the third ESBMC is Z3 with integer/real arithmetic. The last option, which is the default option, will determine the best solver and encoding to be used according to the verification conditions that are generated from your C program.

Instantiation: You can choose either eager or lazy instantiation to solve the SMT instances with Z3 (*lazy* is the default option).

2.1.4. Property Check

You can select which properties you want to check in your program as show in Figure 13. This tab consists of the following options:

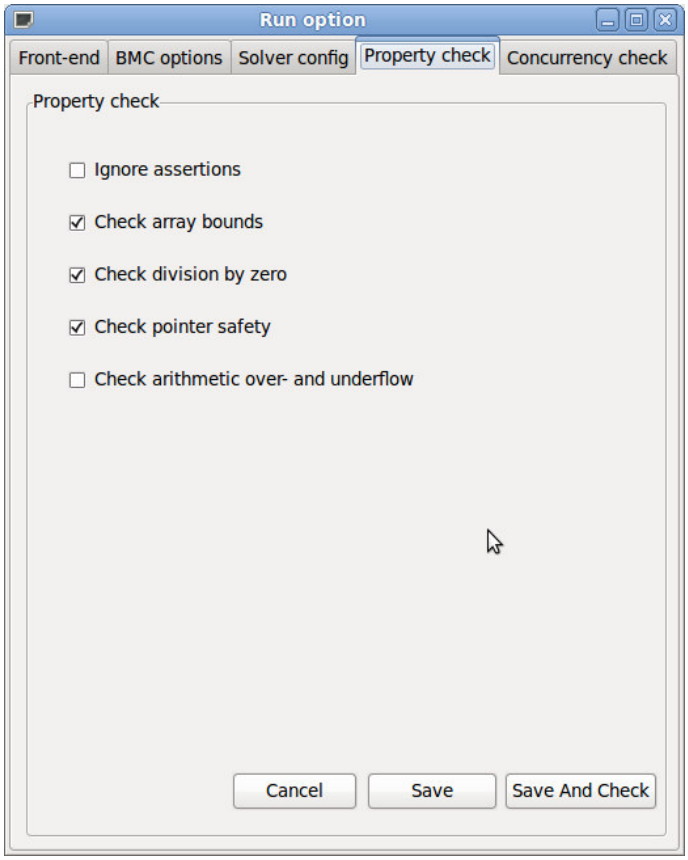

Figure 13: Property check

Ignore assertions: This option ignores all assertions in your C program.

Do not do array bounds check: This option does not allow ESBMC to generate verification conditions related to checking out-of-bounds array indexing.

Do not do division by zero check: This option does not allow ESBMC to generate verification conditions related to checking division by zero in arithmetic expressions.

Do not do pointer check: This option does not allow ESBMC to generate verification conditions related to checking nil-pointer dereferencing.

Enable arithmetic over- and underflow check: This option does not allow ESBMC to generate verification conditions related to checking arithmetic over- and underflow.

2.1.5. Concurrency Check

You can select which approaches and properties you want in order to verify in your multi-threaded program as shown in Figure 14.

