Version 4.4.2

Maintenance Edition

Date of diffusion : January 11th, 2017

The Atelier B is available in two versions:

- The ***Community Edition***, usable by everyone without any restriction. This version is not maintained.
- The ***Maintenance Edition***, access restricted to Atelier B 4 maintenance contract holders (corrective maintenance, anticipated access to new features/tools). Some features are specific to this version (Ada, HIA and C++ code generators, mathematical rules proof tool.

Functionality	Atelier B 4.2.1 Community Edition	Atelier B 4.4.2 Maintenance Edition
Integrated Development Environment	✓	✓
Support of B Language Project	✓	✓
Support of Event-B Language Project	✓	✓
Support of Data Validation Project	✓	✓
Editor of B and Event-B Models	✓	✓
Automatic Refiner	✓	✓
Type Checker	✓	✓
Proof Obligations Generator	✓	✓
Automatic Prover	✓	✓
Interactive Prover	✓	✓
Predicate Prover	✓	✓
C Translator C4B	✓	✓
Ada Translator (MacOS, Linux)		✓
High Integrity Ada Translator (MacOS, Linux)		✓
C++ Translator (MacOS, Linux)		✓
Mathematical Rule Validator Tool		✓

New Functionalities / Characteristics:

Maintenance Edition Atelier B 4.4.2 has been released on December 16th, 2016.

This version fixes 45 bugs and 3 improvements are included:

- Proof obligations displayed in the model editor.
- A new proof command at for *Apply Tactic*.
- A configurable timeout for the pp family proof commands.
- New functionalities for the Proof Rules Validator.

Proof Obligations Displayed in the Model Editor

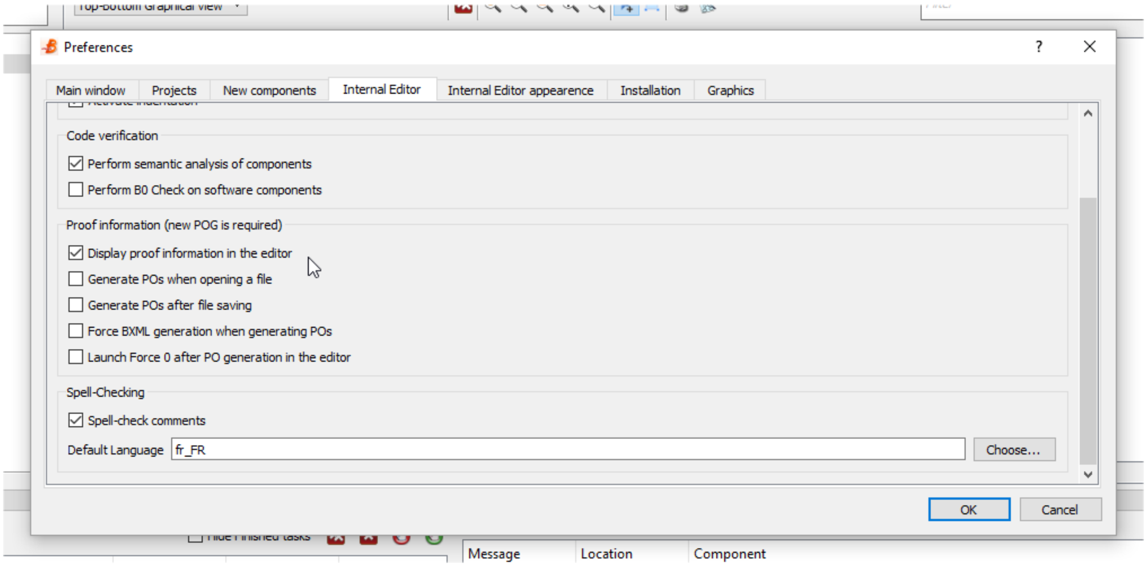
A new functionality has been developed in the editor in order to display the proof obligations in the file being edited.

