**User Guide**

**Individual statement icons**

|  |  |  |
| --- | --- | --- |
| C:\Users\manios\Documents\Eclipse\CodeAnalyser\data\images\trivial.png | Trivial statement | Such statements are considered implementable. |
| C:\Users\manios\Documents\Eclipse\CodeAnalyser\data\images\sourceCode.png | Python code | The only condition to be met for an expression to be marked as code is for the expression to contain a ‘:’ character. |
| C:\Users\manios\Documents\Eclipse\CodeAnalyser\data\images\nonTrivial.png | Non-Implementable statement | A problem is considered implementable only if it does not contain any non-implementable icon. |
| C:\Users\manios\Documents\Eclipse\CodeAnalyser\data\images\invalid.png | Invalid statement | A statement is invalid if some declared variables are not used or there are duplicate variables declared. |
| C:\Users\manios\Documents\Eclipse\CodeAnalyser\data\images\nonAffecting.png | Not affecting output | A statement that does not affect the output can be ignored. |

**Syntax highlighting**

The following syntax highlighting conventions are used:

Blue comparisons () are invertible and transitional logical comparisons between variables. For comparisons that are not blue, the algorithm will not take advantage of their individual transitional properties or of possible inversion of the order between the two sides.

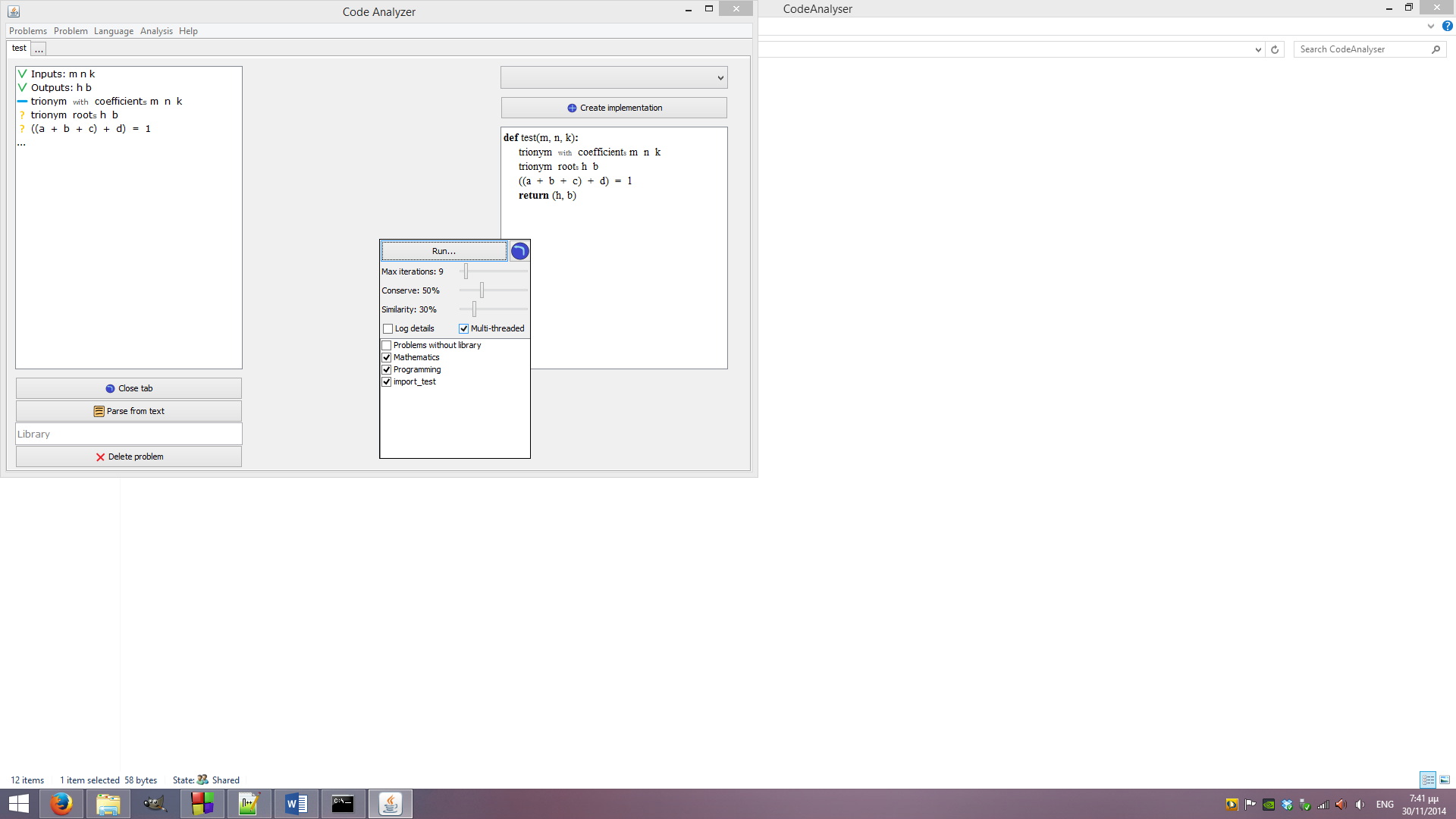
Red equations () are one-way assignments that the algorithm should always ignore when merging assignments together. When writing such a statement use instead of the equation symbol. This way of writing equations should be used for assigning results to returning variables.

**Keyboard shortcuts**

|  |  |
| --- | --- |
| **CTRL + TAB** | change tab (if the ‘…’ tab is selected, the problem selector will open) |
| **+** | create new problem if problem selector is open |
| **ENTER** | select problem in problem selector (change current problem with arrows) |
| **DEL** | delete current problem in problem selector |
| **ESC** | close problem selector |

**How to create new data**

A problem can have only another problem from the same library as a solution. Problems without a library can only have problems without a library as a solution. This measure is in place to ensure easy selection of problems, even if you have a large number of libraries.

**How to run the algorithm**

Select the option ‘Clear Implementation’ in the implementation selection box to clear the existing implementation. When you do so, the button under it will change to ‘Create implementation’. Clicking that button will open the run dialog.

In the new dialog, leave checked all libraries you want the algorithm to use and uncheck those you don’t want to use. Then click on the ‘Run…’ button to start the algorithm. To prevent possible infinite loops the algorithm also offers a limit on the maximum number of iterations. The user can change this number to any value they wish, up to the ten times the number of solved problems in the database.

There is also the option to use multiple threads for performing the analysis as well as weather all analysis details will be displayed on the log dialog. In multi-threaded execution, some details are not shown.

Finally, the ‘Similarity’ option sets the similarity level between ‘bags of words’ in order for them to be considered the same.

**Possible problems** (will create warnings)

* Multiple return statement logic may generate false program flow
* Wrong order of variable assignment with circular reference possible
* Generating intermediate variables that don’t appear in the source problem (when importing)

**Features to add**

* Analysis to display possible problems
* Add undefined returns to all (relevant) statements when importing problems
* Import function names only (will ignore function body)