

REFINITY Documentation

REFINITY is a workbench for modeling and verifying Java-based program transformation rules. It is based on KeY and Abstract Execution. Here, we describe the user interface of REFINITY, how to specify program transformation rules with pre- and postconditions using the tool, and its file format. At the end of the document, you find helpful references / scientific publications (which can also be helpful).

This documentation is under ongoing development.

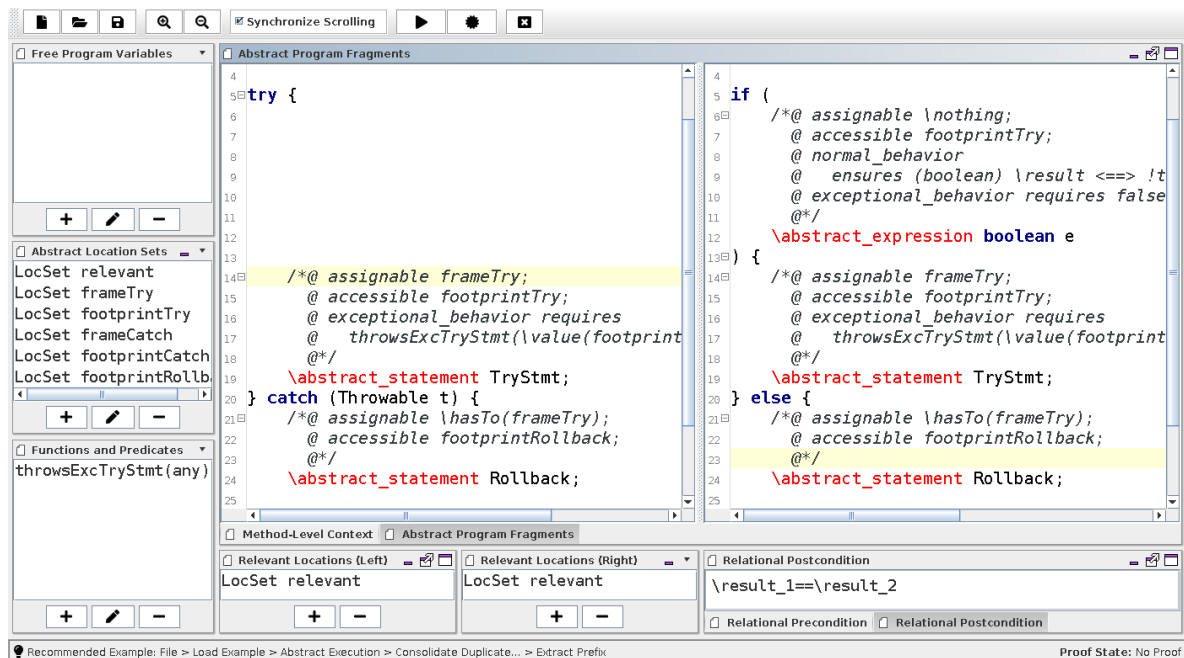
Last update: 2020-08-21.

Getting Started

REFINITY is an add-on to the [KeY program prover](#). As of Aug 2020, it is available in the ["AbstractExecution" branch](#) of KeY. It is planned to merge it into the master branch until end 2020.

You can start REFINITY from within the graphical KeY user interface by pressing the big "REFINITY" button. If you do not see this button, check that you (1) use an up-to-date KeY version with Abstract Execution, and (2) the REFINITY extension is activated: In the GUI, choose Options > Show Settings > Extensions, check AE-Relational, apply and restart.

The User Interface



The REFINITY user interface consists mainly of two panes for Abstract Program Fragments (APFs), i.e., sequences of statements with Abstract Program Elements (APEs). An APE is either an Abstract Statement (AS) or an Abstract Expression (AExp).

On the left-hand side of the user interface, symbols used in the APFs can be declared: Program variables which can be used in both programs without declaration, abstract, set-valued specification variables for I/O specifications of APEs, and abstract function / predicate symbols that can primarily be used for specifying abstract abrupt completion behavior and abstract loop

invariants.

The three compartments at the bottom serve to declare pre- and postconditions of the model.

Most GUI elements have brief tooltip help texts, so hover with the mouse over an element you do not understand and chances are good that you get help.

The functionality of the toolbar buttons are (from left to right in the order of their appearance):

- Create a new REFINITY model. Will open a new window.
- Open an existing REFINITY model (*.aer file). Opens a new window if the current file is not fresh/empty.
- Save the current REFINITY model (as an *.aer file).
- Increase font size (useful for presentations).
- Decrease font size.
- Synchronize scrolling: If you scroll in one abstract program pane, the other one follows synchronously. Especially helpful if you "align" the input and output of your transformation rule, as in the picture above.
- Start proof. Generates a KeY proof obligation, loads it into KeY and starts the automatic proof search.
- Certify proof. Load an existing *.proof file and try to apply it to the current model. Reports success if the certificate could be successfully checked.
- Exit. Closes the REFINITY *and* KeY windows.

In the next section, we explain how abstract programs are specified; afterward, we move to the specification of pre- and postconditions.

☐ Tooltips

☐ Control bar buttons

Specifying Abstract Programs

In progress.

The specification language is described in-depth in reference [2].

Declaring Proof Goals

In progress.


Proof Search and Certification




In progress.

File Format

In progress.

References

1. [Dominic Steinhöfel](#) : "REFINITY to Model and Prove Program Transformation Rules." APLAS 2020, *accepted*.

2. [Dominic Steinhöfel](#) : "Abstract Execution: Automatically Proving Infinitely Many Programs."
Darmstadt University of Technology, Germany, 2020
 3. [Dominic Steinhöfel](#) , [Reiner Hähnle](#) : "Abstract Execution." [FM 2019](#): 319-336
-

REFINITY has been developed by [Dominic Steinhöfel](#).