This functionality has been added for the purpose of:

- Displaying the proof obligations of the component being edited.
- Displaying the proof status of the component being edited.

Enabling the Functionality

This functionality is disabled by default. To activate it, please click on the menu "Atelier B", then on "Preferences", then on the tab "Internal Editor". In the tab, search the section "Proof information" then check the box "Display proof information in the editor".



Code verification

☑ Perform semantic analysis of components

□ Perform B0 Check on software components

Proof information (new POG is required)

☑ Display proof information in the editor

□ Generate POs when opening a file

□ Generate POs after file saving

□ Force BXML generation when generating POs

□ Launch Force 0 after PO generation in the editor

Spell-Checking

☑ Spell-check comments

Default Language fr_FR

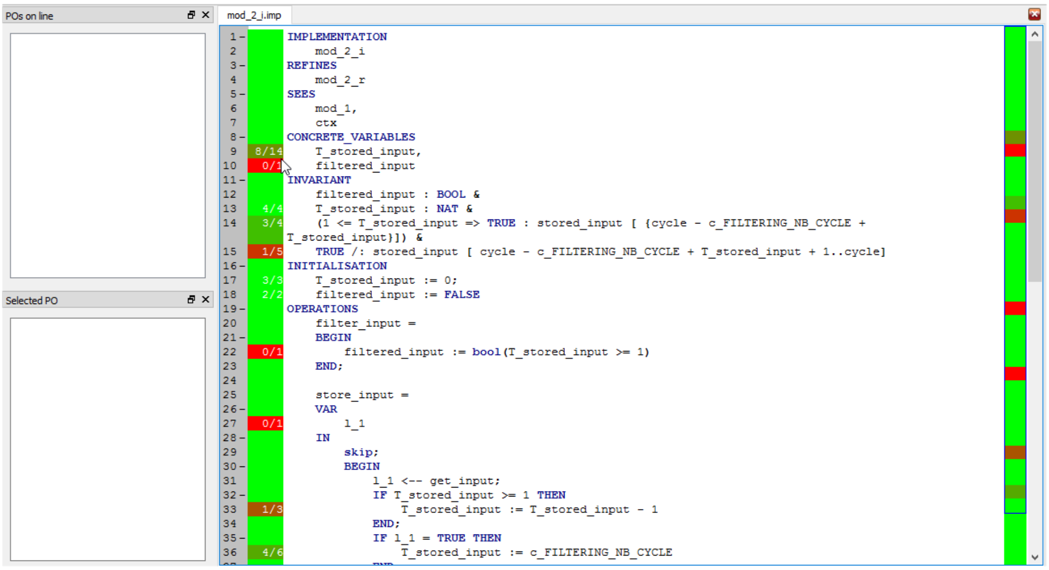
Finally, you can check the other boxes of the section for choosing when will be launched automatically the generation of proof obligations and the force 0.

It should be noted that this functionality only works with the new proof obligation generator.

Effect in the Editor

If the proof obligations have been generated, the editor will be enhanced with new features.