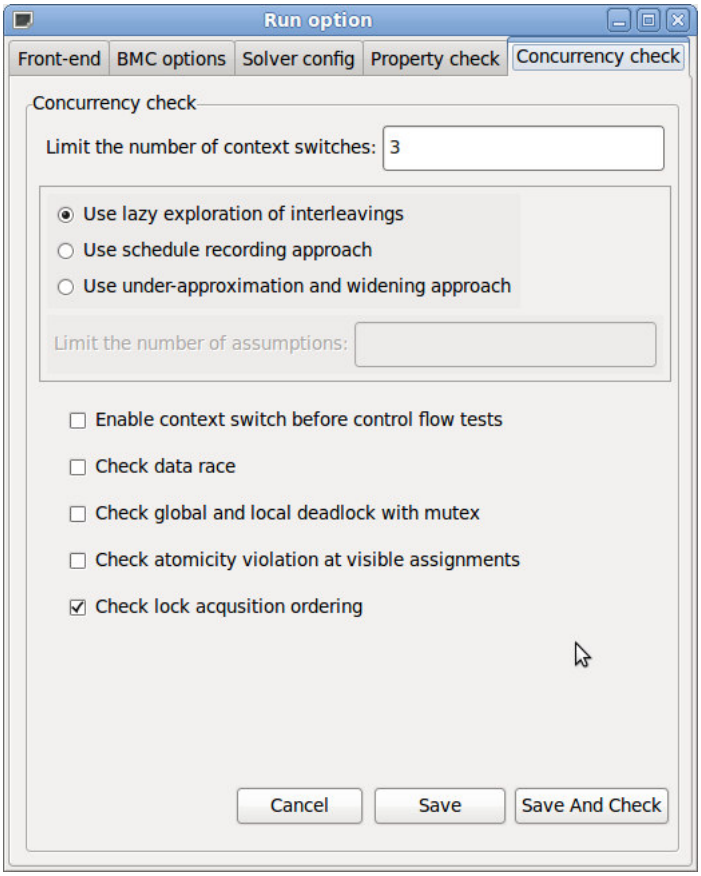

Figure 14: Concurrency check

Limit the number of context switches: Limit the number of context switches allowed per each thread. You have to provide an integer number here.

Use schedule recording approach: This option allows ESBMC to encode all possible interleavings into one single formula and then exploit the high speed of the SMT solvers.

Use under-approximation and winding approach: This option allows ESBMC to check models with an increasing set of allowed interleavings.

Limit the number of assumptions: If you choose *Use under-approximation and winding approach*, then you can limit the number of assumptions in the UW approach. You have to provide an integer number in the text area.

Enable global and local deadlock check with mutex: This option checks whether all threads wait for a mutex (global deadlock) or whether some of the threads form a waiting cycle (local deadlock).

Enable data races check: This option checks whether multiple threads perform unsynchronized accesses to shared data.

Do not do lock acquisition ordering check:
This option checks for unintended sequence of lock and unlock operations among the threads.

Enable atomicity violation check at visible assignments: This option allows ESBMC to break visible statements to check if a region of code executes atomically.

Enable context switch before control flow tests: This option allows ESBMC to simulate the effect of a context switch right after a visible test by hoisting the test out of the conditional, and assigning its result to a new auxiliary variable.

2.2.Model Checking the Program

In order to model check your C program, you should click on the *Verify Current File* menu item or simply type the shortcut *CTRL+ALT+C*. After that, you should (wait for a moment to) see the results as shown in Figure 15. You can also check all parameters that were passed to the ESBMC model checker as well as the verification results. However, note that for large programs, the verification process might time a long time.

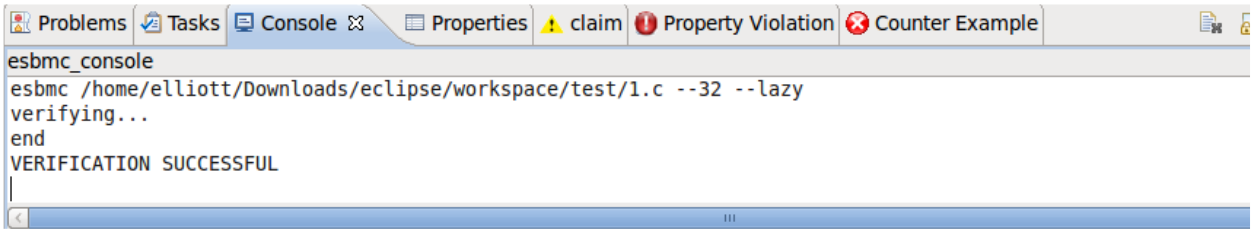

Figure 15: Results of the ESBMC verification

2.3.Counterexample, Property Violation, and Claim Views

When the verification fails (i.e., the property does not hold in the program), you can see details of the *property violation* and *counterexample* (or trace to reproduce the violation) in the corresponding views as shown in the bottom of Figure 16.