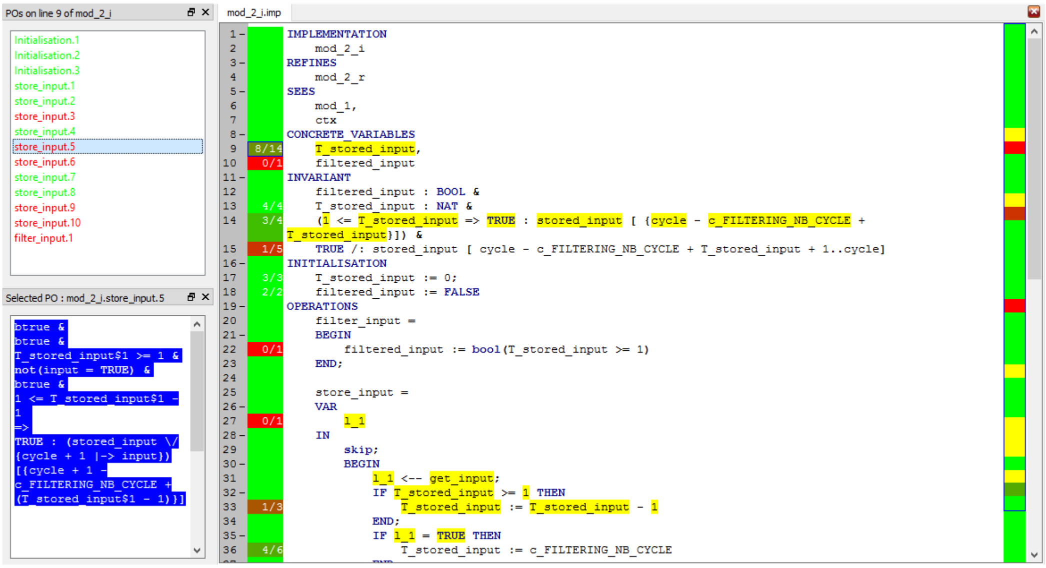
- A vertical bar to the left displaying for each line the proof status.
- A vertical bar to the right displaying the proof status of the file.



```
IMPLEMENTATION
    mod_2_i
REFINES
    mod_2_r
SEES
    mod_1,
    ctx
CONCRETE_VARIABLES
    T_stored_input,
    filtered_input
INVARIANT
    filtered_input : BOOL &
    T_stored_input : NAT &
    (1 <= T_stored_input => TRUE : stored_input [ {cycle - c_FILTERING_NB_CYCLE +
T_stored_input}]) &
    TRUE /: stored_input [ cycle - c_FILTERING_NB_CYCLE + T_stored_input + 1..cycle]
INITIALISATION
    T_stored_input := 0;
    filtered_input := FALSE
OPERATIONS
    filter_input =
    BEGIN
        filtered_input := bool(T_stored_input >= 1)
    END;

    store_input =
    VAR
        l_1
    IN
        skip;
        BEGIN
            l_1 <-- get_input;
            IF T_stored_input >= 1 THEN
                T_stored_input := T_stored_input - 1
            END;
            IF l_1 = TRUE THEN
                T_stored_input := c_FILTERING_NB_CYCLE
```

Moreover, a click on one of the lines of the left bar displays the proof obligations linked to this line. By selecting one of these proof obligations, all lines linked to this proof obligation will be highlighted.



Initialisation.1
Initialisation.2
Initialisation.3
store_input.1
store_input.2
store_input.3
store_input.4
store_input.5
store_input.6
store_input.7
store_input.8
store_input.9
store_input.10
filter_input.1

Selected PO : mod_2_i.store_input.5

```
btrue &
btrue &
T_stored_input$1 >= 1 &
not(input = TRUE) &
btrue &
1 <= T_stored_input$1 -
1
=>
TRUE : (stored_input \/
{cycle + 1 |-> input})
[{cycle + 1 -
c_FILTERING_NB_CYCLE +
(T_stored_input$1 - 1)}]
```

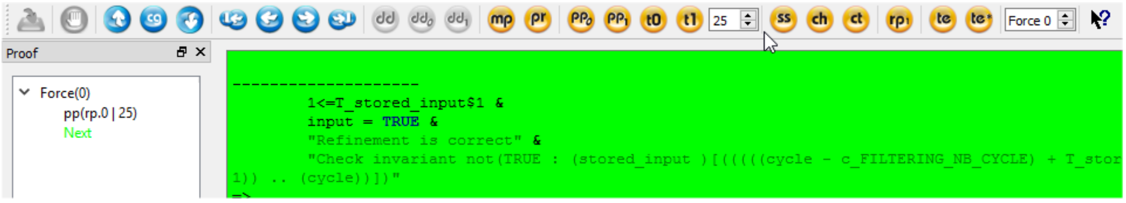
```
IMPLEMENTATION
    mod_2_i
REFINES
    mod_2_r
SEES
    mod_1,
    ctx
CONCRETE_VARIABLES
    T_stored_input,
    filtered_input
INVARIANT
    filtered_input : BOOL &
    T_stored_input : NAT &
    (1 <= T_stored_input => TRUE : stored_input [ {cycle - c_FILTERING_NB_CYCLE +
T_stored_input}]) &
    TRUE /: stored_input [ cycle - c_FILTERING_NB_CYCLE + T_stored_input + 1..cycle]
INITIALISATION
    T_stored_input := 0;
    filtered_input := FALSE
OPERATIONS
    filter_input =
    BEGIN
        filtered_input := bool(T_stored_input >= 1)
    END;

    store_input =
    VAR
        l_1
    IN
        skip;
        BEGIN
            l_1 <-- get_input;
            IF T_stored_input >= 1 THEN
                T_stored_input := T_stored_input - 1
            END;
            IF l_1 = TRUE THEN
                T_stored_input := c_FILTERING_NB_CYCLE
```

Added a Configurable Timeout for the pp Family Proof Commands

In the interactive prover, it is now possible to choose the timeout of commands of the pp family.

In the example below, the timeout of the command `pp0` is set to 25 seconds.



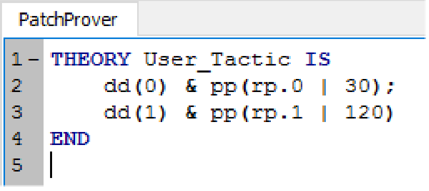
Force(0)
pp(rp.0 | 25)
Next

```
--------------------
        1<=T_stored_input$1 &
        input = TRUE &
        "Refinement is correct" &
        "Check invariant not(TRUE : (stored_input )[(((((cycle - c_FILTERING_NB_CYCLE) + T_stor
1)) .. (cycle))])"
=>
```

A New Proof Command at for Apply Tactic

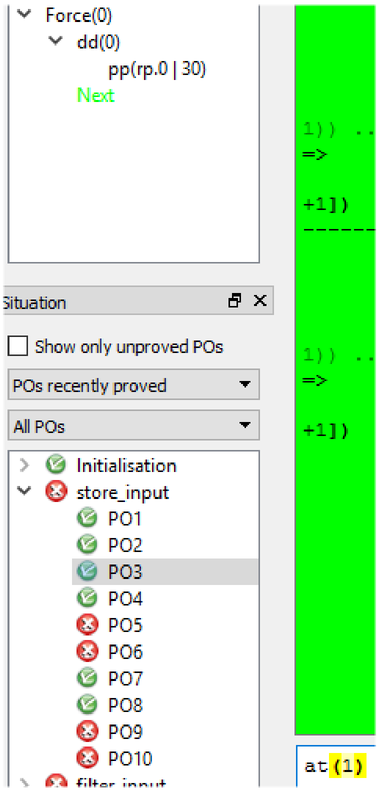
The interactive prover now has a new proof command at (for Apply Tactic). This command has a single parameter which is the index of one of the rules of the theory User_Tactic. Executing this command applies the corresponding rule.

In order to define your own tactics, you can open the PatchProver of your project and add your tactics in the theory User_Tactic.



```
THEORY User_Tactic IS
    dd(0) & pp(rp.0 | 30);
    dd(1) & pp(rp.1 | 120)
END
```

In order to use for example the first tactic, simply run the command at(1) in the interactive prover.