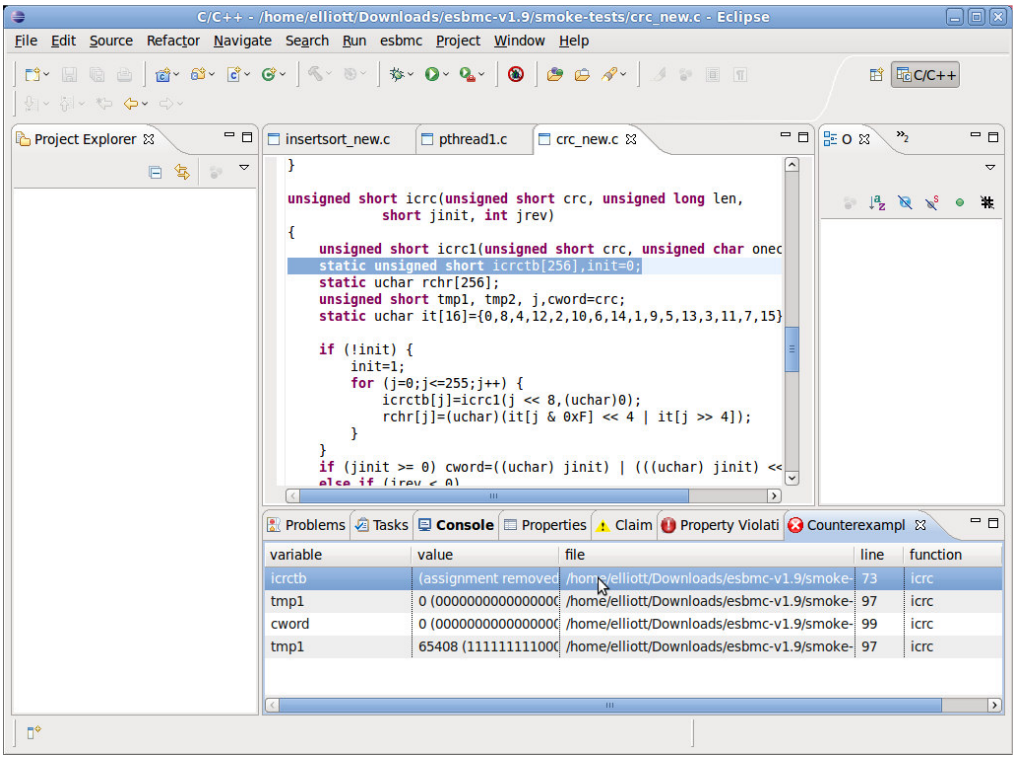

Figure 16: Counterexample view

If you double click on the variable name in the counterexample view, then you go directly to the corresponding line in the program where the error is located. The *property violation* and *claim* views work in the same way as in the *counterexample* view, i.e., you should click in one line of the table in order to go directly to the corresponding line in the program. If you need to obtain more information about the results of other options of the ESBMC model checker (e.g., show program only, show loops), then you can easily visualize them in the console, as shown in Figure 17 (for the option program only).

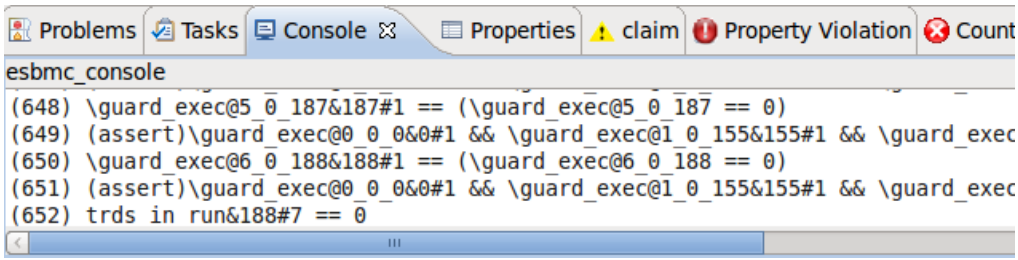

Figure 17: Additional information in the console (program only option)

3. How to uninstall the ESBMC plug-in

In order to uninstall the ESBMC plug-in, you should first click on *Help* and then on *Install New Software* (see Figure 1). After that, you should click on *What is already installed* in the available software window as shown in the bottom of Figure 18.