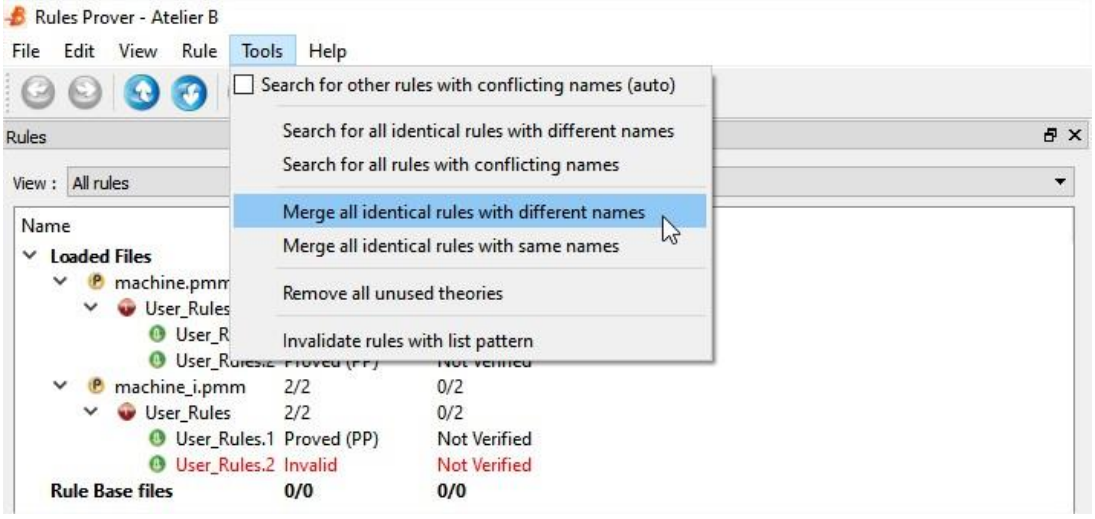


```
at(1)
```

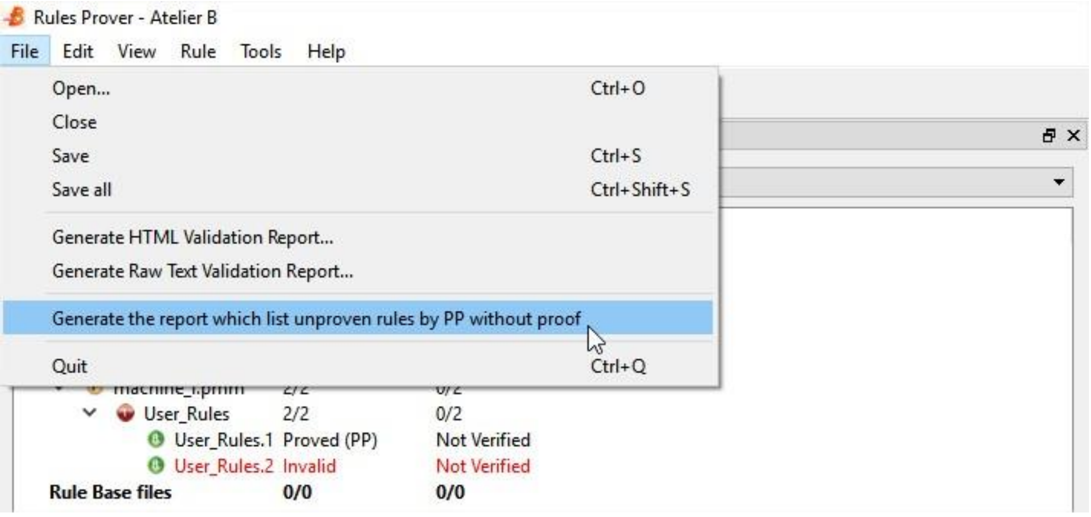
New functionalities for the Proof Rules Validator

Various functionalities have been added to the Proof Rule Validator in order accelerate the time that this activity can take.

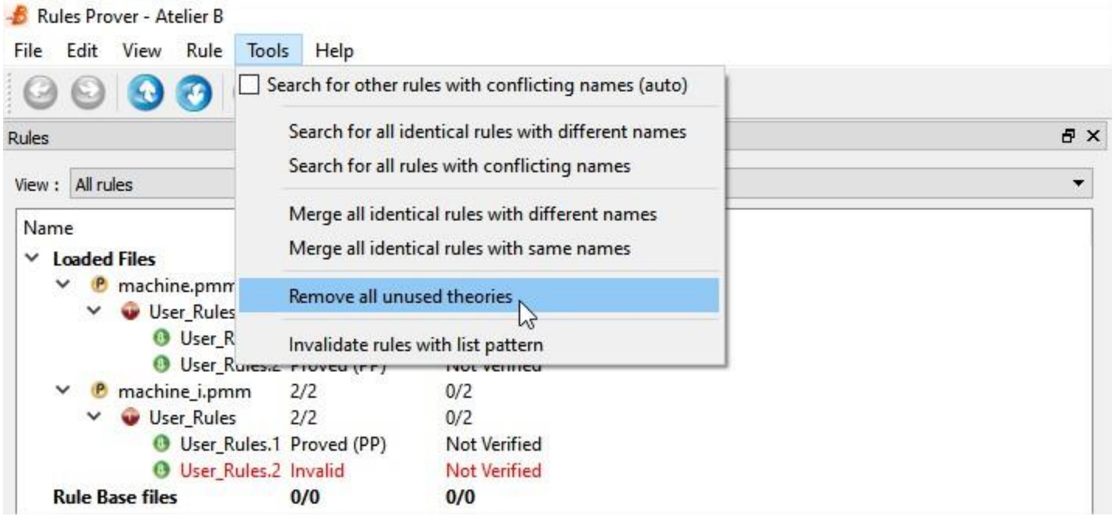
In the design mode, it is possible to automatically merge the name and the demonstration of all identical rules with different names.


Rules Prover - Atelier B
File Edit View Rule Tools Help
□ Search for other rules with conflicting names (auto)
Search for all identical rules with different names
Search for all rules with conflicting names
Merge all identical rules with different names
Merge all identical rules with same names
Remove all unused theories
Invalidate rules with list pattern
Rules
View : All rules
Name
Loaded Files
machine.pmm
User_Rules
User_R
User_Rules.2 Proved (PP) Not Verified
machine_i.pmm 2/2 0/2
User_Rules 2/2 0/2
User_Rules.1 Proved (PP) Not Verified
User_Rules.2 Invalid Not Verified
Rule Base files 0/0 0/0

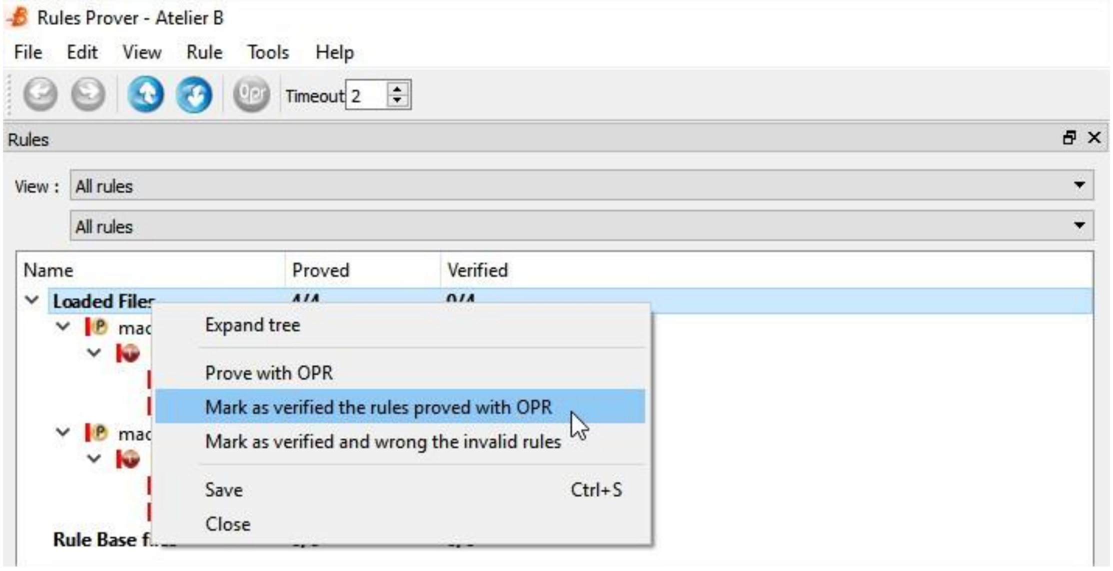
In the design mode, a report listing the rules that are not proved automatically and without demonstration can be exported in the CSV format.