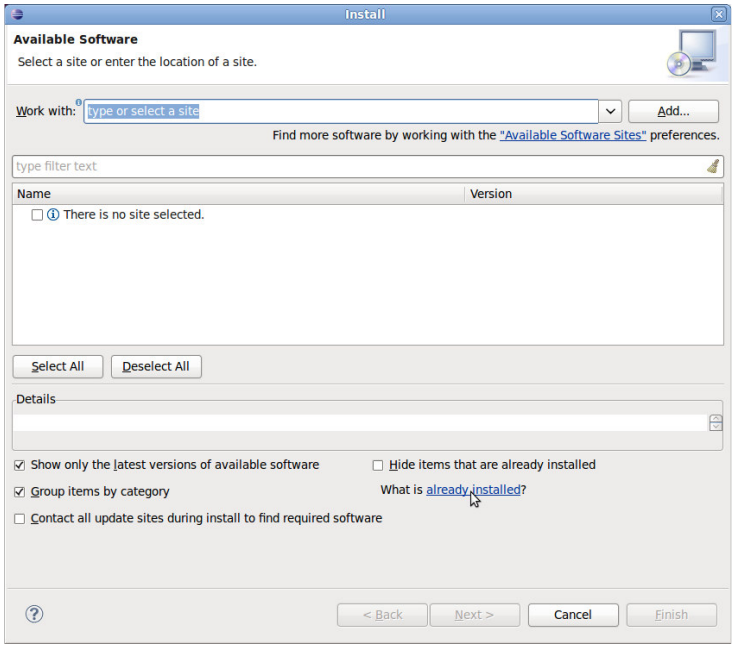

Figure 18: What is already installed?

You should then select the *Esbmcfeature* option (or whatever name you choose in the installation) in the *installed software* tab as shown in Figure 19.

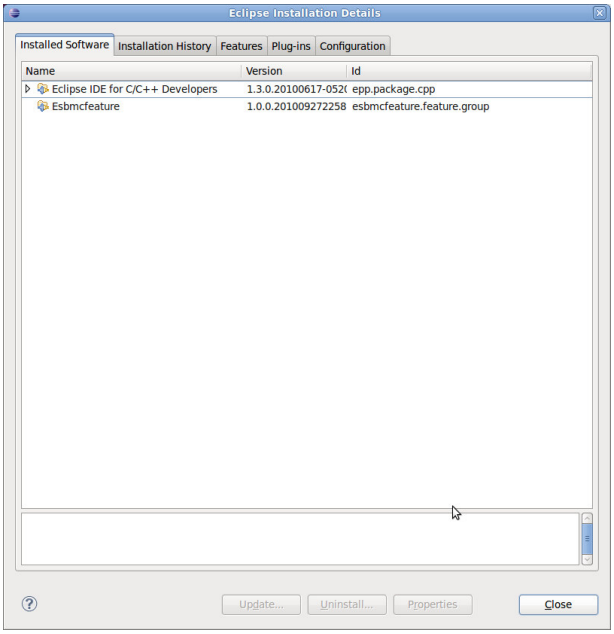

Figure 19: Installed Software

Finally, you should click on the *finish* button in the *Uninstall Details* window to uninstall ESBMC as shown in Figure 20.

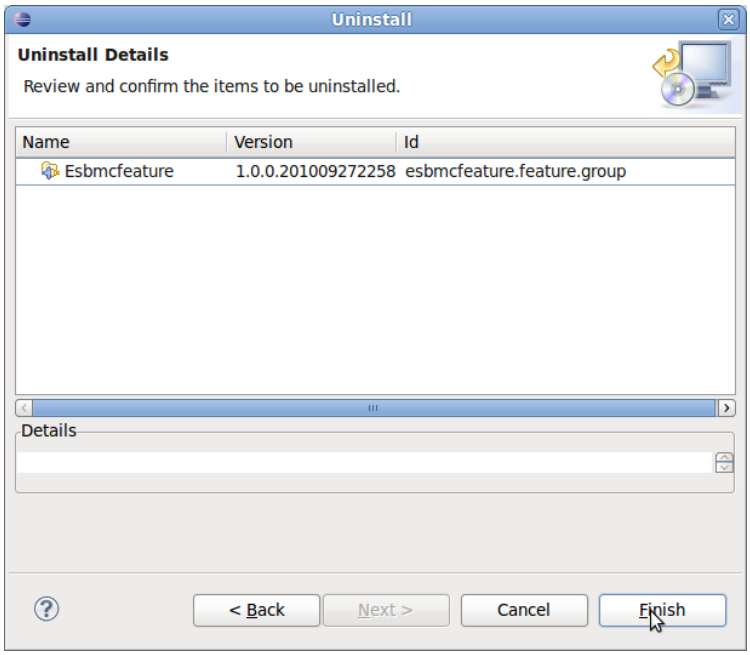

Figure 20: Uninstall Details

4. Contact Information

For any further questions about the ESBMC plug-in, please send an e-mail to the following address esbmc@ecs.soton.ac.uk.

5. Acknowledgments

This work was supported by the School of Electronics and Computer Science (ECS) at the University of Southampton. We thank Qiang Li for developing this plug-in during his internship at ECS.