Rules Prover - Atelier B
File Edit View Rule Tools Help
Open... Ctrl+O
Close
Save Ctrl+S
Save all Ctrl+Shift+S
Generate HTML Validation Report...
Generate Raw Text Validation Report...
Generate the report which list unproven rules by PP without proof
Quit Ctrl+Q
machine_i.pmm 2/2 0/2
User_Rules 2/2 0/2
User_Rules.1 Proved (PP) Not Verified
User_Rules.2 Invalid Not Verified
Rule Base files 0/0 0/0

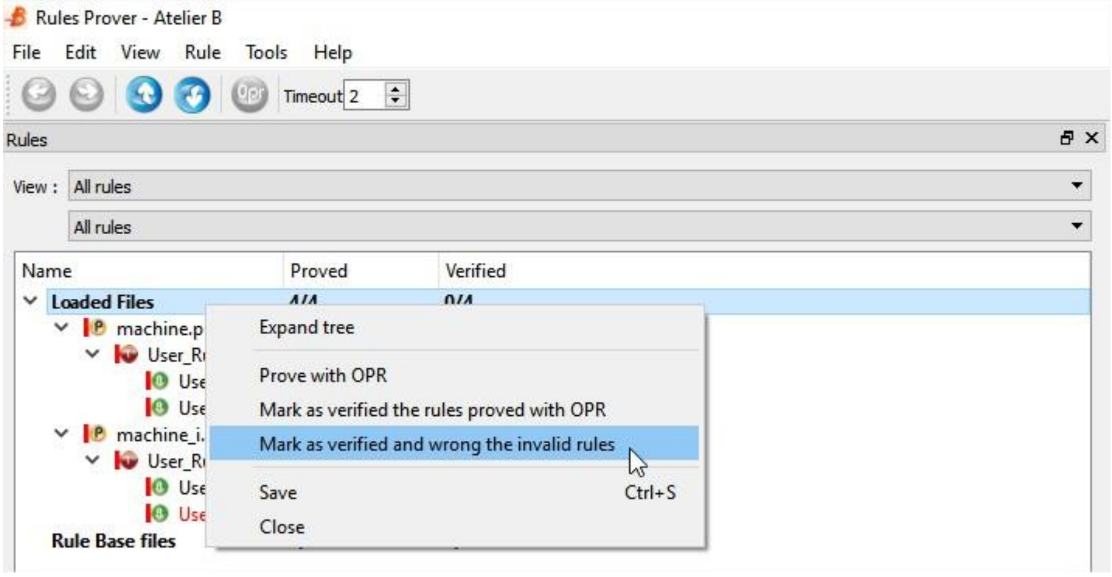
In the design mode, it is possible to remove all the unused theories.


Rules Prover - Atelier B
File Edit View Rule Tools Help
□ Search for other rules with conflicting names (auto)
Search for all identical rules with different names
Search for all rules with conflicting names
Merge all identical rules with different names
Merge all identical rules with same names
Remove all unused theories
Invalidate rules with list pattern
Rules
View : All rules
Name
Loaded Files
machine.pmm
User_Rules
User_R
User_Rules.2 Proved (PP) Not Verified
machine_i.pmm 2/2 0/2
User_Rules 2/2 0/2
User_Rules.1 Proved (PP) Not Verified
User_Rules.2 Invalid Not Verified
Rule Base files 0/0 0/0

In the verification mode, the automatically proved rules can be marked as verified.



In the verification mode, it is possible to mark as verified but wrong all the invalid rules.



In the verification mode, the shortcut Ctrl+N will mark a rule as verified and will skip to the next rule.